# jQuery简介

## 章节目标

* 了解jQuery的作用与优势
* 熟悉jQuery的环境配置
* 了解jQuery的功能

## 认识jQuery

### 从JavaScript到jQuery

JavaScript的出现使得网页和用户之间实现了一种实时的、动态的和交互的关系，使网页可以包含更多活跃的元素和更加精彩的内容。JavaScript自身存在3个弊端：

* 复杂的文档对象模型（DOM）
* 不一致的浏览器实现和便捷的开发
* 调试工具的缺乏

为了简化JavaScript的开发，一些JavaScript程序库诞生了。JavaScript程序库封闭了很多预定义的对象和实用函数，能够帮助使用者轻松地建议有高难度的交互页面。比如：Prototype、Dojo、YUI、MooTools、jQuery等等。

其中jQuery凭借其优势从众多JS库中脱颖而出，成为Web开发人员的最佳选择。

### jQuery的优势

jQuery强调的理念是Write less, do more!其优势如下：

**轻量级的JavaScript库**

jQuery非常轻巧，以1.9.1为例，开发版仅有263KB，而精简版经过UglifyJS压缩后，大小只有91K。

**强大的选择器**

jQuery支持CSS1到CSS3几乎所有的选择器，以及jQuery独创的高级复杂的选择器。另外还通过插件支持XPath选择器。

**出色的DOM操作**

jQuery封装了大量常用的DOM操作，使开发都在编写DOM操作的相关程序时更加方便。jQuery可以轻松的完成原本非常复杂的操作。

**可靠的事件处理**

jQuery的事件处理机制了JavaScript的事件处理函数的精华，使得jQuery在处理事件绑定的时候相当可靠。在预留退路、循环渐进以及非入侵式编程理想方面，jQuery也做得非常不错。

**完善的Ajax**

jQuery将所有的Ajax操作封装到一个函数$.ajax()里，使得开发者处理Ajax的时候能够专心处理业务逻辑而无需关心复杂的浏览器兼容性和XMLHttpRequest对象的创建和使用问题

**不污染的顶级变量**

jQuery只建立一个名为jQuery的对象，其所有的函数方法都在这个对象之下。其别名$也可以随时交出操作权，绝对不会污染其它的对象。使得jQuery可以与其它的JS库共存，在项目中放心地引用而不需要担心后期的冲突。

**浏览器兼容性**

jQuery能够在IE6.0+、FF3.6+、Safari 5.0+、Opera和Chrome等浏览器下正常运行。jQuery同时修复了一些浏览器之间的差异，使得开发都不必在项目中建立浏览器兼容库。注意：从jQuery2.0开始，不再支持IE8.0以下以版本，因此我们在课程中使用jQuery1.9.2版本。

**链式操作**

jQuery 中最有特色的莫过于它的链式操作方式一一即对发生在同一个jQuery 对象上的一组动作，可以直接连写而无需重复获取对象。这一特点使jQuery 的代码无比优雅。

**隐式迭代**

当用jQuery 找到带有“.myClass”类的全部元素，然后隐藏它们时，无需循环遍历每一个返回的元素。相反， jQuery 里的方法都被设计成自动操作对象集合，而不是单独的对象，这使得大量的循环结构变得不再必要，从而大幅地减少了代码量。

**丰富的插件支持**

jQuery 的易扩展性，吸引了来自全球的开发者来编写jQuery 的扩展插件。目前已经有成百上千的官方插件支持，而且还不断有新插件面世。

**完善的文档**

jQuery 的文档非常丰富，不管是英文文档，还是中文文档。我们也在长期更新着jQuery 的中文文档。

**开源**

jQuery 是一个开源的产品，任何人都可以自由地使用并提出改进意见。

## 简单的jQuery使用

### jQuery环境配置

**获取jQuery最新版本**

进入jQuery 的官方网站<http://jquery.com/>。

为了兼容IE8.0及以下版本，本课程以jQuery1.9.1作为授课版本。

**jQuery库类型说明**

jQuery库分为两种类型：开发版、精简版。其中，开发版是完整无压缩的版本，主要用于测试、学习和开发。精简版经过UglifyJS压缩，主要应用于产品和运行项目中。

**环境配置**

jQuery不需要安装，把下载的jQuery.js文件放到网站任何一个公共位置，并在需要使用jQuery的页面中引入该库文件即可。

**引入jQuery库**

在编写的页面代码中，<head>标签内引入jQuery库，就可以使用jQuery了。例如：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-01.html</title>    <meta name=*"keywords"* content=*"keyword1,keyword2,keyword3"*>  <meta name=*"description"* content=*"this is my page"*>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>    <!—引入jQuery库-->  <script src=*”jquery/jquery-1.9.1.js”*></script>  </head>    <body>  This is my HTML page. <br>  </body>  </html> |

### 简单的jQuery代码

在开始编写第1 个jQuery 程序之前，首先应该明确一点，在jQuery 库中，$就是jQuery 的一个简写形式，例如$(“#foo”)和jQuery(“#foo”)是等价的，$.ajax 和jQuery.ajax 是等价的。

Hello World程序：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-01.html</title>    <meta name=*"keywords"* content=*"keyword1,keyword2,keyword3"*>  <meta name=*"description"* content=*"this is my page"*>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>    <!-- 引入jQuery库-->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){ // 页面加载完成事件  alert("Hello World!"); // 弹出提示框  });  </script>  </head>    <body>  This is my HTML page. <br>  </body>  </html> |

在上面的代码中有一个陌生的代码片段，如下：

|  |
| --- |
| $(document).ready(**function**(){ // 页面加载完成事件  alert("Hello World!"); // 弹出提示框  }); |

这段代码的$(document).ready()方法作用类似于传统JavaScript 中的window.onload 方法，不过与window.onload 还是有些区别。$(document).ready()方法当网页中所有的DOM结构绘制完成后执行，此时与DOM元素相关联的资源可能并没有加载完成。

$(document).ready()方法可以简写成$()或$().ready()，例如：

|  |
| --- |
| $(**function**(){ // 页面加载完成事件  alert("Hello World!"); // 弹出提示框  }); |

## jQuery对象和DOM对象

第一次学习 jQuery，经常分辨不清哪些是 jQuery 对象、哪些是 DOM 对象， 因此需要重点了解jQuery 对象和DOM 对象以及它们之间的关系。

### DOM 对象

DOM (Document Object Model ，文档对象模型)，每一份DOM 都可以表示成一棵树。下面来构建一个非常基本的网页，网页代码如下：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-03.html</title>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  </head>    <body>  <h3>DOM树</h3>  <p title=*"选择你喜欢的水果"*>你最喜欢的水果是？</p>  <ul>  <li>苹果</li>  <li>桔子</li>  <li>香蕉</li>  </ul>  </body>  </html> |

可以把上面的HTML 结构描述为一棵DOM 树，如图所示。

html

head

body

meta

title

p

h3

ul

li

li

li

在这棵DOM 树中，＜h3＞、＜p＞、＜ul＞以及＜ul＞的3 个＜Ii＞子节点都是DOM 元素节点。可以通过JavaScript 中的getElementsByTagName 或者getElementByld 来获取元素节点。像这样得到的DOM元素就是DOM 对象。DOM 对象可以使用JavaScript 中的方法，例如：

|  |
| --- |
| **var** domObj = document.getElementById("id"); // 获取DOM对象  **var** objHTML = domObj.innerHTML; // 使用JS获取对象的属性 |

### jQuery 对象

jQuery 对象就是通过jQuery 包装DOM 对象后产生的对象。jQuery 对象是jQuery 独有的。如果一个对象是jQuery 对象，那么就可以使用jQuery 里的方法。例如：

|  |
| --- |
| $("#foo").html(); // 获取id为foo元素的html代码，html()是jQuery中的方法  // 这段代码相当于  document.document.getElementById("foo").innerHTML; |

### jOurery 对象和DOM 对象的相互转换

在讨论jQurey 对象和DOM 对象的相互转换之前，先约定好定义变量的风格。如果获取的对象是jQuery 对象，那么在**变量前面加上$**，例如：

|  |
| --- |
| **var** $variable = jQuery对象;  **var** variable = DOM对象; |

**jQuery 对象转成DOM 对象**

jQuery 对象不能使用DOM 中的方法，但如果对jQuery 对象所提供的方法不熟悉，或者jQuery没有封装想要的方法，不得不使用DOM 对象的时候，有以下两种处理方法。

jQuery提供了两种方法将一个jQuery 对象转换成 DOM 对象， 即［index］和 get ( index ）。

(1) jQuery 对象是一个类似数组的对象，可以通过［index］的方法得到相应的DOM 对象。

jQuery 代码如下：

|  |
| --- |
| **var** $cr = $("#cr"); // jQuery对象  **var** cr = $cr[0]; // 转化为DOM对象 |

(2)另一种方法是jQuery本身提供的，通过get(index)方法得到相应的DOM对象。

jQuery 代码如下：

|  |
| --- |
| **var** $cr = $("#cr"); // jQuery对象  **var** cr = $cr.get(0); // 转化为DOM对象 |

**DOM对象转成jQuery对象**

对于一个DOM对象，只需要使用$()把DOM对象包装起来，就可以获得一个jQuery对象了。方式为：$(DOM对象)。

jQuery 代码如下：

|  |
| --- |
| **var** cr = doucument.getElementById("cr"); // DOM对象  **var** $cr = $(cr); // jQuery对象 |

## 本章作业

完成以下页面：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-04.html</title>    <meta name=*"keywords"* content=*"keyword1,keyword2,keyword3"*>  <meta name=*"description"* content=*"this is my page"*>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>    <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(**function**(){  $("button").click(**function**(){  **var** v = $("input[name='fruit']:checked").val();  alert("你选择的是："+v);  });  });  </script>  </head>    <body>  <h3>你喜欢的水果是？</h3>  <input type=*"radio"* name=*"fruit"* value=*"苹果"*/>苹果</br>  <input type=*"radio"* name=*"fruit"* value=*"桔子"*/>桔子</br>  <input type=*"radio"* name=*"fruit"* value=*"香蕉"*/>香蕉</br>  <button>提交</button>  </body>  </html> |

# jQuery选择器

## 章节目标

* 掌握jQuery选择器的使用
* 能够使用jQuery选择器对DOM元素进行基本操作

## 基本选择器

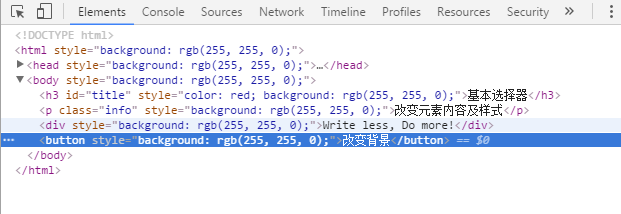
基本选择器是JQuery 中最常用的选择器，也是最简单的选择器，它通过元素id 、class 和标签名等来查找DOM 元素。在网页中，每个id 名称只能使用一次， class 允许重复使用。

|  |  |  |
| --- | --- | --- |
| **选择器** | **实例** | **选取** |
| \* | $("\*") | 所有元素 |
| #*id* | $("#lastname") | id="lastname" 的元素 |
| .*class* | $(".intro") | 所有 class="intro" 的元素 |
| *element* | $("p") | 所有 <p> 元素 |
| .*class*.*class* | $(".intro.demo") | 所有 class="intro" 且 class="demo" 的元素 |

示例：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-01.html</title>    <meta name=*"keywords"* content=*"keyword1,keyword2,keyword3"*>  <meta name=*"description"* content=*"this is my page"*>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>    <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  // id选择器  $("#title").css("color", "red");  // class选择器  $(".info").html("改变元素内容及样式");  // 标签选择器  $("button").click(**function**(){  // 通配选择器  $("\*").css("background", "#FFFF00");  });  });  </script>  </head>    <body>  <h3 id=*"title"*>基本选择器</h3>  <p class=*"info"*></p>  <button>改变背景</button>  </body>  </html> |

运行结果：



## 层次选择器

### 层次选择器

如果想通过DOM 元素之间的层次关系来获取特定元素，例如后代元素、子元素、相邻元素和同辈元素等，那么层次选择器是一个非常好的选择。

|  |  |  |
| --- | --- | --- |
| **选择器** | **实例** | **选取** |
| $(“ancestor descendant”) | $("div span") | <div>里的所有<span>元素 |
| $(“parent>child”) | $("div>span") | <div>下<span>子元素 |
| $(“prev+next”) | $(".one+div") | 选取class为one的下一个<div>同级元素 |
| $(“prev~siblings”) | $(".one~div") | 选取class为one后所有的<div>同级元素 |

**注意**$("div span")和$("div>span")是有区别的，$("div span")会选取div里所有的<span>元素，而$("div>span")只会选取<div>直属的<span>子元素。

示例：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-02.html</title>    <meta name=*"keywords"* content=*"keyword1,keyword2,keyword3"*>  <meta name=*"description"* content=*"this is my page"*>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>    <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  $("#btn").click(**function**(){  $("\*").removeAttr("style");  });  $("#btn1").click(**function**(){  $("body ul").css("background", "#FF6600");  });    $("#btn2").click(**function**(){  $("body>ul").css("background", "#FFFF00");  });    $("#btn3").click(**function**(){  $("#menu1>ul").css("background", "#0099FF");  });    $("#btn4").click(**function**(){  $("#menu1+li").css("background", "#FF00FF");  });    $("#btn5").click(**function**(){  $("#menu2~li").css("background", "#F0000F");  });  });  </script>  </head>    <body>  <h3 id=*"title"*>层次选择器</h3>  <ul>  <li id=*"menu1"*>I  <ul>  <li id=*"menu2"*>1</li>  <li>2</li>  <li>3</li>  </ul>  </li>  <li>II  <ul>  <li>1</li>  <li>2</li>  </ul>  </li>  <li>III  <ul>  <li>1</li>  </ul>  </li>  </ul>  <button id=*"btn"*>恢复</button></br>  $("body ul")<button id=*"btn1"*>改变背景</button></br>  $("body>ul")<button id=*"btn2"*>改变背景</button></br>  $("#menu1>ul")<button id=*"btn3"*>改变背景</button></br>  $("#menu1+li")<button id=*"btn4"*>改变背景</button></br>  $("#menu2~li")<button id=*"btn5"*>改变背景</button></br>  </body>  </html> |

### jQuery元素选择方法

在层次选择器中，第1 个和第2 个选择器比较常用，而后面两个因为在jQuery 里可以用更加简单的方法代替，所以使用的几率相对少些。

可以使用next（）方法来代替$（＇prev + next＇）选择器，nextAll（）方法来代替$（＇prev～siblings’）选择。同时还有siblings()方法来补充nextAll()方法的不足。

|  |  |
| --- | --- |
| **选择方法** | **说明** |
| next() | 等价于$(“prev + next”) |
| nextAll() | 等价于$(“prev～siblings”) |
| siblings() | 类似$(“prev～siblings”)，但无关前后，选择所有同辈节点 |
| children() | 获取匹配元素的所有子元素 |
| parent() | 获取匹配元素的父元素 |
| parents() | 获取匹配元素的所有父元素 |

示例：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-03.html</title>    <meta name=*"keywords"* content=*"keyword1,keyword2,keyword3"*>  <meta name=*"description"* content=*"this is my page"*>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>    <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  $("#btn").click(**function**(){  $("\*").removeAttr("style");  });  $("#btn1").click(**function**(){  $("#menu1+li").css("background", "#FF6600");  });    $("#btn2").click(**function**(){  $("#menu1").next().css("background", "#FFFF00");  });    $("#btn3").click(**function**(){  $("#menu2~li").css("background", "#0099FF");  });    $("#btn4").click(**function**(){  $("#menu2").nextAll().css("background", "#FF00FF");  });    $("#btn5").click(**function**(){  $("#menuII").siblings().css("background", "#F0000F");  });  });  </script>  </head>    <body>  <h3 id=*"title"*>层次选择器</h3>  <ul>  <li id=*"menu1"*>I  <ul>  <li id=*"menu2"*>1</li>  <li>2</li>  <li>3</li>  </ul>  </li>  <li id=*"menuII"*>II  <ul>  <li>1</li>  <li>2</li>  </ul>  </li>  <li>III  <ul>  <li>1</li>  </ul>  </li>  </ul>  <button id=*"btn"*>恢复</button></br>  $("#menu1+li")<button id=*"btn1"*>改变背景</button></br>  next()<button id=*"btn2"*>改变背景</button></br>  $("#menu2~li")<button id=*"btn3"*>改变背景</button></br>  nextAll()<button id=*"btn4"*>改变背景</button></br>  siblings()<button id=*"btn5"*>改变背景</button></br>  </body>  </html> |

## 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，过滤规则与css 中的伪类选择器语法相同，即选择器都以一个冒号（：）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、子元素过滤和表单对象属性过滤选择器。

|  |  |  |
| --- | --- | --- |
| **选择器** | **实例** | **选取** |
| :first | $("p:first") | 第一个 <p> 元素 |
| :last | $("p:last") | 最后一个 <p> 元素 |
| :even | $("tr:even") | 所有偶数 <tr> 元素 |
| :odd | $("tr:odd") | 所有奇数 <tr> 元素 |
| :eq(index) | $("ul li:eq(3)") | 列表中的第四个元素（index 从 0 开始） |
| :gt(no) | $("ul li:gt(3)") | 列出 index 大于 3 的元素 |
| :lt(no) | $("ul li:lt(3)") | 列出 index 小于 3 的元素 |
| :not(selector) | $("input:not(:empty)") | 所有不为空的 input 元素 |
| :has(selector) | $("div:has(p)") | 所有含有<p>元素的<div>元素 |
| :header | $(":header") | 所有标题元素 <h1> - <h6> |
| :animated |  | 所有动画元素 |
| :contains(text) | $(":contains('W3School')") | 包含指定字符串的所有元素 |
| :empty | $(":empty") | 无子（元素）节点的所有元素 |
| :hidden | $("p:hidden") | 所有隐藏的 <p> 元素 |
| :visible | $("table:visible") | 所有可见的表格 |
| [attribute] | $("[href]") | 所有带有 href 属性的元素 |
| [attribute=value] | $("[href='#']") | 所有 href 属性的值等于 "#" 的元素 |
| [attribute!=value] | $("[href!='#']") | 所有 href 属性的值不等于 "#" 的元素 |
| [attribute$=value] | $("[href$='.jpg']") | 所有 href 属性的值包含以 ".jpg" 结尾的元素 |

示例：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-04.html</title>    <meta name=*"keywords"* content=*"keyword1,keyword2,keyword3"*>  <meta name=*"description"* content=*"this is my page"*>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>    <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  // 第一个p标签  $("p:first").css("color", "red");    // 最后一个p标签  $("p:last").css("color", "blue");    // 偶数p标签，从0开始计数  $("p:even").css("background", "yellow");    // 下标为3的p标签，从0开始计数  $("p:eq(3)").css("background", "#FF0000");    // 所有的包含header的标签，任何一级子节点为H都可以  $(":header").css("text-decoration","underline");    // 包含'！'字符串的p标签  $("p:contains('！')").css("font-weight","bold");    });  </script>  </head>    <body>  <div>  <div><h3>茅屋为秋风所破歌</h3></div>  <div><h7>杜甫</h7></div>  <div>  <p>安得广厦千万间，</p>  </div>  <div>  <p>大庇天下寒士俱欢颜，</p>  </div>  <div>  <p>风雨不动安如山！</p>  </div>  <div>  <p>呜呼！</p>  </div>  <div>  <p>何时眼前突兀见此屋，</p>  </div>  <div>  <p>吾庐独破受冻死亦足！</p>  </div>  </div>  </body>  </html> |

## 表单选择器

为了使用户能够更加灵活地操作表单，jQuery中专门加入了表单选择器。利用表单选择器，能够极其方便地获取到表单的某个或某些类型的元素。

|  |  |  |
| --- | --- | --- |
| **选择器** | **实例** | **选取** |
| :input | $(":input") | 所有 <input> 元素 |
| :text | $(":text") | 所有 type="text" 的 <input> 元素 |
| :password | $(":password") | 所有 type="password" 的 <input> 元素 |
| :radio | $(":radio") | 所有 type="radio" 的 <input> 元素 |
| :checkbox | $(":checkbox") | 所有 type="checkbox" 的 <input> 元素 |
| :submit | $(":submit") | 所有 type="submit" 的 <input> 元素 |
| :reset | $(":reset") | 所有 type="reset" 的 <input> 元素 |
| :button | $(":button") | 所有 type="button" 的 <input> 元素 |
| :image | $(":image") | 所有 type="image" 的 <input> 元素 |
| :file | $(":file") | 所有 type="file" 的 <input> 元素 |

示例：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-05.html</title>    <meta name=*"keywords"* content=*"keyword1,keyword2,keyword3"*>  <meta name=*"description"* content=*"this is my page"*>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>    <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){    });  </script>  </head>    <body>  <h3>表单选择器</h3>  <form id=*"from1"*>  <input type=*"button"* value=*"button"*/></br>  <input type=*"checkbox"* name=*"c"*/>1  <input type=*"checkbox"* name=*"c"*/>2  <input type=*"checkbox"* name=*"c"*/>3</br>  <input type=*"file"*/></br>  <input type=*"hidden"* /><div style="display:*none*">test</div></br>  <input type=*"image"* /></br>  <input type=*"password"* /></br>  <input type=*"radio"* name=*"a"* />1  <input type=*"radio"* name=*"a"* />2  <input type=*"radio"* name=*"a"* />3</br>  <input type=*"reset"* /></br>  <input type=*"submit"* value=*"提交"* /></br>  <input type=*"text"* /></br>  <select><option>Option</option></select></br>  <textarea></textarea></br>  <button>Button</button>  </form>  </body>  </html> |

如果想得到表单内表单元素的个数， 代码如下：

|  |
| --- |
| // 获取表单元素个数  $("#form1 :input").length; |

如果想得到表单内单行文本框的个数， 代码如下：

|  |
| --- |
| // 获取表单单行文本框个数  $("#form1 :text").length; |

如果想得到表单内密码框的个数， 代码如下：

|  |
| --- |
| // 获取表单内密码框个数  $("#form1 :password").length; |

## 小结

本章详细讲解了jQuery 中的各种类型的选择器。选择器是行为与文档内容之间连接的纽带，选择器的最终目的就是能够轻松地找到文档中的元素。

在本章列举了的简单例子，介绍了jQuery 选择器，并用所学的jQuery 选择器以及隐式迭代的特性将例子进行改写。此外还讲解了选择器中的一些注意事项，希望能引起初学者的注意。最后以某网站上一个品牌列表作为例子，加深读者对jQuery 选择器用法的理解。

# DOM操作

## 章节目标

* 掌握jQuery节点查找操作
* 掌握节点的创建、插入、删除、替换等操作
* 掌握节点属性的获取与设置
* 掌握样式的相关操作

## 查找节点

首先构建一个非常基本的网页，网页代码如下：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-03.html</title>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>  <style type=*"text/css"*>  *.myclass1*{  color:*red*;  }  *.myclass2*{  background:*yellow*;  }  </style>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  </head>    <body>  <h3>DOM树</h3>  <p title=*"选择你喜欢的水果"*>你最喜欢的水果是？</p>  <ul>  <li>苹果</li>  <li>桔子</li>  <li>香蕉</li>  </ul>  </body>  </html> |

可以把上面的HTML 结构描述为一棵DOM 树，如图所示。

html

head

body

meta

title

p

h3

ul

li

li

li

使用jQuery在文档树上查找节点非常容易，可通过jQuery选择器来完成。

### 查找元素节点

通过jQuery选择器，直接可以获取到节点元素。

|  |
| --- |
| // 查找节点元素  **var** $li = $("ul li:eq(1)"); // 获取<ul>中的第2 个<li>节点  **var** li\_txt = $li.text(); // 获取节点元素的文本内容  alert(li\_txt); // 显示文本内容 |

### 查找节点属性

获取到需要的元素节点后，可以使用attr()方法来获取它的各种属性值。attr()方法的参数可以是一个，也可以是两个，当参数为一个时，是要查询的属性名字。

|  |
| --- |
| // 查找属性节点  **var** $para = $("p"); // 获取<p>节点  **var** p\_txt = $para.attr("title"); // 获取<p>节点的属性title  alert(p\_txt); |

## 创建节点

在DOM 操作中，常常需要动态创建HTML内容，使文档在浏览器里的呈现效果发生变化，并且达到各种各样的人机交互的目的。

### 创建元素节点

例如要创建两个<li>节点，并且需要把它们作为<ul>节点的子节点。则需要两个步骤：

第一步：创建两个<li>节点。

第二步：将这两个节点插入到<ul>中。

|  |
| --- |
| // 创建节点  **var** $li\_1 = $("<li>菠萝</li>"); // 创建一个<li>元素  **var** $li\_2 = $("<li title='梨子'>梨子</li>"); // 创建一个<li>元素  // 插入节点  $("ul").append($li\_1);  $("ul").append($li\_2); |

动态创建的新元素节点不会被自动添加到文档中，而是需要使用append()等方法将其插入文档中。创建节点时，必须使用标准和XHTML格式。

## 插入节点

除了前面使用的append()，在jQuery 中还提供了其他几种插入节点的方法：

|  |  |
| --- | --- |
| **方法** | **描述** |
| append() | 向每个匹配的元素内部追加内容 |
| appendTo() | 将所有匹配的元素追加到指定的元素中。  实际上，使用该方法是颠倒了常规的  $(A).append(B）的操作，即不是将B 追加到  A 中，而是将A 追加到B 中 |
| prepend() | 向每个匹配的元素内部前置内容 |
| prependTo() | 将所有匹配的元素前置到指定的元素中。  实际上，使用该方法是颠倒了常规的  $(A).prepend(B）的操作，即不是将B 前置到  A 中，而是将A 前置到B 中 |
| after() | 在每个匹配的元素之后插入内容 |
| insertAfter() | 将所有匹配的元素插入到指定元素的后  面。实际上，使用该方法是颠倒了常规的  $(A).after(B）的操作，即不是将B 插入到A  后面，而是将A 插入到B 后面 |
| before() | 在每个匹配元素之前插入内容 |
| insertBefore() | 将所有匹配的元素插入到指定的元素的前面，实际上，使用该方法是颠倒了常规的$(A).before(B）的操作，即不是将B 插入到A前面，而是将A 插入到B 前面 |

示例：

|  |
| --- |
| //插入节点  append()  // $("p").append("<b>你好</b>");  appendTo()  // $("<b>吗</b>").appendTo("p");  prepend()  // $("p").prepend("<b>问：</b>");  prependTo()  // $("<b>请</b>").prependTo("p");  after()  // $("li").after("<b>^\_^</b>");  insertAfter()  // $("<b>-\_-</b>").insertAfter("li");  before()  // $("li").before("<b>^o^</b>");  insertBefore()  // $("<b>>﹏<</b>").insertBefore("li"); |

## 删除节点

如果文档中某一个元素多余，那么应将其删除。jQuery 提供了三种删除节点的方法，即remove(), detach()和empty()。

remove()方法从DOM 中删除所有匹配的元素，传入的参数用于根据jQuery 表达式来筛选元素。当某个节点用remove（）方法删除后，该节点所包含的所有后代节点将同时被删除。这个方法的返回值是一个指向己被删除的节点的引用，因此可以在以后再使用这些元素。

|  |
| --- |
| // 删除节点  $("ul").remove(); |

detach() 和remove()一样，也是从DOM 中去掉所有匹配的元素。但需要注意的是，这个方法不会把匹配的元素从jQuery 对象中删除，因而可以在将来再使用这些匹配的元素。与remove()不同的是，所有绑定的事件、附加的数据等都会保留下来。

empty()方法并不是删除节点，而是清空节点，它能清空元素中的所有后代节点。

## 复制节点

如果要复制一个<li>元素，可以使用cloneO方法来完成。在clone()方法可以不带参数，也可以传递了一个参数true，它的含义是复制元素的同时复制元素中所绑定的事件。

|  |
| --- |
| // 复制节点  $("li").clone().appendTo("ul"); |

## 替换节点

如果要替换某个节点， jQuery 提供了相应的方法，即replaceWith()和replaceAll()。

replaceWith()方法的作用是将所有匹配的元素都替换成指定的HTML或者DOM 元素。replaceAll()方法与replaceWith（）方法的作用相同，只是颠倒了replaceWith（）操作。

|  |
| --- |
| // 替换节点  $("p").replaceWith("<strong>你最不喜欢的水果是？</strong>"); |

注意：如果在替换之前，元素已经绑定了事件，替换后原绑定事件将会一同消失，因此需要在新元素上重新绑定事件。

## 包裹节点

如果要将某个节点用其他标记包裹起来， jQuery 提供了相应的方法，即wrap（）。该方法对于需要在文档中插入额外的结构化标记非常有用，而且它不会破坏原始文档的语义。

|  |
| --- |
| // 包裹节点  $("p").wrap("<b></b>"); |

wrapAll()方法该方法会将所有匹配的元素用一个元素来包裹。 它不同于wrap()方法， wrap()方法是将所有的元素分别进行包裹。

|  |
| --- |
| // 包裹节点  $("li").wrap("<b></b>");  $("li").wrapAll("<b></b>"); |

wraplnner()方法将每一个匹配的元素的子内容(包括文本节点)用其他结构化的标记包裹起来。例如可以使用它来包裹<p>标签的子内容。

|  |
| --- |
| // 包裹节点  $("p").wrapInner("<b></b>"); |

## 属性操作

在jQuery 中，用attr（）方法来获取和设置元素属性， removeAttr（）方法来删除元素属性。

### 获取属性和设置属性

如果要获取＜p＞元素的属性title，那么只需要给attr（）方法传递一个参数，即属性名称。

如果要设置＜p＞元素的属性title 的值，也可以使用同一个方法，不同的是，需要传递两个参数即属性名称和对应的值。

如果需要一次性为同一个元素设置多个属性，可以使用JSON格式的数据作为参数。

|  |
| --- |
| // 属性操作  **var** $para = $("p");  // 获取属性  **var** p\_txt = $para.attr("title");  alert(p\_txt);  // 设置属性  $("p").attr("title", "多吃水果");  // 多属性设置  $("p").attr({"align": "center", "title":"多喝热水"});// JSON格式 |

### 删除属性

在某些情况下，需要删除文档中某个元素的特定属性，可以使用removeAttr（）方法来完成该任务。

|  |
| --- |
| // 删除属性  $("p").removeAttr("align", "center"); |
|  |

## 样式操作

### 设置样式和获取样式

class 也是节点元素的属性，因此获取class 和设置class 都可以使用attr()方法来完成。

|  |
| --- |
| // 设置样式  $("p").attr("class", "myclass1");  // 获取样式  **var** p\_class = $("p").attr("class");  alert(p\_class); |

### 追加样式

直接使用attr()方法设置class属性，会将原来的class属性替换掉。如果在添加新的样式时，还需要保留原有样式，可以使用addClass()方法。

|  |
| --- |
| // 追加样式  $("p").addClass("myclass2"); |

**注意：**当一个节点元素包含了多个class样式时，将按照以下方式进行处理：

(1) 如果给一个元素添加了多个 class 值， 那么就相当于合并了它们的样式。

(2) 如果有不同的 class 设定了同一样式属性， 则后者覆盖前者。

### 移除样式

如果要删除class某个值，可以使用removeClass()方法完成，它的作用是从匹配的元素中删除全部或者指定的class。不传递参数时，删除所有；给定参数时只删除指定的class。

|  |
| --- |
| // 移除样式  $("p").removeClass(); // 移除所有样式  $("p").removeClass("myclass1"); // 移除myclass2样式 |

### 判断是否含有某个样式

hasClass()方法可以用来判断元素中是否包含某个class，如果有则返回true，否则返回false。

|  |
| --- |
| // 判断样式  $("p").hasClass("myclass1"); |

## 设置和获取HTML、文本和值

html()方法类似于JavaScript中的innerHTML属性，可用来读取或者设置某个元素中的HTML内容。

text()方法类似于JavaScript中的innerText属性，可以用来读取或者设置某个元素中的文本内容。

val()方法类似于JavaScript中的value属性，可以用来设置和获取元素的值。无论元素是文本框，下拉列表，还是单选框，它都可以返回元素的值。如果元素为多选，则返回一个包含所有选择的值的数组。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-02.html</title>    <meta name=*"keywords"* content=*"keyword1,keyword2,keyword3"*>  <meta name=*"description"* content=*"this is my page"*>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>    <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  // 设置HTML  $("#title").html("<h3>用户登录</h3>");  // 获取HTML  alert($("#title").html());  $("#username").val("请输入密码");    // 焦点事件  $("#username").focus(**function**(){  **var** txt\_val = $(**this**).val();  **if** (txt\_val=="请输入密码"){  $(**this**).val("");  }  });  // 失去焦点  $("#username").blur(**function**(){  **var** txt\_val = $(**this**).val();  **if** (txt\_val==""){  $(**this**).val("请输入密码");  }  });  });  </script>  </head>    <body>  <div id=*"title"*></div>  <form id=*"form1"*>  帐号：<input type=*"text"* id=*"username"* name=*"username"* /></br>  密码：<input type=*"password"* id=*"password"* name=*"password"* /></br>  <input type=*"submit"* value=*"登录"* />  <input type=*"reset"* value=*"重置"* />  </form>  </body>  </html> |

## CSS操作

CSS-DOM 技术简单来说就是读取和设置style 对象的各种属性。style 属性很有用，但最大不足是无法通过它来提取到通过外部css 设置的样式信息，然而在jQuery 中，这些都是非常的简单。

可以直接利用css（）方法获取元素的样式属性。例如：

|  |
| --- |
| // 获取 <p>元素的颜色值  $("p").css("color"); |

无论color 属性是外部css 导入，还是直接拼接在HTML 元素里(内联）， css()方法都可以获取到属性style 里的其他属性的值。

也可以直接利用css（）方法设置某个元素的单个样式，例如：

|  |
| --- |
| // 设置 <p>元素的颜色值  $("p").css("color", "red"); |

与attr()方法一样， css()方法也可以同时设置多个样式属性，代码如下：

|  |
| --- |
| $("p").css({"color":"red", "fontSize": "30px"}); |

对于高度和宽度的操作，jQuery提供了height()和width()方法直接获取和设置。

|  |
| --- |
| // 获取 <p>元素的宽度  $("p").width();  // 设置<p>元素的高度  $("p").height("100px"); |

## 小结

本章主要介绍了使用jQuery对DOM元素进行各种操作。首先是获取DOM元素节点，以及对节点的修改、删除。然后对详细地介绍了对元素属性及样式的操作。

## 作业

超链接提示和图片提示效果。



# 事件和动画

## 章节目标

* 理解jQuery的事件机制
* 理解ready事件的触发时机
* 掌握事件的绑定与移除
* 掌握常用事件的使用
* 掌握三种常用的jQuery动画

## 事件

jQuery 事件处理方法是 jQuery 中的核心函数。

事件处理程序指的是当 HTML 中发生某些事件时所调用的方法。术语由事件“触发”（或“激发”）经常会被使用。

### ready事件

以浏览器装载文档为例，在页面加载完毕后，浏览器会通过JavaScript 为DOM 元素添加事件。在常规的JavaScript 代码中，通常使用window.onload 方法，而在jQuery 中，使用的是$( document).ready（）方法。$（document).ready。方法是事件模块中最重要的一个函数，可以极大地提高Web 应用程序的响应速度。jQuery 就是用$（document) .ready（）方法来代替传统JavaScript 的window.onload 方法的。通过使用该方法，可以在DOM 载入就绪时就对其进行操纵并调用执行它所绑定的函数。在使用过程中，需要注意$（document) .ready（）方法和window.onload 方法之间的细微区别。

**1.执行时间**

window.onload必须等到页面内包括图片的所有元素加载完毕后才能执行。

$(document).ready()是DOM结构绘制完毕后就执行，不必等到加载完毕。

**2.编写个数不同**

window.onload不能同时编写多个，如果有多个window.onload方法，只会执行一个

$(document).ready()可以同时编写多个，并且都可以得到执行

**3.简化写法**

window.onload没有简化写法

$(document).ready(function(){})可以简写成$(function(){});

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-01.html</title>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  alert("$(document).ready()-1");  });  $(document).ready(**function**(){  alert("$(document).ready()-2");  });      window.onload = **function**(){  alert("window.onload\_1");  };  window.onload = **function**(){  alert("window.onload\_2");  };  </script>  </head>    <body>  <h3>$(document).ready()与window.onload()区别</h3>    </body>  </html> |

注意，在jQuery中也提供了页面加载的方法，其写法如下：

|  |
| --- |
| $(window).load(**function**(){  // 处理代码  }); |

该方法与window.onload()方法是等价的。

### 事件绑定

在文档装载完成后，如果打算为元素绑定事件来完成某些操作，则可以使用bind（）方法来对匹配元素进行特定事件的绑定， bind（）方法的调用格式为：

bind(type[, data], fn);

第1 个参数是事件类型，类型包括： blur、focus 、load 、resize 、scroll 、unload、click、dblclick、mousedown 、mouseup、 mousemove 、mouseover 、mouseout 、mouseenter 、mouseleave 、change 、select、submit 、keydown 、keypress 、keyup 和error 等， 当然也可以是自定义名称。

第2 个参数为可选参数，作为event.data 属性值传递给事件对象的额外数据对象。

第3 个参数则是用来绑定的处理函数。

例如：在一个页面中要对一个按钮添加点击事件，并修改<p>标签字体颜色，代码如下：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-02.html</title>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  $("button").bind("click", **function**(){  $("p").css("color", "red");  });  });  </script>  </head>    <body>  <h3>事件绑定</h3>  <p>为按钮绑定事件</p>  <button>改变字体颜色</button>  </body>  </html> |

bind()方法可以为任何元素绑定事件，但对于一些常用的事件，jQuery还提供了更加简便的事件方法来完成相就的操作，下表列举了几个常用事件方法：

|  |  |
| --- | --- |
| **Event 函数** | **绑定函数至** |
| $(document).ready(function) | 将函数绑定到文档的就绪事件（当文档完成加载时） |
| $(selector).click(function) | 触发或将函数绑定到被选元素的点击事件 |
| $(selector).dblclick(function) | 触发或将函数绑定到被选元素的双击事件 |
| $(selector).focus(function) | 触发或将函数绑定到被选元素的获得焦点事件 |
| $(selector). blur(function) | 触发或将函数绑定到被选元素的失去焦点事件 |
| $(selector).mouseover(function) | 触发或将函数绑定到被选元素的鼠标悬停事件 |
| $(selector).mouseout(function) | 触发、或将函数绑定到指定元素的移开事件 |

以鼠标移入、移出事件为例：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-03.html</title>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  $("div").mouseover(**function**(){  $(**this**).css({"color": "red", "border":"1px solid red"});  }).mouseout(**function**(){  $(**this**).css({"color": "blue", "border":"1px solid blue"});  });  });  </script>  </head>    <body>  <h3>常用事件</h3>  <div>鼠标移入、移出事件</div>  </body>  </html> |

### 合成事件

jQuery 有两个合成事件一－hoverO方法和toggleO方法，类似前面讲过的ready（）方法， hover()方法和toggle（）方法都属于jQuery 自定义的方法。

**hover（）方法**

hover()方法用于模拟光标悬停事件。当光标移动到元素上时，会触发指定的第1 个函数(enter);当光标移出这个元素时，会触发指定的第2 个函数(leave) 。

将上面的例子改写成使用hover()方法：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-04.html</title>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  $("div").hover(**function**(){  $(**this**).css({"color": "red", "border":"1px solid red"});  },**function**(){  $(**this**).css({"color": "blue", "border":"1px solid blue"});  });  });  </script>  </head>    <body>  <h3>hover方法</h3>  <div>鼠标移入、移出</div>  </body>  </html> |

**toggle()方法**

toggle()方法可以切换元素的可见状态。如果元素是可见的， 切换后则为隐藏；如果元素是隐藏的，切换后则为可见的。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-05.html</title>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**() {  $("button").click(**function**() {  $("h3").toggle();  });  });  </script>  </head>    <body>  <h3>toggle方法</h3>  <button>显示\隐藏</button>  </body>  </html> |

此外在早期的jQuery版本中toggle()方法还可用于模拟鼠标的连接点击事件。第一次单击元素时，触发第一个函数；当再次单击时，触发第二个函数；如果还有更多函数，则依次触发，直到最后一个。随后再单击元素再重复调用所有的函数。但在jQuery1.9版本中，不再支持这种功能。

### 事件移除

unbind() 方法可以移除被选元素的事件处理程序。

该方法能够移除所有的或被选的事件处理程序，或者当事件发生时终止指定函数的运行。

unbind()的语法结构如下：

unbind([type], [fnName]);

第 1 个参数是事件类型， 第 2 个参数是将要移除的函数， 具体说明如下。

(1) 如果没有参数， 则删除所有绑定的事件。

(2) 如果提供了事件类型作为参数， 则只删除该类型的绑定事件。

(3) 如果把在绑定时传递的处理函数作为第 2 个参数， 则只有这个特定的事件处理函数会被删除。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-06.html</title>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**() {    $("#btn\_bind").click(**function**(){    $("#btn\_toggle").bind("click", fun1 = **function**(){  $("p").toggle();  }).bind("click", fun2 = **function**(){  $("h3").toggle();  })  ;    });    $("#btn\_unbind").click(**function**(){  $("#btn\_toggle").unbind();  });  $("#btn\_unbindp").click(**function**(){  $("#btn\_toggle").unbind("click", fun1);  });  });  </script>  </head>    <body>  <h3>移除事件</h3>  <p>显示文字</p>  <button id=*"btn\_toggle"*>显示\隐藏</button>  <button id=*"btn\_bind"*>绑定事件</button>  <button id=*"btn\_unbind"*>解除绑定</button>  <button id=*"btn\_unbindp"*>解除p标签操作</button>  </body>  </html> |

## 动画

### 显示和隐藏

show（）方法和hide（）方法是jQuery 中最基本的动画方法。

在HTML 文档里，为一个元素调用hide()方法，会将该元素的display 样式改为“none''。

当把元素隐藏后，可以使用show（）方法将元素的display 样式设置为先前的显示状态（“block ”或“ inline ”或其他除了“none ”之外的值）。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-07.html</title>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**() {  $("#btn\_show").click(**function**(){  $("p").show();  });    $("#btn\_hide").click(**function**(){  $("p").hide();  });  });  </script>  </head>    <body>  <h3>显示隐藏</h3>  <p>显示文字</p>  <button id=*"btn\_show"*>显示</button>  <button id=*"btn\_hide"*>隐藏</button>  </body>  </html> |

show（）方法和hide（）方法在不带任何参数的情况下，相当于css（”display'',"none/ block/inline），作用是立即隐藏或显示匹配的元素，不会有任何动画。如果希望在调用show（）方法时，元素慢慢地显示出来，可以为show（）方法指定一个速度参数，例如，指定一个速度关键字“ slow 、normal、 fast”。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-08.html</title>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**() {  $("#btn\_show").click(**function**(){  $("p").show("slow");  });    $("#btn\_hide").click(**function**(){  $("p").hide(1000);  });  });  </script>  </head>    <body>  <h3>带有动画的显示隐藏</h3>  <p>显示文字</p>  <button id=*"btn\_show"*>显示</button>  <button id=*"btn\_hide"*>隐藏</button>  </body>  </html> |

slow 、normal、 fast实际上相当于600、400、200的值，用来设定整个显示/隐藏过程消耗多少毫秒。

### 淡入淡出

与showO方法不相同的是， fadein（）方法和fadeout()方法只改变元素的不透明度。fadeOut()方法会在指定的一段时间内降低元素的不透明度，直到元素完全消失（“ display: none “）。fadein()方法则相反。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-09.html</title>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**() {  $("#btn\_fadein").click(**function**(){  $("p").fadeIn();  });    $("#btn\_fadeout").click(**function**(){  $("p").fadeOut(1000);  });  });  </script>  </head>    <body>  <h3>淡入淡出</h3>  <p>显示文字</p>  <button id=*"btn\_fadein"*>淡入</button>  <button id=*"btn\_fadeout"*>淡出</button>  </body>  </html> |

### 滑动

slideUp（）方法和slideDown（）方法只会改变元素的高度。如果一个元素的display 属性值为“ none 气当调用slideDown（）方法时，这个元素将由上至下延伸显示。slideUp()方法正好相反，元素将由下到上缩短隐藏。使用slideUp（）方法和slideDown（）方法再次对“内容”的显示和隐藏方式进行改变。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-10.html</title>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**() {  $("#btn\_slideUp").click(**function**(){  $("p").slideUp();  });    $("#btn\_slideDown").click(**function**(){  $("p").slideDown(1000);  });  });  </script>  </head>    <body>  <h3>滑动</h3>  <p style="background:*red*">  显</br>  示</br>  文</br>  字</p>  <button id=*"btn\_slideUp"*>缩短</button>  <button id=*"btn\_slideDown"*>拉伸</button>  </body>  </html> |

### 自定义动画

jQuery animate() 方法用于创建自定义动画。

语法：

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

必需的 params 参数定义形成动画的 CSS 属性。

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是动画完成后所执行的函数名称。

提示：默认地，所有 HTML 元素都有一个静态位置，且无法移动。

如需对位置进行操作，要记得首先把元素的 CSS position 属性设置为 relative、fixed 或 absolute！

animate()方法在设置动画时，又可以分为以下几种：

**简单动画**

animate()方法设置动画效果时，主要依赖对CSS的设置，例如要让一个div向右移动100px，则将其left属性设置为100px。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-10.html</title>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**() {  $("#btn\_left").click(**function**(){  $("#txt").animate({"left":"200px"},1000);  });  });  </script>  </head>    <body>  <h3>动画</h3>  <div style="width:*300px*;height:*300px*;background:*red*;">  <div id=*"txt"* style="position:*relative*">显示文字</div>  </div>  <button id=*"btn\_left"*>向右移动</button>  </body>  </html> |

**累加动画**

在之前的代码中，设置了｛left:”200px”｝作为动画参数。如果在200px 之前加上“＋＝＇＇或者“一＝”符号即表示在当前位置累加或者累减。

|  |
| --- |
| $("#txt").animate({"top":"+=100px"},1000); |

**叠加动画**

如果需要同时执行多个动画，例如在元素向右滑动的同时，再向下移动。根据animate（）方法的语法结构，可以写出如下的jQuery 代码：

|  |
| --- |
| $("#txt").animate({"left":"200px", "top":"200px"},1000); |

**多重动画**

上例中，两个动画效果是同时发生的，如果想要按顺序执行动画，只需把代码拆开，然后按照顺序写就可以了， jQuery 代码如下：

|  |
| --- |
| $("#txt").animate({"left":"200px"},1000); $("#txt").animate({"top":"200px"},1000); |

### 回调函数

如果要在动画执行结束后，进行一定的操作，就需要使用动画的回调函数。例如在动画结束后，弹框提示动画完成，可使用如下代码：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-12.html</title>    <meta name=*"keywords"* content=*"keyword1,keyword2,keyword3"*>  <meta name=*"description"* content=*"this is my page"*>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>    <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  $("button").click(**function**(){  $("#txt").animate({"left":"200px", "height":"100px"},  1000,  **function**(){  alert("动画结束");  });  });  });  </script>  </head>    <body>  <h3>动画回调</h3>  <div style="width:*300px*;height:*300px*;background:*red*;">  <div id=*"txt"* style="position:*relative*; background:*blue*">显示文字</div>  </div>  <button>开始动画</button>xian  </body>  </html> |

回调函数适用于所有的动画方法，例如show()和hide()方法也可以加入回调函数。

### 其它动画方法

stop() 方法用于停止动画或效果，在它们完成之前。stop() 方法适用于所有 jQuery 效果函数，包括滑动、淡入淡出和自定义动画。

语法

$(selector).stop(stopAll,goToEnd);

可选的 stopAll 参数规定是否应该清除动画队列。默认是 false，即仅停止活动的动画，允许任何排入队列的动画向后执行。

可选的 goToEnd 参数规定是否立即完成当前动画。默认是 false。

因此，默认地，stop() 会清除在被选元素上指定的当前动画。

下面的例子演示 stop() 方法，不带参数：

|  |
| --- |
| $("#stop").click(**function**(){  $("#panel").stop();  }); |

在动画执行的过程中，如果想对动画进行延迟操作，可以使用delay()方法。

|  |
| --- |
| $("#txt").animate({"left":"200px"},1000)  .delay(1000) // 延迟1000毫秒再执行后面的动画  .animate({"top":"200px"},1000); |

## 作业

滚动页面



# 表单表格操作

## 章节目标

* 掌握jQuery表单元素的操作
* 掌握jQuery对表格的特效操作

## 表单应用

### 单行文本框

文本框是表单域中最基本的元素，基于文本框的应用有很多。此处只简单介绍其中的一个应用一一获取和失去焦点改变样式。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-01.html</title>    <meta name=*"keywords"* content=*"keyword1,keyword2,keyword3"*>  <meta name=*"description"* content=*"this is my page"*>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>    <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  $("#username").val("请输入密码");  $("#username").css("color", "#999999");    // 焦点事件  $("#username").focus(**function**(){  **var** txt\_val = $(**this**).val();  **if** (txt\_val=="请输入密码"){  $(**this**).val("");  $(**this**).css("color", "#000000");  }  });  // 失去焦点  $("#username").blur(**function**(){  **var** txt\_val = $(**this**).val();  **if** (txt\_val==""){  $(**this**).val("请输入密码");  $(**this**).css("color", "#999999");  }  });  });  </script>  </head>    <body>  <h3>用户登录</h3>  <form id=*"form1"*>  帐号：<input type=*"text"* id=*"username"* name=*"username"* /></br>  密码：<input type=*"password"* id=*"password"* name=*"password"* /></br>  <input type=*"submit"* value=*"登录"* />  <input type=*"reset"* value=*"重置"* />  </form>  </body>  </html> |

本例是一个常见的用户登录页面，通过对用户名输入框的焦点变化，来显示一些操作提示信息。实现过程也非常简单，通过焦点事件的响应，更改CSS样式即可。

### 多行文本框

( 1 ）当单击“放大”按钮后，如果评论框的高度小于300px,

则在原有高度的基础上增加50px 。

( 2 ）当单击“缩小”按钮后，如果评论框的高度大于50px,

则在原有高度的基础上减去50pxo

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>jQuery-03.html</title>    <meta name=*"keywords"* content=*"keyword1,keyword2,keyword3"*>  <meta name=*"description"* content=*"this is my page"*>  <meta name=*"content-type"* content=*"text/html; charset=UTF-8"*>    <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  $("#bigger").click(**function**(){  **if** ($("textarea").height() < 300)  $("textarea").height($("textarea").height()+50);  });    $("#smaller").click(**function**(){  **if** ($("textarea").height() > 50)  $("textarea").height($("textarea").height()-50);  });  });  </script>  </head>    <body>  <h3>多行文本</h3>  <textarea rows=*"8"* cols=*"20"*></textarea></br>  <button id=*"bigger"*>放大</button>  <button id=*"smaller"*>缩小</button>  </body>  </html> |

### 复选框

对复选框最基本的应用，就是对复选框进行全选、反选和全不选等操作。复杂的操作需要与选项挂钩，来达到各种级联反应效果。

如果需要使复选框处于选中或者不选状态，必须通过控制元素的checked 属性来达到目的。如果属性checked 的值为true，说明被选中：如果值为false ，说明没被选中。因此可以基于这个属性来完成需求。

全选操作就是当用户单击“全选”按钮时，需要将复选框组全部选中。此时，需要为“全选”按钮绑定单击事件，然后使用选择符寻找符合要求的复选框，最后通过attr（）方法来设置属性checked 的值，使之选中。

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns=*"http://www.w3.org/1999/xhtml"*>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"* />  <title></title>  <!-- 引入jQuery库 -->  <script src=*"jquery/jquery-1.9.1.js"*></script>  <script type=*"text/javascript"*>  $(**function**(){  //全选  $("#CheckedAll").click(**function**(){  $('[name=items]:checkbox').attr('checked', **true**);  });  //全不选  $("#CheckedNo").click(**function**(){  $('[type=checkbox]:checkbox').attr('checked', **false**);  });  //反选  $("#CheckedRev").click(**function**(){  $('[name=items]:checkbox').each(**function**(){  //此处用JQ写法颇显啰嗦。体现不出JQ飘逸的感觉。  //$(this).attr("checked", !$(this).attr("checked"));  //$(this).prop("checked", !$(this).prop("checked"));    //直接使用JS原生代码，简单实用  **this**.checked=!**this**.checked;  });  });  //输出值  $("#send").click(**function**(){  **var** str="你选中的是：\r\n";  $('[name=items]:checkbox:checked').each(**function**(){  str+=$(**this**).val()+"\r\n";  })  alert(str);  });  })  </script>  </head>  <body>  <form method=*"post"* action=*""*>  你爱好的运动是？  <br/>  <input type=*"checkbox"* name=*"items"* value=*"足球"*/>足球  <input type=*"checkbox"* name=*"items"* value=*"篮球"*/>篮球  <input type=*"checkbox"* name=*"items"* value=*"羽毛球"*/>羽毛球  <input type=*"checkbox"* name=*"items"* value=*"乒乓球"*/>乒乓球  <br/>  <input type=*"button"* id=*"CheckedAll"* value=*"全　选"*/>  <input type=*"button"* id=*"CheckedNo"* value=*"全不选"*/>  <input type=*"button"* id=*"CheckedRev"* value=*"反　选"*/>  <input type=*"button"* id=*"send"* value=*"提　交"*/>  </form>  </body>  </html> |

### 下拉框

某网站的一个后台新增界面，在“负责频道”这个区域，用户可以通过中间的按钮将左边选中的选项添加到右边，也可以将右边的选项添加到左边，或者双击选项，将其添加给对方。

需要实现的功能如下。

(1 ）将选中的选项添加给对方。

(2 ）将全部选项添加给对方。

(3 ）双击某个选项将其添加给对方。

首先实现第1 个功能，即将下拉列表中被选中的选项添加给对方。

首先要获取下拉列表中被选中的选工页，然后将当前下拉列表中选中的选项删除，最后将删除的选项添加给对方。

假设先将左边的选项添加到右边。可以使用appendTo()方法完成。

|  |
| --- |
| $('#add').click(**function**() {  //获取选中的选项，删除并追加给对方  $('#select1 option:selected').appendTo('#select2');  }); |

然后实现第2个功能，即将全部选项添加给对方。

将全部选项添加给对方，和将选中的选项添加给对方之间的唯一不同就是获取的对象不同。

|  |
| --- |
| //全部移到右边  $('#add\_all').click(**function**() {  //获取全部的选项,删除并追加给对方  $('#select1 option').appendTo('#select2');  }); |

最后实现第3个功能，即双击某个选项将其添加给对方。

这里需要对下拉列表添加双击事件，然后通过$(“option:selected”, this)方法获取被选中的项。

|  |
| --- |
| //双击选项  $('#select1').dblclick(**function**(){ //绑定双击事件  //获取全部的选项,删除并追加给对方  $("option:selected",**this**).appendTo('#select2'); //追加给对方  }); |

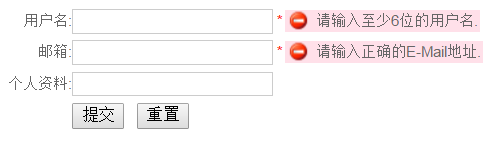
完整代码如下：

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns=*"http://www.w3.org/1999/xhtml"*>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"* />  <title></title>  <style type=*"text/css"*>  \* { margin:*0*; padding:*0*; }  **div***.centent* {  float:*left*;  text-align: *center*;  margin: *10px*;  }  **span** {  display:*block*;  margin:*2px 2px*;  padding:*4px 10px*;  background:*#898989*;  cursor:*pointer*;  font-size:*12px*;  color:*white*;  }  </style>  <!-- 引入jQuery -->  <script src=*"../../scripts/jquery.js"* type=*"text/javascript"*></script>  <script type=*"text/javascript"*>  $(**function**(){  //移到右边  $('#add').click(**function**() {  //获取选中的选项，删除并追加给对方  $('#select1 option:selected').appendTo('#select2');  });  //移到左边  $('#remove').click(**function**() {  $('#select2 option:selected').appendTo('#select1');  });  //全部移到右边  $('#add\_all').click(**function**() {  //获取全部的选项,删除并追加给对方  $('#select1 option').appendTo('#select2');  });  //全部移到左边  $('#remove\_all').click(**function**() {  $('#select2 option').appendTo('#select1');  });  //双击选项  $('#select1').dblclick(**function**(){ //绑定双击事件  //获取全部的选项,删除并追加给对方  $("option:selected",**this**).appendTo('#select2'); //追加给对方  });  //双击选项  $('#select2').dblclick(**function**(){  $("option:selected",**this**).appendTo('#select1');  });  });  </script>  </head>  <body>  <div class=*"centent"*>  <select multiple=*"multiple"* id=*"select1"* style="width:*100px*;height:*160px*;">  <option value=*"1"*>选项1</option>  <option value=*"2"*>选项2</option>  <option value=*"3"*>选项3</option>  <option value=*"4"*>选项4</option>  <option value=*"5"*>选项5</option>  <option value=*"6"*>选项6</option>  <option value=*"7"*>选项7</option>  </select>  <div>  <span id=*"add"* >选中添加到右边&gt;&gt;</span>  <span id=*"add\_all"* >全部添加到右边&gt;&gt;</span>  </div>  </div>  <div class=*"centent"*>  <select multiple=*"multiple"* id=*"select2"* style="width: *100px*;height:*160px*;">  <option value=*"8"*>选项8</option>  </select>  <div>  <span id=*"remove"*>&lt;&lt;选中删除到左边</span>  <span id=*"remove\_all"*>&lt;&lt;全部删除到左边</span>  </div>  </div>  </body>  </html> |

### 表单验证

表单作为HTML 最重要的一个组成部分，几乎在每个网页上都有体现，例如用户提交信息、用户反馈信息和用户查询信息等，因此它是网站管理者与浏览者之间沟通的桥梁。在表单中，表单验证的作用也是非常重要的，它能使表单更加灵活、美观和丰富。

以一个简单的用户注册为例。



原始代码如下：

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns=*"http://www.w3.org/1999/xhtml"*>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"* />  <title></title>  <link href=*"css/style.css"* rel=*"stylesheet"* type=*"text/css"* />  </head>  <body>  <form method=*"post"* action=*""*>  <div class=*"int"*>  <label for=*"username"*>用户名:</label>  <input type=*"text"* id=*"username"* class=*"required"* />  </div>  <div class=*"int"*>  <label for=*"email"*>邮箱:</label>  <input type=*"text"* id=*"email"* class=*"required"* />  </div>  <div class=*"int"*>  <label for=*"personinfo"*>个人资料:</label>  <input type=*"text"* id=*"personinfo"* />  </div>  <div class=*"sub"*>  <input type=*"submit"* value=*"提交"* id=*"send"*/><input type=*"reset"* id=*"res"*/>  </div>  </form>  </body>  </html> |

在表单内class 属性为“required”的文本框是必须填写的，因此需要将它与其他的非必须填写表单元素加以区别，即在文本框后面追加一个红色的小星星标识。可以使用append（）方法来完成。

当用户在“用户名”文本框中填写完信息后，将光标的焦点从“用户名”移出时，需要即时判断用户名是否符合验证规则。当光标的焦点从“邮箱”文本框移出时，需要即时判断“邮箱”填写是否正确，因此需要给表单元素添加失去焦点事件，即blur （）。

验证表单元素步骤如下。

1. 判断当前失去焦点的元素是“用户名”还是“邮箱飞然后分别处理。
2. 如果是“用户名气判断元素的值的长度是否小于6 ，如果小于6 ，则用红色提醒用户输入不正确，反之，则用绿色提醒用户输入正确。
3. 如果是“邮箱飞判断元素的值是否符合邮箱的格式，如果不符合，则用红色提醒用户输入不正确，反之，则用绿色提醒用户输入正确。
4. 将提醒信息追加到当前元素的父元素的最后。

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns=*"http://www.w3.org/1999/xhtml"*>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"* />  <title></title>  <link href=*"css/style.css"* rel=*"stylesheet"* type=*"text/css"* />  <!-- 引入jQuery -->  <script src=*"../jquery/jquery-1.9.1.js"* type=*"text/javascript"*></script>  <script type=*"text/javascript"*>  //<![CDATA[  $(**function**(){  //如果是必填的，则加红星标识.  $("form :input.required").each(**function**(){  **var** $required = $("<strong class='high'> \*</strong>"); //创建元素  $(**this**).parent().append($required); //然后将它追加到文档中  });  //文本框失去焦点后  $('form :input').blur(**function**(){  **var** $parent = $(**this**).parent();  $parent.find(".formtips").remove();  //验证用户名  **if**( $(**this**).is('#username') ){  **if**( **this**.value=="" || **this**.value.length < 6 ){  **var** errorMsg = '请输入至少6位的用户名.';  $parent.append('<span class="formtips onError">'+errorMsg+'</span>');  }**else**{  **var** okMsg = '输入正确.';  $parent.append('<span class="formtips onSuccess">'+okMsg+'</span>');  }  }  //验证邮件  **if**( $(**this**).is('#email') ){  **if**( **this**.value=="" || ( **this**.value!="" && !/.+@.+\.[a-zA-Z]{2,4}$/.test(**this**.value) ) ){  **var** errorMsg = '请输入正确的E-Mail地址.';  $parent.append('<span class="formtips onError">'+errorMsg+'</span>');  }**else**{  **var** okMsg = '输入正确.';  $parent.append('<span class="formtips onSuccess">'+okMsg+'</span>');  }  }  }).keyup(**function**(){  $(**this**).triggerHandler("blur");  }).focus(**function**(){  $(**this**).triggerHandler("blur");  });//end blur    //提交，最终验证。  $('#send').click(**function**(){  $("form :input.required").trigger('blur');  **var** numError = $('form .onError').length;  **if**(numError){  **return false**;  }  alert("注册成功,密码已发到你的邮箱,请查收.");  });  //重置  $('#res').click(**function**(){  $(".formtips").remove();  });  })  //]]>  </script>  </head>  <body>  <form method=*"post"* action=*""*>  <div class=*"int"*>  <label for=*"username"*>用户名:</label>  <input type=*"text"* id=*"username"* class=*"required"* />  </div>  <div class=*"int"*>  <label for=*"email"*>邮箱:</label>  <input type=*"text"* id=*"email"* class=*"required"* />  </div>  <div class=*"int"*>  <label for=*"personinfo"*>个人资料:</label>  <input type=*"text"* id=*"personinfo"* />  </div>  <div class=*"sub"*>  <input type=*"submit"* value=*"提交"* id=*"send"*/><input type=*"reset"* id=*"res"*/>  </div>  </form>  </body>  </html> |

## 表格应用

### 表格变色

一张人员资料表，其HTML代码如下：

|  |
| --- |
| <table>  <thead>  <tr><th>姓名</th><th>性别</th><th>暂住地</th></tr>  </thead>  <tbody>  <tr><td>张山</td><td>男</td><td>浙江宁波</td></tr>  <tr><td>李四</td><td>女</td><td>浙江杭州</td></tr>  <tr><td>王五</td><td>男</td><td>湖南长沙</td></tr>  <tr><td>找六</td><td>男</td><td>浙江温州</td></tr>  <tr><td>Rain</td><td>男</td><td>浙江杭州</td></tr>  <tr><td>MAXMAN</td><td>女</td><td>浙江杭州</td></tr>  </tbody>  </table> |

这是一张非常普通的表格，现在需要给表格进行隔行变色操作。

首先定义两个样式。然后选择表格奇数行和偶数行分别添加样式，可以使用选择器来完成。

|  |
| --- |
| *.even* { background:*#FFF38F*;} /\* 偶数行样式\*/  *.odd* { background:*#FFFFEE*;} /\* 奇数行样式\*/ |

|  |
| --- |
| $(**function**(){  $("tr:odd").addClass("odd"); /\* 奇数行添加样式\*/  $("tr:even").addClass("even"); /\* 偶数行添加样式\*/  }) |

如果还需要将某一行变为高亮显示状态，那么可以使用contains 选择器来实现。

|  |
| --- |
| $("tr:contains('王五')").addClass("selected"); |

### 折叠效果

在上例的人员表格的基础上， 增加人员分类。

|  |
| --- |
| <table>  <thead>  <tr><th>姓名</th><th>性别</th><th>暂住地</th></tr>  </thead>  <tbody>  <tr class=*"parent"* id=*"row\_01"*><td colspan=*"3"*>前台设计组</td></tr>  <tr class=*"child\_row\_01"*><td>张山</td><td>男</td><td>浙江宁波</td></tr>  <tr class=*"child\_row\_01"*><td>李四</td><td>女</td><td>浙江杭州</td></tr>  <tr class=*"parent"* id=*"row\_02"*><td colspan=*"3"*>前台开发组</td></tr>  <tr class=*"child\_row\_02"*><td>王五</td><td>男</td><td>湖南长沙</td></tr>  <tr class=*"child\_row\_02"*><td>找六</td><td>男</td><td>浙江温州</td></tr>  <tr class=*"parent"* id=*"row\_03"*><td colspan=*"3"*>后台开发组</td></tr>  <tr class=*"child\_row\_03"*><td>Rain</td><td>男</td><td>浙江杭州</td></tr>  <tr class=*"child\_row\_03"*><td>MAXMAN</td><td>女</td><td>浙江杭州</td></tr>  </tbody>  </table> |

现在需要实现的是当单击分类行时， 可以关闭相应的内容。例如单击“前台设计组” 行， 则它对应的“张山和李四” 两行将收缩。

在这个表格中， 给每个＜tr＞元素设置属性是非常重要的， 读者可以在HTML 代码中看出一些规则， 即给分类行设置了class＝”parent”属性， 同时也分别给它们设置了id 值， 而在它们下面的行，只设置了class 属性， 并且这个class 的值是在id 值的基础上通过加上“ child ,, 来设置的。

|  |
| --- |
| $(**function**(){  $('tr.parent').click(**function**(){ // 获取所谓的父行  $(**this**)  .toggleClass("selected") // 添加/删除高亮  .siblings('.child\_'+**this**.id).toggle(); // 隐藏/显示所谓的子行  });  }) |

### 内容筛选

利用该选择器再结合jQuery 的filter（）筛选方法， 可以实现表格内容的过滤。

首先在表格上方添加一个文本框，用于根据用户输入的内容来筛选表格内容，然后为文本框绑定keyup 事件，最后将.filter(": contains（＇李）“）代码中的“李”用变量值代替。

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  <html xmlns=*"http://www.w3.org/1999/xhtml"* xml:lang=*"en"* lang=*"en"*>  <head>  <title></title>  <link href=*"css/style.css"* rel=*"stylesheet"* type=*"text/css"* />  <!-- 引入jQuery -->  <script src=*"../jquery/jquery-1.9.1.js"* type=*"text/javascript"*></script>  <script type=*"text/javascript"*>  $(**function**(){  $("#filterName").keyup(**function**(){  $("table tbody tr")  .hide()  .filter(":contains('"+( $(**this**).val()  )+"')")  .show();  }).keyup();  })  </script>  </head>  <body>  <div>  <br/>  筛选：  <input id=*"filterName"* />  <br/>  </div>  <table>  <thead>  <tr><th>姓名</th><th>性别</th><th>暂住地</th></tr>  </thead>  <tbody>  <tr><td>张山</td><td>男</td><td>浙江宁波</td></tr>  <tr><td>李四</td><td>女</td><td>浙江杭州</td></tr>  <tr><td>王五</td><td>男</td><td>湖南长沙</td></tr>  <tr><td>找六</td><td>男</td><td>浙江温州</td></tr>  <tr><td>Rain</td><td>男</td><td>浙江杭州</td></tr>  <tr><td>MAXMAN</td><td>女</td><td>浙江杭州</td></tr>  <tr><td>王六</td><td>男</td><td>浙江杭州</td></tr>  <tr><td>李字</td><td>女</td><td>浙江杭州</td></tr>  <tr><td>李四</td><td>男</td><td>湖南长沙</td></tr>  </tbody>  </table>  </body>  </html> |

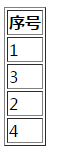
### 表头排序

利用参数和宏实现报表中点击表头排序功能，该方式实现的表头排序，需要进行页面刷新，实际上要重新进行报表运算。利用jQuery可以方便地实现报表的客户端排序，能减轻对服务器造成的负担，也能给用户带来更好的体验。

我们先从一个最简单的单列表格来看：

|  |
| --- |
| <table border=*"1"*>  <thead>  <tr>  <th>序号</th>  </tr>  </thead>  <tbody>  <tr>  <td>1</td>  </tr>  <tr>  <td>3</td>  </tr>  <tr>  <td>2</td>  </tr>  <tr>  <td>4</td>  </tr>  </tbody>  </table> |

该表格只有一个列，包含表头和四个行。在网页中显示为：



对于该表格，我们需要，当首次点击表头时，每行的数据可以按照升序排序，再次点击表头时，按当前顺序的反序进行排列。jQuery实现如下：

|  |
| --- |
| $(**function**(){  // 表格元素  **var** $table = $("table");  **var** $tHead = $table.children("thead");  **var** $tTh = $tHead.find("tr th");  **var** $tBody = $table.children("tbody");  **var** $tBodyTr = $tBody.find("tr");    // 排序列  **var** sortIndex = -1;    // 表头点击事件  $tTh.click(**function**(){  // 创建数组，存放数据  **var** trsValue = **new** Array();    // 遍历每一行<tr>  $tBodyTr.each(**function**(){  trsValue.push($(**this**).html());    $(**this**).html("");  });    // 没有排好顺序的使用排序算法排序  **if** (sortIndex != 0){  // 调用排序算法  sort(trsValue);    sortIndex = 0;  }  **else** {  // 数组反序 已经排列好顺序可以直接反序  trsValue.reverse();  }    // 重新填充  **for**(**var** i=0; i< trsValue.length; i++){  $("tbody tr:eq(" + i + ")").html(trsValue[i]);  }  });    });    // 排序算法  // 参数：数组  **function** sort(trsValue){  // 冒泡排序 升序  **for** (**var** i=0; i<trsValue.length-1; i++){  **for** (**var** j=0; j<trsValue.length-1-i; j++){  **if** (trsValue[j] > trsValue[j+1]){  **var** temp = trsValue[j];  trsValue[j] = trsValue[j+1];  trsValue[j+1] = temp;  }  }  }  } |
|  |

再看如下表格：

|  |
| --- |
| <table id=*"tableSort"*>  <thead>  <tr>  <th type=*"number"*>序号</th>  <th type=*"string"*>书名</th>  <th type=*"number"*>价格（元）</th>  <th type=*"string"*>出版时间</th>  <th type=*"number"*>印刷量（册）</th>  <th type=*"ip"*>IP</th>  </tr>  </thead>  <tbody>  <tr class=*"hover"*>  <td class=*"sequence"*>1</td>  <td>狼图腾</td>  <td>29.80</td>  <td>2009-10</td>  <td>50000</td>  <td>192.168.1.125</td>  </tr>  <tr>  <td class=*"sequence"*>2</td>  <td>孝心不能等待</td>  <td>29.80</td>  <td>2009-09</td>  <td>800</td>  <td>192.68.1.125</td>  </tr>  <tr>  <td class=*"sequence"*>3</td>  <td>藏地密码2</td>  <td>28.00</td>  <td>2008-10</td>  <td></td>  <td>192.102.0.12</td>  </tr>  <tr>  <td class=*"sequence"*>4</td>  <td>藏地密码1</td>  <td>24.80</td>  <td>2008-10</td>  <td></td>  <td>215.34.126.10</td>  </tr>  <tr>  <td class=*"sequence"*>5</td>  <td>设计模式之禅</td>  <td>69.00</td>  <td>2011-12</td>  <td></td>  <td>192.168.1.5</td>  </tr>  <tr>  <td class=*"sequence"*>6</td>  <td>轻量级 Java EE 企业应用实战</td>  <td>99.00</td>  <td>2012-04</td>  <td>5000</td>  <td>192.358.1.125</td>  </tr>  <tr>  <td class=*"sequence"*>7</td>  <td>Java 开发实战经典</td>  <td>79.80</td>  <td>2012-01</td>  <td>2000</td>  <td>192.168.1.25</td>  </tr>  <tr>  <td class=*"sequence"* onclick="sortArray()">8</td>  <td>Java Web 开发实战经典</td>  <td>69.80</td>  <td>2011-11</td>  <td>2500</td>  <td>215.168.54.125</td>  </tr>  </tbody>  </table> |

在这个表格中，包含了多个列，我们需要实现当点击每一个列头时，都需要以当前列的内容对表格进行排序。这时由于不同列的数据不再是简单的数字，而且包含了更为复杂的类型，比如IP地址。因此，需要根据不同的类型实现不同的排序算法。

jQuery实现如下：

|  |
| --- |
| $(document).ready(  **function**() {  **var** tableObject = $('#tableSort');// 获取id为tableSort的table对象  **var** tbHead = tableObject.children('thead');// 获取table对象下的thead  **var** tbHeadTh = tbHead.find('tr th');// 获取thead下的tr下的th  **var** tbBody = tableObject.children('tbody');// 获取table对象下的tbody  **var** tbBodyTr = tbBody.find('tr');// 获取tbody下的tr  **var** sortIndex = -1;  tbHeadTh.each(**function**() {// 遍历thead的tr下的th  **var** thisIndex = tbHeadTh.index($(**this**));// 获取th所在的列号  $(**this**).mouseover(**function**() {  tbBodyTr.each(**function**() {// 编列tbody下的tr  **var** tds = $(**this**).find("td");// 获取列号为参数index的td对象集合  $(tds[thisIndex]).addClass("hover");  });  }).mouseout(**function**() {  tbBodyTr.each(**function**() {  **var** tds = $(**this**).find("td");  $(tds[thisIndex]).removeClass("hover");  });  });  $(**this**).click(**function**() {  **var** dataType = $(**this**).attr("type");  checkColumnValue(thisIndex, dataType);  });  });  $("tbody tr").removeClass();// 先移除tbody下tr的所有css类  $("tbody tr").mouseover(**function**() {  $(**this**).addClass("hover");  }).mouseout(**function**() {  $(**this**).removeClass("hover");  });  // 对表格排序  **function** checkColumnValue(index, type) {  **var** trsValue = **new** Array();  tbBodyTr.each(**function**() {  **var** tds = $(**this**).find('td');  trsValue.push(type + ".separator" + $(tds[index]).html()  + ".separator" + $(**this**).html());  $(**this**).html("");  });  **var** len = trsValue.length;  **if** (index == sortIndex) {  trsValue.reverse(); // 颠倒数组元素顺序  } **else** {  // 冒泡排序  **for** (**var** i = 0; i < len; i++) {  type = trsValue[i].split(".separator")[0];  **for** (**var** j = i + 1; j < len; j++) {  value1 = trsValue[i].split(".separator")[1];  value2 = trsValue[j].split(".separator")[1];  **if** (type == "number") {  value1 = value1 == "" ? 0 : value1;  value2 = value2 == "" ? 0 : value2;  **if** (parseFloat(value1) > parseFloat(value2)) {  **var** temp = trsValue[j];  trsValue[j] = trsValue[i];  trsValue[i] = temp;  }  } **else** **if** (type == "ip") {  **if** (ip2int(value1) > ip2int(value2)) {  **var** temp = trsValue[j];  trsValue[j] = trsValue[i];  trsValue[i] = temp;  }  } **else** {  **if** (value1.localeCompare(value2) > 0) {// 该方法不兼容谷歌浏览器  **var** temp = trsValue[j];  trsValue[j] = trsValue[i];  trsValue[i] = temp;  }  }  }  }  }  **for** (**var** i = 0; i < len; i++) {  $("tbody tr:eq(" + i + ")").html(  trsValue[i].split(".separator")[2]);  }  sortIndex = index;  }  //IP转成整型  **function** ip2int(ip) {  **var** num = 0;  ip = ip.split(".");  num = Number(ip[0]) \* 256 \* 256 \* 256 + Number(ip[1]) \* 256  \* 256 + Number(ip[2]) \* 256 + Number(ip[3]);  **return** num;  }  }) |

# jquery插件的使用和写法

## 概述

插件也称为扩展，是一种遵循一定规范的应用程序接口编写出来的程序。

jQuery的易扩展性，吸引了全球很多开发者共同来编写jQuery插件。目前，已经有超过几千种的插件应用在全球的不同类型的项目上。使用这些经过无数人检验和完善的优秀插件，可以帮助用户开发出稳定的应用系统，节约项目成本。

最新最全的插件可以从jQuery官方网站的插件版块中获取，网站地址<http://plugins.jquery.com/>.

本章内容主要介绍三种插件表单验证插件、表单插件及UI插件。

## jQuery表单验证插件——Validation

### Validation 简介

最常使用JavaScript 的场合就是表单的验证，而jQuery 作为一个优秀的JavaScript 库，也提供

了一个优秀的表单验证插件—Validation。Validation是历史最悠久的jQuery插件之一，经过了全球范围内不同项目的验证，并得到了许多Web开发者的好评，作为一个标准的验证方法库，Validation 拥有如下特点：

* 内置验证规则：拥有必填、数字、E-Mail、URL和信用卡号等19类内置验证规则。
* 自定义验证规则：可以很方便地自定义验证规则。
* 简单强大的验证信息提示：默认了验证信息提示，并提供自定义覆盖默认提示信息的功能。
* 实时验证：可以通过keyup 或blur 事件触发验证，而不仅仅在表羊提交的时候验证。

下载地址：<http://plugins.jquery.com/validation/>

### 快速入门

以一个简单的评论表单提交的例子引入Validation插件的用法：

**comment.jsp**代码如下：

|  |
| --- |
| <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+  ":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <base href=*"*<%=basePath%>*"*>  <script type=*"text/javascript"* src=*"js/jquery-1.9.1.js"*>  </script>  <script type=*"text/javascript"* src=*"js/jquery.validate.js"*>  </script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  $("#commentForm").validate();  });  </script>  </head>    <body>  <form action=*""* id=*"commentForm"* method=*"get"*>  <p>  <label for=*"username"*>姓名</label><em>\*</em>  <input id=*"username"* name=*"username"* size=*"25"*  class=*"required"* minlength=*"2"*>  </p>  <p>  <label for=*"email"*>电子邮件</label><em>\*</em>  <input id=*"email"* name=*"email"* size=*"25"*  class=*"required email"* >  </p>  <p>  <label for=*"url"*>网址</label>  <input id=*"url"* name=*"url"* size=*"25"*  class=*"url"* >  </p>  <p>  <label for=*"comment"*>评论</label>  <textarea cols=*"22"* class=*"required"*>  </textarea>  </p>  <p>  <input type=*"submit"* class=*"submit"* value=*"提交"*>  </p>  </form>  </body>  </html> |

上面代码完成了以下验证。

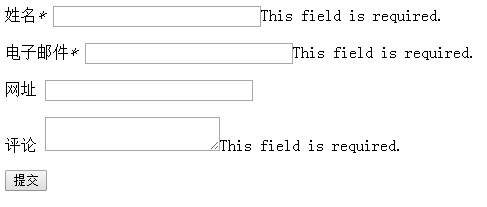
( 1 ）对“姓名”的必填和长度至少是两位的验证。

( 2 ）对“电子邮件”的必填和是否为E-mail 格式的验证。

( 3 ）对“网址”是否为url 的验证。

( 4 ）对“你的评论”的必填验证。

当用户单击“提交”按钮后，显示如图所示的效果。



当用户在“姓名”对应的文本框中输入字符时，表单元素也会实时响应验证，产生图所示的效果。



当用户输入字符时，表单元素就会实时响应验证信息，而不是只在用户单击“提交”按钮后才

出现。这样做的好处是极大地方便了用户，促使用户填写出符合格式的数据。

从例子中可以看到，只需完成以下几步操作，就可以将一个普通的表单改造为可以进行

Validation 验证的表单。

( 1 ）引入jQuery 库和Validation 插件。

|  |
| --- |
| <script type=*"text/javascript"* src=*"js/jquery-1.9.1.js"*>  </script>  <script type=*"text/javascript"* src=*"js/jquery.validate.js"*>  </script> |

（2）确定哪个表单需要被验证

|  |
| --- |
| $("#commentForm").validate(); |

( 3 ）针对不同的字段，进行验证规则编码，设置字段相应的属性。

class＝”required”为必须填写， minlength＝”2”为最小长度为2 。

class=”required email”为必须填写和内容需要符合E-mail 格式。

class＝”url”为url 格式验证。

### 增加中文验证的写法

在上节的例子中，我们发现，验证插件默认是英文的验证方式，validate插件提供了多种语言的支持文件，只需在页面**comment\_zh.jsp**当中加入中文支持的js文件即可：

|  |
| --- |
| <script type=*"text/javascript"* src=*"js/messages\_zh.js"*>  </script> |

显示效果如下：



当然可以通过修改中文支持的js文件来改变显示的提示内容。

### 去除class验证方式的写法

在上节的例子中，开发者必须把required、url 和email 写到class 属性里，才能完成必填验证、

url 验证和E-mail 验证：把minlength 属性的值设置为2，才能完成最小长度为2 的验证。虽然class

和minlength 属性都符合W3C 规范，但对于开发者来说，将与验证相关的信息写在class 属性里，同时class属性又往往用于css样式的选择器，故而会显得表单元素过于臃肿，所以validate插件提供了去除class验证方式的写法，修改**comment\_outclass.jsp**页面js代码如下：

|  |
| --- |
| $(document).ready(**function**(){  $("#commentForm").validate({  rules: {  username: {  required:**true**,  minlength:2  },  email: {  required: **true**,  email: **true**  },  url: "url",  comment:"required"  },  messages: {  username:{  required:"请输入用户名",  minlength:"用户名长度不小于5个字符"  },  email: {  required: "请输入Email地址",  email: "请输入正确的email地址"  },  url:"请输入路径",  comment:"请输入评论"  }  });  }); |

同时，删除掉网页元素当中class属性，代码如下：

|  |
| --- |
| <form action=*""* id=*"commentForm"* method=*"get"*>  <p>  <label for=*"username"*>姓名</label><em>\*</em>  <input id=*"username"* name=*"username"* size=*"25"*  >  </p>  <p>  <label for=*"email"*>电子邮件</label><em>\*</em>  <input id=*"email"* name=*"email"* size=*"25"*  >  </p>  <p>  <label for=*"url"*>网址</label>  <input id=*"url"* name=*"url"* size=*"25"*  >  </p>  <p>  <label for=*"comment"*>评论</label>  <textarea cols=*"22"* name=*"comment"* id=*"comment"*>  </textarea>  </p>  <p>  <input type=*"submit"* class=*"submit"* value=*"提交"*>  </p>  </form> |

显示效果如下图：



最终操作步骤如下：

1. 在$(“#commentForm”).validate()方法中增加rules属性
2. 通过每个字段的name属性来匹配验证规则
3. 定义验证规则，例如required:true,email:true,minlength:2等

## jQuery表单插件-Form

### Form插件简介

jQuery Form 插件是一个优秀的Ajax 表单插件，可以非常容易地、无侵入地升级HTML 表单

以支持Ajaxo jQuery Form 有两个核心方法句ax.Form（）和ajaxSubmitO，它们集合了从控制表单

元素到决定如何管理提交进程的功能。另外，插件还包括其他的一些方法： formTo Array（）、

formSerializeO 、fieldSerializeO 、fieldValueO 、clearForm（）、clearFields（）和resetForm()等。

Mike Alsup 设计了jQueryForm插件，并进行改善和维护。

下载地址：

<http://malsup.com/jquery/form/>

### 快速上手

页面**form.jsp**当中添加form表单，然后引入jQuery库和Form插件，并编写Ajax提交jQuery代码如下：

|  |
| --- |
| <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+  ":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <base href=*"*<%=basePath%>*"*>  <script type=*"text/javascript"* src=*"js/jquery-1.9.1.js"*>  </script>  <script type=*"text/javascript"* src=*"js/jquery.form.js"*>  </script>  <script type=*"text/javascript"*>  $(**function**(){  $("#myForm").ajaxForm(**function**(){  alert("提交成功！");  });  });  </script>  </head>    <body>  <form action=*"index.jsp"* method=*"post"* id=*"myForm"*>  <p>  <label for=*"username"*>姓名</label><em>\*</em>  <input id=*"username"* name=*"username"* size=*"25"*  minlength=*"2"*>  </p>  <p>  <label for=*"email"*>电子邮件</label><em>\*</em>  <input id=*"email"* name=*"email"* size=*"25"*  >  </p>  <p>  <label for=*"comment"*>评论</label>  <textarea cols=*"22"*>  </textarea>  </p>  <p>  <input type=*"submit"* value=*"提交"*>  </p>  </form>  </body>  </html> |

当表单被提交时，“姓名”、“邮箱”和“评论”字段的值会以无刷新的方式提交到文件index.jsp中。如果服务器返回一个成功状态，那么用户将会看到“提交成功！”的提示。

### 核心方法一ajaxForm()和ajaxSubmit()

正如上例的代码所示，通过核心方法ajaxForm（），能很容易地将表单升级为Ajax 提交方式。

|  |
| --- |
| $("#myForm").ajaxForm(**function**(){  alert("提交成功！");  }); |

Form 插件还有一个核心方法ajaxSubmit（），也能完成同样的功能,代码如下：

|  |
| --- |
| $("#myForm").submit(**function**(){  $(**this**).ajaxSubmit(**function**(){  alert("提交成功！");  });  **return false**;//阻止表单默认提交  }); |

通过调用ajaxSubmit()方法来响应用户的提交表单操作，从而使表单的提交方式由传统的提交方式转变为Ajax 提交方式。

通过Form 插件的这两个核心方法，都可以在不修改表单的HTML 代码结构的情况下，轻易地

将表单的提交方式升级为Ajax 提交方式。

### ajaxForm()方法和ajaxSubmit()方法的参数

ajaxForm（）方法和ajaxSubmit（）方法都能接受0 个或者1 个参数。当为单个的参数时，该参数既可以是一个回调函数，也可以是一个options 对象。上面例子的参数就是回调函数。接下来介绍options 对象，通过给ajaxForm（）方法和ajaxSubmitO方法传递options 对象，使得它们对表单拥有更多的控制权。

首先定义一个对象options ，然后在对象里设置参数，代码如下：

|  |
| --- |
| **var** options={  target:'#output',//把服务器返回的内容放入id为output的元素中  beforeSubmit: showRequest,//提交前的回调函数  success:showResponse,//提交后的回调函数  url:’index.jsp’,//默认是form的action，如果声明，则会覆盖  type:’post’,//默认是form的method，如果声明，则会覆盖  dataType:**null**,//'xml'or'json'(接受服务器返回的类型)  clearForm:**true**,//成功提交后删除所有表单元素的值  resetForm:**true**,//成功提交后重置所有表单元素的值  timeout:3000//限制请求的时间，超过3秒，跳出请求  } |

定义options 对象之后，就可以把这个options 对象传递给ajaxFormO方法， jQuery 代码如下：

|  |
| --- |
| $("#myForm").ajaxForm(options); |

或者提交给ajaxSubmit（）方法

|  |
| --- |
| $("#myForm").submit(**function**(){  $(**this**).ajaxSubmit(options);  **return false**;//阻止表单默认提交  }); |

在options 对象里，指定了两个回调函数，即beforeSubmit: showRequest 和success: showResponse，它们分别会在表单提交前和表单提交后被调用。

下面来看看这两个回调函数具体有哪些参数。

* beforeSubmit 一一提交前的回调函数

提交前的回调函数的代码如下：

|  |
| --- |
| **function** showRequest(formData,jqForm,options){  **var** queryString = $.param(formData);  alert(queryString);  **return true**;  } |

这个回调函数有3 个参数。

第1 个参数formData 是数组对象。在这里，使用$.param（）方法把它转化为字符串，得到如下这种格式：

username=1&&email=2

需要注意的是，当表单提交时， Form 插件会以Ajax 方式自动提交这些数据。

第2 个参数jqForm 是一个jQuery 对象，它封装了表单的元素。

第3 个参数options 就是options 对象。前面已经声明了options 对象里的一些属性，其他没有声明的，则会使用默认的属性。

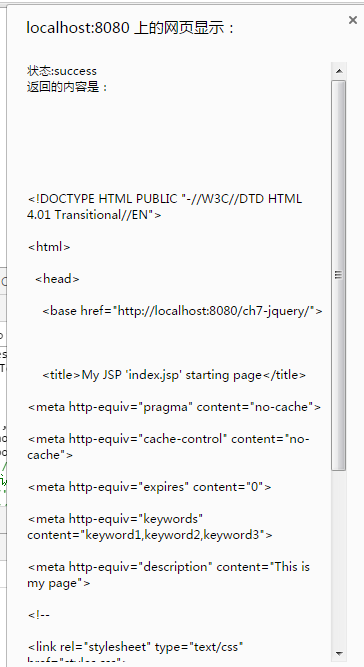
在这个回调函数中，只要不返回false，表单都将被允许提交：如果返回false，则会阻止表单提交。可以利用这个特性，在表单提交之前验证数据（如果不符合验证规则，则阻止表单提交。

* success 一一提交后的回调函数

提交后的回调函数的代码如下：

|  |
| --- |
| **function** showResponse(responseText,statusText,xhr,$form){  alert("状态:"+statusText+"\n"+"返回的内容是：\n"+responseText);  } |

弹出效果如下：



success 有4 个参数responseText, status Text，对览和$form 。其中responseText 和statusText2 个

比较常用。

status Text 只是一个返回状态，例如success 、error 等。

response Text 携带着服务器返回的数据内容。responseText 会根据设置的options 对象中的

dataType 属性来返回相应格式的内容。具体情况如下。

(1 ）对于缺省的HTML 返回，回调函数的第1个参数是XMLHttpRequest 对象的responseText

属性。

(2 ）当dataType 属性被设置为xml 时，回调函数的第1 个参数是XMLHttpRequest对象的

responseXML属性。

首先先创建一个服务器端的servlet，便于请求访问处理。代码如下：

|  |
| --- |
| **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  // 这里响应类型必须是text/xml  response.setContentType("text/xml;charset=gbk");  request.setCharacterEncoding("gbk");  PrintWriter out = response.getWriter();  // XML头，没有这个可能Javascript解析不出XML，最好设一下  StringBuilder sb = **new** StringBuilder("<?xml version=\"1.0\" encoding=\"gbk\"?>");  // 根元素<root></root>一定需要，否则Javascript解析不出XML，但名字不一定要叫root  sb.append("<root><info id='1'><b>a</b></info ></root>");  //写出XML  out.println(sb.toString());  out.flush();  out.close();  } |

例如声明服务器返回数据的类型为xml，然后以XML 方式解析数据，代码如下：

|  |
| --- |
| $("#myForm").ajaxForm({  dataType:'xml',  success:processXml  });  **function** processXml(responseXML){  **var** a = $('a',responseXML).text();  $("#xmlOut").html(a);  } |

声明服务器返回数据的类型为json时，回调函数的第一个参数是从服务器返回的json数据对象，

Servlet则需要引入json相关的包来创建json对象，具体包名如下：



servlet代码如下：

|  |
| --- |
| **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  PrintWriter out = response.getWriter();  response.setCharacterEncoding("utf-8");  request.setCharacterEncoding("utf-8");  response.setContentType("text/html;charset=UTF-8");  String username = request.getParameter("username");  JSONObject json = **new** JSONObject();  json.put("name", username);  out.print(json.toString());  out.flush();  out.close();  } |

|  |
| --- |
| $("#myForm").ajaxForm({  dataType:'json',  success:processJson  });  **function** processJson (data){  $("#jsonOut").html(data.name);  } |

## jQuery UI插件

### jquery UI 简介

jQuery UI源自于一个jQuery 插件一－Interface 。 Interface 插件是由Stefan Petre 创建的， Paul Bakaus 也为该插件的开发做出了重大贡献。后来随着版本的迭代，由于改动越来越大，改名为jquery UI，同时也确立了jquery UI 官方插件的地位，并逐步走向完善。

jQuery UI 主要分为3 个部分，交互、微件和效果库

### 快速上手

下载地址：<http://jqueryui.com/>

jQuery UI 插件的大部分API 己经统一了。以Draggables 为例，有4 个基本的API 。

* draggable ( options ）：这是用来让一个DOM 对象变成可拖动的对象的方法。其中的options

可以设置各种不同的参数。

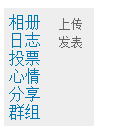
* draggable （“disable”）：让对应的DOM 对象暂时禁用拖动。
* draggable （“enable”）：让对应的DOM 对象重新启用拖动。
* draggable （“destroy”）：彻底移除拖动功能。

以上这些就是jQuecy UI 中大部分插件通用的用法。jQuecy UI 插件系列众多，鉴于文章篇幅有限，

这里只挑选出其中一个实用的拖动排序组件Sortable 来进行简单介绍和讲解，其他组件由学员自行参考jQuery ui 的官方网站。

案例代码如下

|  |
| --- |
| <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+  ":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <base href=*"*<%=basePath%>*"*>  <title>jQuery UI Sortable</title>  <style type=*"text/css"*>  *#myList*{  width: *80px*;  background: *#EEE*;  padding: *5px*;  list-style: *none*;  }  *#myList* **a**{  text-decoration: *none*;  color: *#0077B0*;  }  *#myList* **a***:hover*{  text-decoration: *underline*;  }  *#myList* *.qlink*{  font-size: *12px*;  color: *#666*;  margin-left: *10px*;  }  </style>  </head>    <body>  <ul id=*"myList"*>  <li><a href=*"#"*>心情</a></li>  <li>  <a href=*"#"*>相册</a>  <a href=*"#"* class=*"qlink"*>上传</a>  </li>  <li>  <a href=*"#"*>日志</a>  <a href=*"#"* class=*"qlink"*>发表</a>  </li>  <li><a href=*"#"*>投票</a></li>  <li><a href=*"#"*>分享</a></li>  <li><a href=*"#"*>群组</a></li>  </ul>  <script type=*"text/javascript"* src=*"js/jquery-1.7.2.js"*></script>  <script type=*"text/javascript"* src=*"js/jquery.ui.core.js"*></script>  <script type=*"text/javascript"* src=*"js/jquery.ui.widget.js"*></script>  <script type=*"text/javascript"* src=*"js/jquery.ui.mouse.js"*></script>  <script type=*"text/javascript"* src=*"js/jquery.ui.sortable.js"*></script>  <script type=*"text/javascript"*>  $(document).ready(**function**(){  $("#myList").sortable(); //直接让myList下的元素可以拖动排序  });  </script>  </body>  </html> |



此段代码可实现上图中各项目的随意拖拽。

### 与单击事件的冲突

在某些特殊情况下会因为拖动事件抢在单击事件之前而导致单击事件失效。如果出现这种情

况，可以设置参数delay 延时1 毫秒，即改为：

|  |
| --- |
| $(document).ready(**function**(){  $("#myList").sortable({delay:1}); //直接让myList下的元素可以拖动排序  }); |