# jQuery笔记

学习参考网站：<http://www.runoob.com/jquery/jquery-tutorial.html>。

<http://www.w3school.com.cn/jquery/>

jQuery API手册：<http://www.runoob.com/manual/jquery/>

## jQuery简介

|  |
| --- |
| jQuery 是一个 JavaScript 库。  jQuery 极大地简化了 JavaScript 编程。  jQuery 很容易学习。 |

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

|  |
| --- |
| 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 |

jQuery的核心特性可以总结为：具有独特的链式语法和短小清晰的多功能接口；具有高效灵活的css选择器，并且可对CSS选择器进行扩展；拥有便捷的插件扩展机制和丰富的插件。jQuery兼容各种主流浏览器，如IE 6.0+、FF 1.5+、Safari 2.0+、Opera 9.0+等。

jQuery 团体知道JS在不同浏览器中存在着大量的兼容性问题，目前jQuery**兼容于所有主流浏览器**, 包括Internet Explorer 6。

## jQuery安装

### 方式一

jQuery 库是一个 JavaScript 文件，可以使用 HTML 的 <script> 标签引用下载好的jQuery文件：

|  |
| --- |
| <script src="jquery-1.10.2.min.js"></script> |

可以从 [jquery.com](http://jquery.com/download/) 下载 jQuery 库：

* 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）

### 方式二

如果不希望下载并存放 jQuery，那么也可以通过 CDN（内容分发网络） 引用它。

百度、又拍云、新浪、谷歌和微软的服务器都存有 jQuery 。

如果你的站点用户是国内的，建议使用百度、又拍云、新浪等国内CDN地址，如果你站点用户是国外的可以使用谷歌和微软。

* 菜鸟教程CDN：

|  |
| --- |
| <script src="http://cdn.static.runoob.com/libs/jquery/1.10.2/jquery.min.js"> </script> |

* 百度 CDN：

|  |
| --- |
| <script src="https://apps.bdimg.com/libs/jquery/2.1.4/jquery.min.js"> </script> |

* Google CDN:

|  |
| --- |
| <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"> </script> |

说明：

|  |
| --- |
| **使用方式二，有一个很大的优势（上线推荐使用，本地调试或者断网状态推荐使用方式一）**：  许多用户在访问其他站点时，已经从百度、新浪、谷歌或微软加载过 jQuery。所以结果是，当用户访问您的站点时，会从缓存中加载 jQuery，这样可以减少加载时间。同时，大多数 CDN 都可以确保当用户向其请求文件时，会从离用户最近的服务器上返回响应，这样也可以提高加载速度。 |

## jQuery 语法

jQuery 语法是通过选取 HTML 元素，并对选取的元素执行某些操作。

**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

案例：

|  |
| --- |
| $(this).hide()-------隐藏当前元素  $("p").hide()-------隐藏所有 <p> 元素  $("p.test").hide()-------隐藏所有 class="test" 的 <p> 元素  $("#test").hide()-------隐藏所有 id="test" 的元素 |

### 文档就绪事件

|  |
| --- |
| **$(document).ready(function(){**  **// 开始写 jQuery 代码...**  **});** |

简洁写法（与以上写法效果相同）：

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

以上两种方式可以选择你喜欢的方式实现文档就绪后执行 jQuery 方法。

### 3.2. DOM对象与jQuery对象转换

jQuery对象就是通过jQuery包装DOM对象后产生的对象。

DOM对象就是使用JavaScript中的DOM API操作获得的元素对象。

jQuery对象转换成DOM对象方式一：

|  |
| --- |
| var $div = $("div");  var div = $div[0];  alert(div.innerHTML); |

jQuery对象转换成DOM对象方式二：

|  |
| --- |
| var $div = $("div");  var div = $div.get(0);  alert(div.innerHTML); |

DOM对象转换成jQuery对象方法：

|  |
| --- |
| var div = document.getElementById("id");  var $div = $(div); |

## jQuery 选择器

jQuery 选择器允许您对 HTML 元素组或单个元素进行操作。

jQuery 选择器基于元素的 id、类、类型、属性、属性值等"查找"（或选择）HTML 元素。 它基于已经存在的 CSS 选择器，除此之外，它还有一些自定义的选择器。

jQuery 中**所有选择器**都以**美元符号开头**：**$()**。

### 元素选择器

jQuery 元素选择器**基于元素名选取**元素。

如：在页面中选取所有 <p> 元素：$("p")。

案例：用户点击按钮后，所有 <p> 元素都隐藏

|  |
| --- |
| **$(document).ready(function(){  $("button").click(function(){  $("p").hide();  }); });** |

### id 选择器

jQuery #id 选择器通过 HTML 元素的 id 属性选取指定的元素。

页面中元素的 **id 应该是唯一**的，所以您要在页面中选取唯一的元素需要通过 #id 选择器。

通过 id 选取元素语法如下：**$("#test")**

案例：当用户点击按钮后，有 id="test" 属性的元素将被隐藏

|  |
| --- |
| **$(document).ready(function(){  $("button").click(function(){  $("#test").hide();  }); });** |

### 类选择器(class选择器)

jQuery 类选择器可以通过指定的 class 查找元素。

语法如下：**$(".test")**

案例：用户点击按钮后所有带有 class="test" 属性的元素都隐藏

|  |
| --- |
| **$(document).ready(function(){  $("button").click(function(){  $(".test").hide();  }); });** |

### 常用选择器

|  |  |  |
| --- | --- | --- |
| **语法** | **描述** | **实例** |
| **$("\*")** | **选取所有元素** | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_all2) |
| **$(this)** | **选取当前 HTML 元素** | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_this) |
| **$("p.intro")** | **选取 class 为 intro 的 <p> 元素** | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_pclass) |
| **$("p:first")** | **选取第一个 <p> 元素** | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_pfirst) |
| **$("ul li:first")** | **选取第一个 <ul> 元素的第一个 <li> 元素** | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_ullifirst) |
| **$("ul li:first-child")** | **选取每个 <ul> 元素的第一个 <li> 元素** | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_ullifirstchild) |
| **$("[href]")** | **选取带有 href 属性的元素** | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_hrefattr) |
| **$("a[target='\_blank']")** | **选取所有 target 属性值等于 "\_blank" 的 <a> 元素** | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_hrefattrblank) |
| **$("a[target!='\_blank']")** | **选取所有 target 属性值不等于 "\_blank" 的 <a> 元素** | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_hrefattrnotblank) |
| **$(":button")** | **选取所有 type="button" 的 <input> 元素 和 <button> 元素** | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_button2) |
| **$("tr:even")** | **选取偶数位置的 <tr> 元素** | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_even) |
| **$("tr:odd")** | **选取奇数位置的 <tr> 元素** | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_odd) |

## jQuery事件

**页面对不同访问者的响应叫做事件**。

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。

在事件中经常使用术语"触发"（或"激发"）例如： "当您按下按键时触发 keypress 事件"。

常见 DOM 事件：

|  |  |  |  |
| --- | --- | --- | --- |
| **鼠标事件** | **键盘事件** | **表单事件** | **文档/窗口事件** |
| **click** | **keypress** | **submit** | **load** |
| **dblclick** | **keydown** | **change** | **resize** |
| **mouseenter** | **keyup** | **focus** | **scroll** |
| **mouseleave** |  | **blur** | **unload** |

### jQuery事件语法

在 jQuery 中，大多数 DOM 事件都有一个等效的 jQuery 方法。

页面中指定一个点击事件： **$("p").click()**;

下一步是定义什么时间触发事件。您可以通过一个事件函数实现：

|  |
| --- |
| **$("p").click(function () {  // 动作触发后执行的代码!!  });** |

### 常用的jQuery事件方法

* $(document).ready()：允许我们在文档完全加载完后执行函数
* click()：当按钮点击事件被触发时会调用一个函数
* dblclick()：当双击元素时，会发生 dblclick 事件
* mouseenter()：当鼠标指针穿过元素时，会发生 mouseenter 事件
* mouseleave()：当鼠标指针离开元素时，会发生 mouseleave 事件
* mousedown()：当鼠标指针移动到元素上方，并按下鼠标按键时，会发生 mousedown 事件
* hover()：用于模拟光标悬停事件
* focus()：当元素获得焦点时，发生 focus 事件
* blur()：元素失去焦点时，发生 blur 事件

### $(document).ready()与window.onload

详情可以参考：<http://yunlian0621.iteye.com/admin/blogs/2360461>

|  |
| --- |
| **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，只会执行最后一个。  $(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(){}); |

$(document).ready(function(){})不一定要等所有的js和图片加载完毕，就可以执行一些方法。不过有些时候，必须要等所有的元素都加载完毕，才可以执行一些方法的时候：比如说，部分图片或者什么其他方面还没有加载好，这个时候，点击某些按钮，会导致出现意外的情况，这个时候，我们可以采用$(window).load(function(){...})，**$(window).load(function(){...})是在页面所有元素(包括html标签以及引用到的所有图片、Flash等媒体)加载完毕后执行的**。

Load() 方法会在元素的 onload 事件中**绑定一个处理函数**。

**如果处理函数绑定给 window 对象**，则会在所有内容 ( ***包括窗口、框架、对象和图像等*** ) 加载完毕后触发；

**如果处理函数绑定在元素上**，则会在元素的内容加载完毕后触发

## jQuery效果

### 隐藏、显示

#### 6.1.1 show()、hide()

|  |
| --- |
| **$(document).ready(function () {  $(".showAll").hide();//默认隐藏  //点击显示全部  $(".needAll").click(function () {  $(".showLess").hide();//隐藏  $(".showAll").show();//显示  });  //点击收起  $(".needLess").click(function () {  $(".showAll").hide();//隐藏  $(".showLess").show();//显示  }); });** |
| **<p class="showLess">**  这些代码也是jQuery语法的最初雏形。当时John的想法很简单：他发现这种语法相对现有的JavaScript库更为简洁。但他没想到的是，这篇文章一经发布就引起了业界的关注。于是John开始认真思考着这件事情（编写语法更为简洁的JavaScript程序库），直到2006年1月14日，John正式宣布以jQuery的名称发布自己的程序库。随之而来的是jQuery的快速发展。  2006年1月John Resig等入创建了jQuery......  **<a href="javascript:void(0)" class="needAll">显示全部</a>** **</p> <p class="showAll">**  这些代码也是jQuery语法的最初雏形。当时John的想法很简单：他发现这种语法相对现有的JavaScript库更为简洁。但他没想到的是，这篇文章一经发布就引起了业界的关注。于是John开始认真思考着这件事情（编写语法更为简洁的JavaScript程序库），直到2006年1月14日，John正式宣布以jQuery的名称发布自己的程序库。随之而来的是jQuery的快速发展。  2006年1月John Resig等入创建了jQuery；8月，jQuery的第一个稳定版本，并且已经支持CSS选择符、事件处理和AJAX交互。  2007年7月，jQuery 1.1.3版发布，这次小版本的变化包含了对jQuery选择符引擎执行速度的显著提升。从这个版本开始，jQuery的性能达到了Prototype、Mootools以及Dojo等同类JavaScript库的水平。同年9月，jQuery 1.2版发布，它去掉了对XPath选择符的支持，原因是相对于CSS语法它已经变得多余了。这一版能够对效果进行更为灵活的定制，而且借助新增的命名空间事件，也使插件开发变得更容易。同时，jQuery UI项目也开始启动，这个新的套件是作为曾经流行但已过时的Interface插件的替代项目而发布的。jQuery UI中包含大量预定义好的部件（widget），以及一组用于构建高级元素（例如可拖放、拖拽、排序）的工具。  **<a href="javascript:void(0)" class="needLess">收起</a> </p>** |

#### 6.1.1.2 toggle()

通过 jQuery，可以使用 toggle() 方法来自动切换 hide() 和 show() 方法的效果。

如果元素已隐藏，则 toggle () 会向元素添加显示效果。

如果元素已显示，则 toggle () 会向元素添加隐藏效果。

案例：

|  |
| --- |
| **$(document).ready(function () {  $(".second").toggle();//页面加载时，默认隐藏该部分信息  $(".showSecond").click(function () {  $(".first").toggle();  $(".second").toggle();  });  $(".showFirst").click(function () {  $(".first").toggle();  $(".second").toggle();  }); });** |
| **<p class="first">这是部分信息。。。。。。  <a href="javascript:void (0)" class="showSecond">显示全部</a> </p> <p class="second">这是部分信息。。。。。。这是部分信息。。。。。。这是部分信息。。。。。。这是部分信息。。。。。。  <a href="javascript:void (0)" class="showFirst">收起</a> </p>** |

#### 6.1.1.3 bind()

从上面的代码中，可以看出，红色加粗部分的代码是完全一样的，因此我们可以通过bind方法来将点击事件与点击事件触发的函数绑定到指定的元素。

**bind() 方法为被选元素添加一个或多个事件处理程序，并规定事件发生时运行的函数。**

**语法：$(selector).bind(event,data,function)**

* event：事件，必填参数。
* data：传递到函数的额外数据，可选参数。
* function：当事件发生时运行的函数，必填参数。

**语法：$(selector).bind({event:function, event:function, ...})**

|  |
| --- |
| **$(document).ready(function () {  $("button").bind({  click: function () {  $("p").slideToggle();  },  mouseover: function () {  $("body").css("background-color", "red");  },  mouseout: function () {  $("body").css("background-color", "#FFFFFF");  }  }); });** |

因此上一小节中的js代码可以用以下方式实现：

|  |
| --- |
| **$(document).ready(function () {  $(".second").toggle();//页面加载时，默认隐藏该部分信息  $(".showFirst,.showSecond").bind("click", function () {  $(".first").toggle();  $(".second").toggle();  }) });** |

因此，当某些元素触发了相同的事件，并且需要执行相同的事情时，则可以采用bind方法实现，这样不仅减少了代码的复用，也方便项目的后期维护。

### 淡入、淡出

#### 6.2.1 fadeIn()、fadeOut()

fadeIn() 用于淡入已隐藏的元素。

fadeOut() 方法用于淡出可见元素

**语法：**

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

**$(selector).fadeOut(speed,callback);**

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 fading 完成后所执行的函数名称。

案例：

|  |
| --- |
| **$(document).ready(function () {  $(".fadeInDemo").click(function () {  $("#div1").fadeIn();  $("#div2").fadeIn("slow");//效果不明显  $("#div3").fadeIn(9000);//效果明显  });  $(".fadeOutDemo").click(function () {  $("#div1").fadeOut();  $("#div2").fadeOut("slow");//效果不明显  $("#div3").fadeOut(5000);//效果明显  }); });** |
| **<p>以下实例演示了淡入淡出使用了不同参数的效果。</p> <button class="fadeInDemo">点击淡入 div 元素。</button> <button class="fadeOutDemo">点击淡出 div 元素。</button> <br><br> <div id="div1" style="width:80px;height:80px;display:none;background-color:red;"></div> <br> <div id="div2" style="width:80px;height:80px;display:none;background-color:green;"></div> <br> <div id="div3" style="width:80px;height:80px;display:none;background-color:blue;"></div>** |

#### 6.2.2 fadeToggle()

fadeToggle() 方法可以在 fadeIn() 与 fadeOut() 方法之间进行切换。

如果元素已淡出，则 fadeToggle() 会向元素添加淡入效果。

如果元素已淡入，则 fadeToggle() 会向元素添加淡出效果。

**语法：$(selector).fadeToggle(speed,callback)**

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 fading 完成后所执行的函数名称。

案例：

|  |
| --- |
| **$(".autoFade").click(function () {  $("#div1").fadeToggle();  $("#div2").fadeToggle("slow");//效果不明显  $("#div3").fadeToggle(5000);//效果明显 });** |

#### 6.2.3 fadeTo()

fadeTo() 方法允许渐变为给定的不透明度（值介于 0 与 1 之间）。

**语法：$(selector).fadeTo(speed,opacity,callback);**

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

fadeTo() 方法中必需的 opacity 参数将淡入淡出效果设置为给定的不透明度（值介于 0 与 1 之间）。

可选的 callback 参数是该函数完成后所执行的函数名称。

案例：

|  |
| --- |
| **$(".fadeToDemo").click(function(){  $("#div1").fadeTo("fast",0.15);  $("#div2").fadeTo("slow",0.4);  $("#div3").fadeTo(5000,0.05);//值越小，越透明，颜色越淡 });** |

### 动画

我们可以采用jQuery的animate()方法创建自定义动画。

默认情况下，所有 HTML 元素的位置都是静态的，并且无法移动。如需对位置进行操作，**记得首先把元素的 CSS的position 属性设置为relative、fixed 或 absolute**。

**语法规则：$(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。

案例1：将div元素左移250像素

|  |
| --- |
| **$(document).ready(function(){  $("button").click(function(){  $("div").animate({left:'250px'});  }); });** |
| **<div style="background:#98bf21;height:100px;width:100px;position:absolute;"> </div>** |

案例2：将div变大变透明（操作多个css属性）

|  |
| --- |
| **$(document).ready(function(){  $("button").click(function(){  $("div").animate({  left:'250px',  opacity:'0.5',  height:'150px',  width:'150px'  });  }); });** |

案例3：使用相对值将div变大

|  |
| --- |
| **$(document).ready(function(){  $("button").click(function(){  $("div").animate({  left:'250px',  height:'+=150px',  width:'+=150px'  });  }); });** |

案例4：使用队列功能动态改变大小

|  |
| --- |
| **$(document).ready(function(){  $("button").click(function(){  var div=$("div");  div.animate({height:'300px',opacity:'0.4'},"slow");  div.animate({width:'300px',opacity:'0.8'},"slow");  div.animate({height:'100px',opacity:'0.4'},"slow");  div.animate({width:'100px',opacity:'0.8'},"slow");  }); });** |

案例5：移动div元素，并设置字体样式

|  |
| --- |
| **$(document).ready(function(){  $("button").click(function(){  var div=$("div");  div.animate({left:'100px'},"slow");  div.animate({fontSize:'3em'},"slow");  }); });** |

## HTML（操作DOM）

DOM = Document Object Model（文档对象模型）。

jQuery 中非常重要的部分，就是操作 DOM 的能力。

jQuery 提供一系列与 DOM 相关的方法，这使访问和操作元素和属性变得很容易。

### 获取值

三个简单实用的用于 DOM 操作的 jQuery 方法：

* **text()** - 设置或返回所选元素的文本内容---对应javascript的innerText。
* **html()** - 设置或返回所选元素的内容（包括 HTML 标记）---对应javascript的innerHTML。
* **val()** - 设置或返回表单字段的值---对应javascript的value。
* **attr()**- 获取属性值---对应javascript的getAttribute(“”)。

### 设置值

* **text(“value”)** - 设置或返回所选元素的文本内容---对应javascript的innerText。
* **html(“value”)** - 设置或返回所选元素的内容（包括 HTML 标记）---对应javascript的innerHTML。
* **val(“value”)** - 设置或返回表单字段的值---对应javascript的value。
* **attr(“name”,”value”)**- 获取属性值---对应javascript的setAttribute(“name”,”value”)。

### 添加元素

通过jQuery，可以很容易的添加新元素。jQuery中，用于添加新元素的方法如下：

* append() - 在被选元素的结尾插入内容
* prepend() - 在被选元素的开头插入内容
* after() - 在被选元素之后插入内容
* before() - 在被选元素之前插入内容

|  |
| --- |
| **$(document).ready(function(){  //添加到元素开头  $(".addFirst").click(function () {  $(".navi").prepend("<li>插入到第一个</li>")  });  //添加到元素结尾  $(".addEnd").click(function () {  $(".navi").append("<li>插入到最后一个</li>")  });  //添加到某元素之后  $(".addAfter").click(function () {  $(".navi\_li").after("<li>插入某个元素之后</li>")  });  //添加到某元素之前  $(".addBefore").click(function () {  $(".navi\_li").before("<li>插入某个元素之前</li>")  }); });** |
| <**ul class="navi"**>  <**li**>第一条</**li**>  <**li class="navi\_li"**>第二条</**li**>  <**li**>第三条</**li**> </**ul**> <**button class="addFirst"**>添加到元素开头</**button**> <**button class="addEnd"**>添加到元素结尾</**button**> <**button class="addAfter"**>添加到某元素之后</**button**> <**button class="addBefore"**>添加到某元素之前</**button**> |

### 删除元素

如需删除元素和内容，一般可使用以下两个 jQuery 方法：

* remove() - **删除被选元素**（及其子元素）
* empty() - 从被选元素中**删除子元素**

|  |
| --- |
| **$(document).ready(function(){  //删除子元素  $(".deletePart").click(function () {  $(".vegi").empty();//vegi保留，但是它的li元素没有了  });  //删除某元素  $(".deleteAll").click(function () {  $(".navi").remove();//navi元素没有了  }); });** |
| <**button class="deletePart"**>删除蔬菜的子元素</**button**> <**button class="deleteAll"**>删除某元素</**button**> <**ul class="navi"**>  <**li**>第1条</**li**>  <**li**>蔬菜：  <**ul class="vegi"**>  <**li**>大白菜</**li**>  <**li**>小青菜</**li**>  </**ul**>  </**li**>  <**li**>水果：  <**ul class="fruit"**>  <**li**>苹果</**li**>  <**li**>葡萄</**li**>  </**ul**>  </**li**> </**ul**> <**ul class="navi\_bott"**>  <**li**>第111条</**li**>  <**li**>第222条</**li**> </**ul**> |

### 设置class（设置css类）

jQuery 拥有若干进行 CSS 操作的方法。常用的方法如下：

* addClass() - 向被选元素添加一个或多个类
* removeClass() - 从被选元素删除一个或多个类
* toggleClass() - 对被选元素进行添加/删除类的切换操作
* css() - 设置或返回样式属性

|  |
| --- |
| **<!DOCTYPE html> <html lang="en"> <head>  <meta charset="UTF-8">  <title>设置class</title>  <script src="jquery/jquery-1.11.2.js"></script>  <script>  $(document).ready(function(){  //方式一： // $(".saveBtn").mouseover(function () { // //当光标悬停时 // $(".saveBtn").addClass("dark"); // $(".saveBtn").removeClass("light"); // }); // $(".saveBtn").mouseout(function () { // //当光标离开(移出)后 // $(".saveBtn").addClass("light"); // $(".saveBtn").removeClass("dark"); // });  //方式二：  $(".saveBtn").bind("mouseover mouseout",function () {  $(".saveBtn").toggleClass("dark");  $(".saveBtn").toggleClass("light");  })  });  </script>  <style>  .saveBtn{  width: 150px;/\*按钮的宽度\*/  height: 60px;/\*按钮的高度\*/  background-color: beige;/\*按钮的背景颜色\*/  font-size: 30px;/\*字体大小\*/  font-family: kaiti;/\*字体：楷体\*/  font-weight: bold;/\*字体加粗\*/  color: crimson;/\*字体颜色\*/  border-radius: 10px;/\*边框的圆角\*/  box-shadow: 3px 5px 8px burlywood;/\*阴影效果\*/  }  .dark{  background-color: antiquewhite;  font-size: 38px;  }  .light{  background-color: beige;  }  </style> </head>  <body> <button class="saveBtn light">保存</button> </body> </html>** |

### 设置css

css() 方法设置或返回被选元素的一个或多个样式属性。

* 返回指定的 CSS 属性的值：**css("propertyname");**
* 设置指定的 CSS 属性：**css("propertyname","value");**
* 设置多个 CSS 属性：

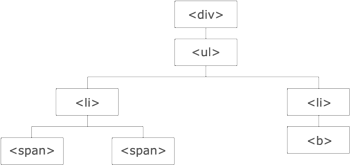
**css({"propertyname":"value","propertyname":"value",...});**

|  |
| --- |
| $(**".saveBtn"**).css(**"background-color"**);  $(**".saveBtn"**).css(**"background-color"**,**"red"**); |

## jQuery遍历

jQuery 遍历，意为"移动"，用于根据其相对于其他元素的关系来"查找"（或选取）HTML 元素。以某项选择开始，并沿着这个选择移动，直到抵达您期望的元素为止。

下图展示了一个家族树。通过 jQuery 遍历，您能够从被选（当前的）元素开始，轻松地在家族树中向上移动（祖先），向下移动（子孙），水平移动（同胞）。这种移动被称为对 DOM 进行遍历。



图示解析：

* <div> 元素是 <ul> 的父元素，同时是其中所有内容的祖先。
* <ul> 元素是 <li> 元素的父元素，同时是 <div> 的子元素
* 左边的 <li> 元素是 <span> 的父元素，<ul> 的子元素，同时是 <div> 的后代。
* <span> 元素是 <li> 的子元素，同时是 <ul> 和 <div> 的后代。
* 两个 <li> 元素是同胞（拥有相同的父元素）。
* 右边的 <li> 元素是 <b> 的父元素，<ul> 的子元素，同时是 <div> 的后代。
* <b> 元素是右边的 <li> 的子元素，同时是 <ul> 和 <div> 的后代。

jQuery 提供了多种遍历 DOM 的方法。遍历方法中最大的种类是树遍历（tree-traversal）。

### 祖先

祖先是父、祖父或曾祖父等等，通过 jQuery，您能够向上遍历 DOM 树，以查找元素的祖先。常用的方法：

**parent()：**parent() 方法返回被选元素的直接父元素

**parents()：**返回被选元素的所有祖先元素，一路向上直到文档的根元素 (<html>)

**parentsUntil()：**返回介于两个给定元素之间的所有祖先元素

|  |
| --- |
| $("span").parent();  $("span").parents();  $("span").parents("ul");//找出span的祖先元素为ul的祖先元素 |

### 后代

后代是子、孙、曾孙等等。通过 jQuery，您能够向下遍历 DOM 树，以查找元素的后代。常用于向下遍历 DOM 树的 jQuery 方法：

* children()：
* find()

|  |
| --- |
| $("div").children();//返回每个 <div> 元素的所有**直接子元素**  $("div").children("p.1");//返回类名为 "1" 的所有 <p> 元素，并且它们是 <div> 的直接子元素  $("div").find("span");//返回属于 <div> **后代**的所有 <span> 元素  $("div").find("\*");//返回 <div> 的**所有后代** |

### 同胞

同胞拥有相同的父元素，通过 jQuery，您能够在 DOM 树中遍历元素的同胞元素。有许多有用的方法让我们在 DOM 树进行水平遍历：

* siblings()：返回被选元素的所有同胞元素
* next()：返回被选元素的下一个同胞元素
* nextAll()：返回被选元素的所有跟随的同胞元素
* nextUntil()：返回介于两个给定参数之间的所有跟随的同胞元素
* prev()
* prevAll()
* prevUntil()

|  |
| --- |
| $("h2").siblings();//返回 <h2> 的所有同胞元素  $("h2").siblings("p");//返回属于 <h2> 的同胞元素的所有 <p> 元素  $("h2").next();//返回 <h2> 的下一个同胞元素  $("h2").nextAll();//返回 <h2> 的所有跟随的同胞元素  $("h2").nextUntil("h6");//返回介于 <h2> 与 <h6> 元素之间的所有同胞元素 |

**prev(), prevAll() 以及 prevUntil() 方法的工作方式与上面的方法类似**，只不过方向相反而已：它们返回的是前面的同胞元素（在 DOM 树中沿着同胞元素向后遍历，而不是向前）。

### 过滤

三个最基本的过滤方法是：first(), last() 和 eq()，它们允许您基于其在一组元素中的位置来选择一个特定的元素。

其他过滤方法（详情可以参考API），比如 filter() 和 not() 允许您选取匹配或不匹配某项指定标准的元素。

|  |
| --- |
| $("div p").first();//选取首个 <div> 元素内部的第一个 <p> 元素 |

## AJAX

jQuery 提供多个与 AJAX 有关的方法。

通过 jQuery AJAX 方法，您能够使用 HTTP Get 和 HTTP Post 从远程服务器上请求文本、HTML、XML 或 JSON - 同时您能够把这些外部数据直接载入网页的被选元素中。

**提示：**如果没有 jQuery，AJAX 编程还是有些难度的。

编写常规的 AJAX 代码并不容易，因为不同的浏览器对 AJAX 的实现并不相同。这意味着您必须编写额外的代码对浏览器进行测试。不过，jQuery 团队为我们解决了这个难题，我们只需要一行简单的代码，就可以实现 AJAX 功能。

### 9.1 GET 和 POST

以下是W3C上的说法：

在客户机和服务器之间进行请求-响应时，两种最常被用到的方法是：GET 和 POST。

GET - 从指定的资源请求数据。

POST - 向指定的资源提交要被处理的数据

#### 9.1.1 GET方法

**请注意，查询字符串（名称/值对）是在 GET 请求的 URL 中发送的：**

/test/demo\_form.asp?name1=value1&name2=value2

有关 GET 请求的其他一些注释：

* GET 请求可被缓存（在url后面添加时间戳避免缓存）
* GET 请求保留在浏览器历史记录中
* GET 请求可被收藏为书签
* GET 请求不应在处理敏感数据时使用
* GET 请求有长度限制
* GET 请求只应当用于取回数据

#### 9.1.2 POST方法

请注意，查询字符串（名称/值对）是在 POST 请求的 HTTP 消息主体中发送的：

POST /test/demo\_form.asp HTTP/1.1

Host: w3schools.com

name1=value1&name2=value2

有关 POST 请求的其他一些注释：

* POST 请求不会被缓存
* POST 请求不会保留在浏览器历史记录中
* POST 不能被收藏为书签
* POST 请求对数据长度没有要求

#### 9.1.3比较 GET 与 POST

下面的表格比较了两种 HTTP 方法：GET 和 POST。

|  |  |  |
| --- | --- | --- |
|  | **GET** | **POST** |
| 后退按钮/刷新 | 无害 | 数据会被重新提交（浏览器应该告知用户数据会被重新提交）。 |
| 书签 | 可收藏为书签 | 不可收藏为书签 |
| 缓存 | 能被缓存 | 不能缓存 |
| 编码类型 | application/x-www-form-urlencoded | application/x-www-form-urlencoded 或 multipart/form-data。  为二进制数据使用多重编码。 |
| 历史 | 参数保留在浏览器历史中。 | 参数不会保存在浏览器历史中。 |
| 对数据长度的限制 | 是的。当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。 | 无限制。 |
| 对数据类型的限制 | 只允许 ASCII 字符。 | 没有限制。也允许二进制数据。 |
| 安全性 | 与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。  在发送密码或其他敏感信息时绝不要使用 GET ！ | POST 比 GET 更安全，因为参数不会被保存在浏览器历史或  web 服务器日志中。 |
| 可见性 | 数据在 URL 中对所有人都是可见的。 | 数据不会显示在 URL 中。 |

### 9.2 jQuery $.get()方法

**语法：$.get(URL,callback);**

|  |
| --- |
| $("button").click(function(){  $.get("admin/user.do?id=5&flag=delete",function(data,status){  alert("Data: " + data + "\nStatus: " + status);  });  }); |

$.get() 的第一个参数是我们希望请求的 URL：

（" admin/user.do?id=5&flag=delete "）。

第二个参数是回调函数：第一个回调参数存有被请求页面的数据内容，第二个回调参数存有请求的状态。

### 9.3 jQuery $.post()方法

**语法：$.post(URL,data,callback);**

|  |
| --- |
| $("button").click(function(){  $.post("admin/user.do",  {  id:5,  flag:"delete"  },  function(data,status){  alert("Data: " + data + "\nStatus: " + status);  });  }); |

## 表单验证框架

* 引入jQuery Validate 插件：[jQuery Validate 官网](http://jqueryvalidation.org/)

表单验证需要引入：

jquery.validate.js（或者jquery.validate.min.js）文件

详情可参考网站：<http://yunlian0621.iteye.com/blog/2362027>

自定义校验需要引入：

additional-methods.js（或者additional-methods.min.js）文件

自定义校验：<http://yunlian0621.iteye.com/blog/2364695>