jQuery+Ajax

18级大数据
王娜
田一岚

jQuery

jQuery 是一个 JavaScript 库。

Gower Point

jQuery 使用 $ 符号作为 jQuery 的简记方式。

例子：

当按钮的点击事件被触发时会调用一个函数：

```
$("button").click(function() {..some code... } )
```

某些其他 JavaScript 库中的函数（比如 Prototype）同样使用 $ 符号。

jQuery 使用名为 noConflict() 的方法来解决该问题。

例如：var jq=jQuery.noConflict()，用（jq）来代替 $ 符号。

For those who have seen the Earth from space, and for the hundreds and perhaps thousands

jQuery 事件

下面是 jQuery 中事件方法的一些例子：

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).mouseover(function)	触发或将函数绑定到被选元素的鼠标悬停事件

jQuery - 获得内容

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
- html() - 设置或返回所选元素的内容（包括 HTML 标记）
- val() - 设置或返回表单字段的值

例子：

```
$("#btn1").click(function(){ alert("Text: " + $("#test").text()); });
$("#btn2").click(function(){ alert("HTML: " + $("#test").html()); });
```

jQuery设置内容


```
$("#btn1").click(function(){ $("#test1").text("Hello world!"); });
$("#btn2").click(function(){ $("#test2").html("<b>Hello world!</b>"); });
$("#btn3").click(function(){ $("#test3").val("Dolly Duck"); });
```

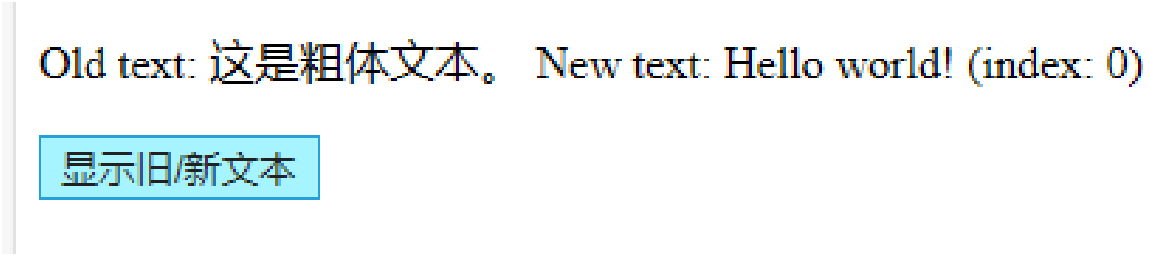
text()、html() 以及 val() 的回调函数

这三个 jQuery 方法：text()、html() 以及 val()的回调函数由两个参数：被选元素列表中当前元素的下标，以及原始（旧的）值。然后以函数新值返回希望使用的字符串。

实例：

```
$("#btn1").click(function(){
        $("#test1").text(function(i,origText){
                return "Old text: " + origText +
                        " New text: Hello world! (index: " + i + ")";
        });
});
```

```
<body>
<p id="test1">这是<b>粗体</b>文本。</p>
<button id="btn1">显示旧/新文本</button>
</body>
</html>
```



Gower Point

jQuery获取属性

jQuery attr() 方法用于获取属性值。

例子:

```
$("button").click(function(){ alert($("#w3s").attr("href"));});
```

For those who have seen the Earth from space, and for the hundreds and perhaps thousands

jQuery设置属性 - attr()

设置单个属性

```
$("button").click(function(){
        $("#w3s").attr("href",http://www.w3school.com.cn/jquery
        );
});
```

设置多个属性

```
$("button").click(function(){
$("#w3s").attr({
"href" : "http://www.w3school.com.cn/jquery",
"title":"W3School jQuery Tutorial“
        });
});
```

jQuery - 添加元素

添加新的 HTML 内容

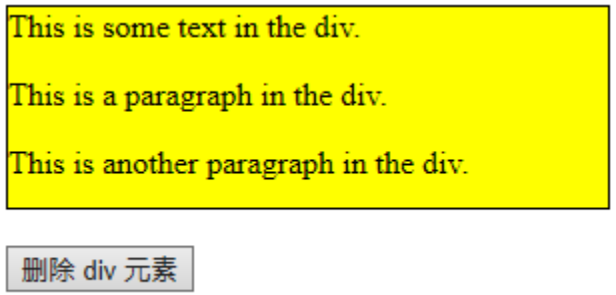
- append() - 在被选元素的结尾插入内容
- prepend() - 在被选元素的开头插入内容
- after() - 在被选元素之后插入内容
- before() - 在被选元素之前插入内容

```
$("p").append("Some appended text.");
$("p").prepend("Some prepended text.");
$("img").after("Some text after");
$("img").before("Some text before");
```

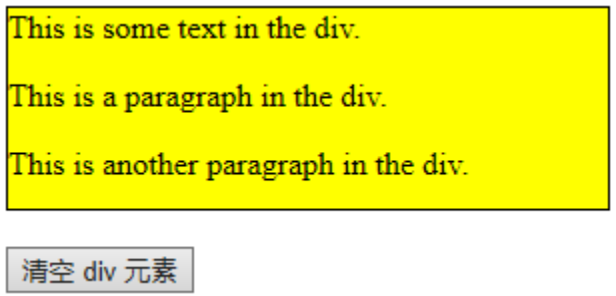
jQuery - 删除元素

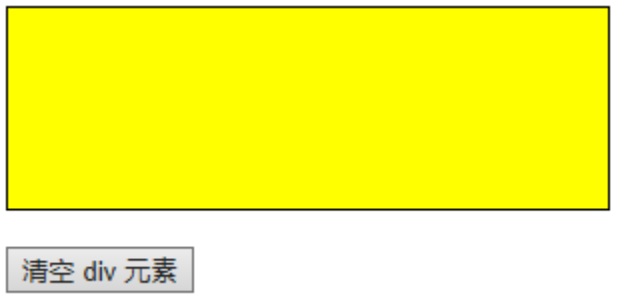
- remove() - 删除被选元素（及其子元素）
- empty() - 从被选元素中删除子元素

```
$("button").click(function(){
  $("#div1").remove();
 });
```




```
$("button").click(function(){
  $("#div1").empty();
 });
```



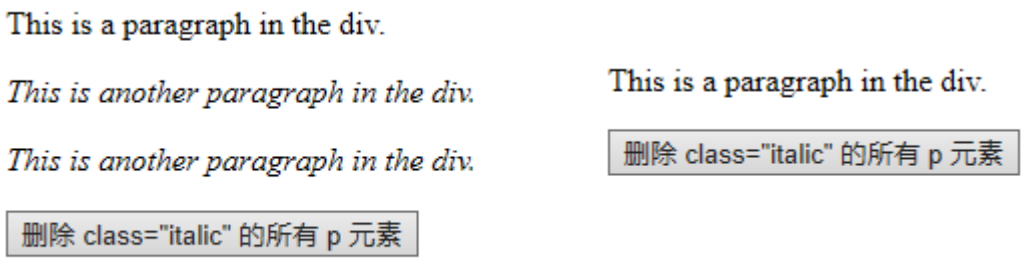


jQuery过滤被删除的元素

jQuery remove() 方法也可接受一个参数，允许对被删元素进行过滤。该参数可以是任何 jQuery 选择器的语法。

实例（删除 class="italic" 的所有 <p> 元素）

```
$("p").remove(".italic");
```



jQuery - 获取并设置 CSS 类

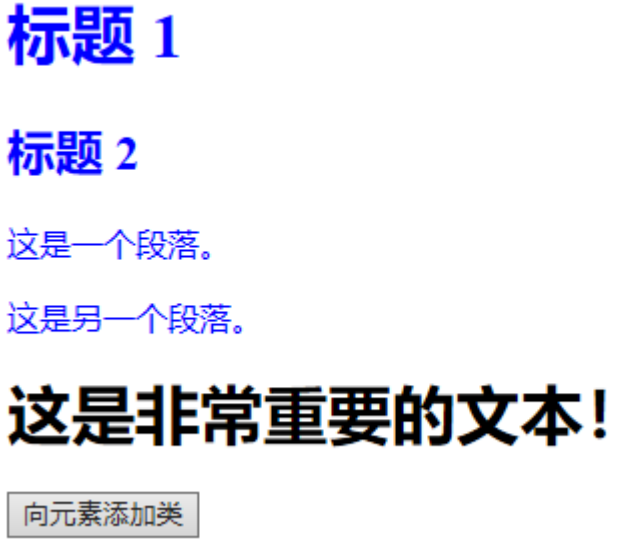
通过 jQuery，可以很容易地对 CSS 元素进行操作。

- 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

实例：

```
.important { font-weight:bold; font-size:xx-
large; }
.blue { color:blue; }

$("button").click(function(){ $("h1,h2,p").addC
lass("blue");
$("div").addClass("important"); });
```



jQuery - css() 方法

返回 CSS 属性 css("propertyname");

实例（将返回首个匹配元素的 background-color 值）
$("p").css("background-color");

设置 CSS 属性 css("propertyname","value");

实例（为所有匹配元素设置 background-color 值）
$("p").css("background-color","yellow");

设置多个 CSS 属性 css({"propertyname":"value","propertyname":"value",...});

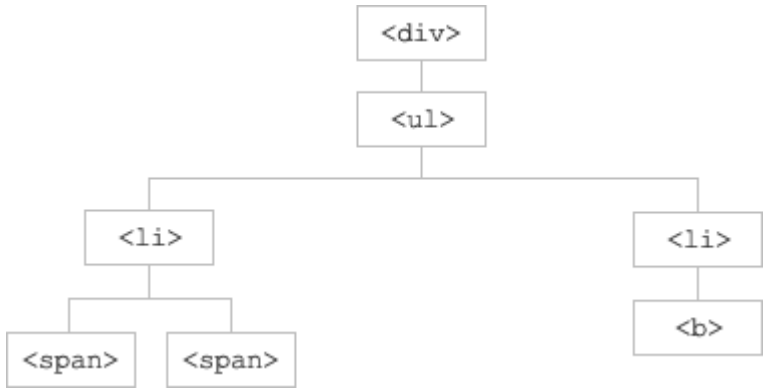
Gower Point

jQuery 遍历

什么是遍历？

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，直到抵达期望的元素为止。

下图展示了一个家族树。通过 jQuery 遍历，能够从被选（当前的）元素开始，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。



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

jQuery 遍历 - 祖先

- parent() 返回被选元素的直接父元素，只会向上一级对 DOM 树进行遍历。
- parents() 返回被选元素的所有祖先元素，它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。

实例（返回介于 与 元素之间的所有祖先元素）

```
$(document).ready(function(){
  $("span").parentsUntil("div");
});
```

实例（返回所有 元素的所有祖先，并且它是 <ul> 元素）

```
$(document).ready(function(){
  $("span").parents("ul");
});
```

jQuery 遍历 - 后代

children()　　返回被选元素的所有直接子元素，该方法只会向下一级对 DOM 树进行遍历。

find()　　返回被选元素的后代元素，一路向下直到最后一个后代。

实例（返回 的所有后代）

```
$(document).ready(function(){
  $("div").find("*");
});
```

jQuery 遍历 - 同胞

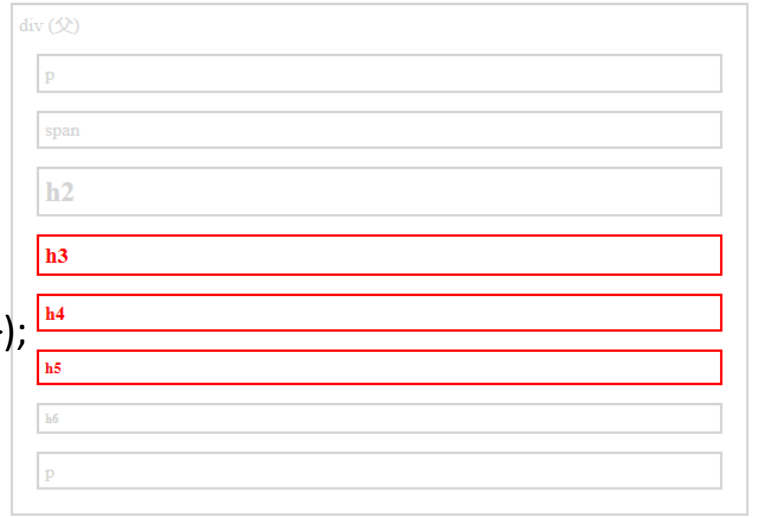
Gower Point

For those who have seen the Earth from space, and for the hundreds and perhaps thousands

- 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。该方法只返回一个元素。
- 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() 返回被选元素的前面一个同胞元素
- prevAll() 返回被选元素的前面所有同胞元素
- prevUntil() 返回介于两个给定参数之间的所有前面的同胞元素

```
<div>div (父)
  <p>p</p>
  <span>span</span>
  <h2>h2</h2>
  <h3>h3</h3>
  <h4>h4</h4>
  <h5>h5</h5>
  <h6>h6</h6>
  <p>p</p>
</div>
```

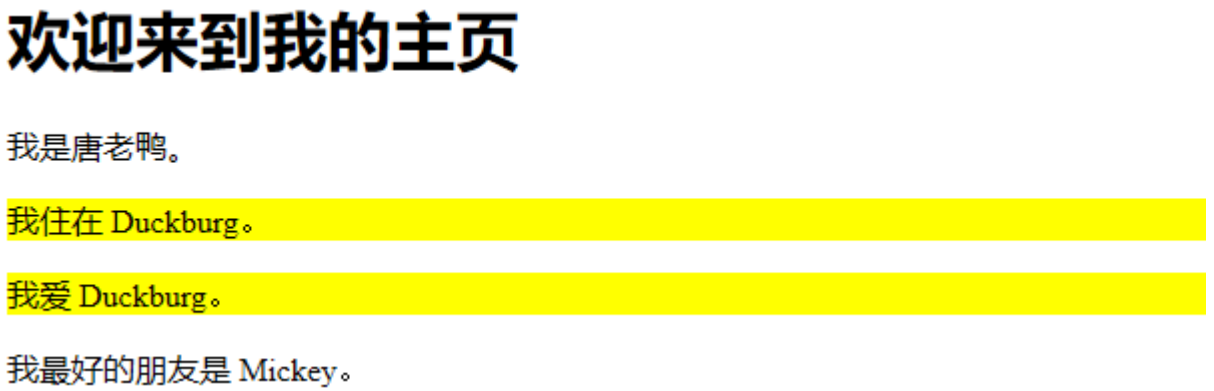
```
$(document).ready(function(){
  $("h2").nextUntil("h6").css({"color":"red","border":"2px solid red"});
});
```



jQuery 遍历 - 过滤

first()	返回被选元素的首个元素。
last()	返回被选元素的最后一个元素
eq()	返回被选元素中带有指定索引号的元素，索引号从 0 开始
Filter（）	返回不匹配这个标准的元素会被从集合中删除，匹配的元素会被返回。
Not（）	返回不匹配标准的所有元素。

```
<body>
<h1>欢迎来到我的主页</h1>
<p>我是唐老鸭。</p>
<p class="intro">我住在 Duckburg。</p>
<p class="intro">我爱 Duckburg。</p>
<p>我最好的朋友是 Mickey。</p>
</body>
```




```
$(document).ready(function(){
  $("p").filter(".intro").css("background-color","yellow");
});
```

Ajax

AJAX简介

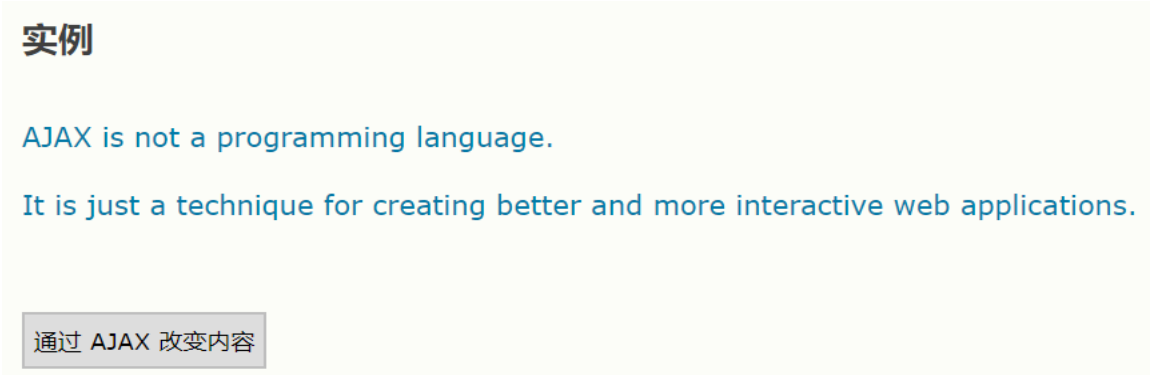
AJAX即“Asynchronous Javascript And XML”(异步JavaScript 和XML)，是一种用于创建快速动态网页的技术。

通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

传统的网页（不使用 AJAX）如果需要更新内容，必需重载整个网页面。

有很多使用 AJAX 的应用程序案例：新浪微博、Google 地图、开心网等等。

AJAX 实例



上面的 AJAX 应用程序包含一个 div 和一个按钮。div 部分用于显示来自服务器的信息。当按钮被点击时，它负责调用名为 loadXMLDoc() 的函数：

```
<html>
<body>
<div id="myDiv"><h3>Let AJAX change this text</h3></div>
<button type="button" onclick="loadXMLDoc()">Change
Content</button>
</body>
</html>
```

在页面的 head 部分有一个 <script> 标签。该标签中包含了这个 loadXMLDoc() 函数：

```
<head>
<script type="text/javascript">
function loadXMLDoc()
{
.... AJAX script goes here ...
}
</script>
</head>
```

Gower Point

For those who have seen the Earth from space, and for the hundreds and perhaps thousands

AJAX基础——XMLHttpRequest

XMLHttpRequest 用于在后台与服务器交换数据。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

创建 XMLHttpRequest 对象的语法：
variable=new XMLHttpRequest();

```
var xmlhttp;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
```

为了应对所有的现代浏览器，包括 IE5 和 IE6，请检查浏览器是否支持 XMLHttpRequest 对象。如果支持，则创建 XMLHttpRequest 对象。如果不支持，则创建 ActiveXObject：

AJAX - 向服务器发送请求

将请求发送到服务器，我们使用 XMLHttpRequest 对象的 open() 和 send() 方法：

```
xmlhttp.open("GET","test1.txt",true);
xmlhttp.send();
```

方法	描述
open(*method*,*url*,*async*)	规定请求的类型、URL 以及是否异步处理请求。 •*method*：请求的类型；GET 或 POST •*url*：文件在服务器上的位置 •*async*：true（异步）或 false（同步）
send(*string*)	将请求发送到服务器。 •*string*：仅用于 POST 请求

AJAX - 向服务器发送请求

GET vs POST

与 POST 相比，GET 更简单也更快，并且在大部分情况下都能用。

然而，在以下情况中，请使用 POST 请求：
1.无法使用缓存文件（更新服务器上的文件或数据库）
2.向服务器发送大量数据（POST 没有数据量限制）
3.发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

Gower Point

AJAX - 向服务器发送请求-get

For those who have seen the Earth from space, and for the hundreds and perhaps thousands

一个简单的 GET 请求：

```
xmlhttp.open("GET","demo_get.asp",true);
xmlhttp.send();
```

通过 GET 方法发送信息，请向 URL 添加信息：

```
xmlhttp.open("GET","demo_get2.asp?fname=Bill&lname=Gates",true);
xmlhttp.send();
```

AJAX - 向服务器发送请求-post

一个简单 POST 请求：

```
xmlhttp.open("POST","demo_post.asp",true);
xmlhttp.send();
```

如果需要POST 数据，则使用 setRequestHeader() 来添加 HTTP 头。然后在 send() 方法中规定希望发送的数据：

```
xmlhttp.open("POST","ajax_test.asp",true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send("fname=Bill&lname=Gates");
```

方法	描述
setRequestHeader(*header*,*value*)	向请求添加 HTTP 头。 •*header*: 规定头的名称 •*value*: 规定头的值

AJAX - 向服务器发送请求

XMLHttpRequest 对象如果要用于 AJAX 的话，其 open() 方法的 async 参数必须设置为 true。

当使用 async=true 时，要规定在响应处于 onreadystatechange 事件中的就绪状态时执行的函数。

通过 AJAX，JavaScript 无需等待服务器的响应，而是：

- 在等待服务器响应时执行其他脚本
- 当响应就绪后对响应进行处理

当使用 async=false 时，不需要编写 onreadystatechange 函数 - 把代码放到 send() 语句后面即可：

```
xmlhttp.open("GET","test1.txt",false);
xmlhttp.send();
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

AJAX - 服务器响应

属性	描述
responseText	获得字符串形式的响应数据。
responseXML	获得 XML 形式的响应数据。

responseText 属性：返回字符串形式的响应，可以这样使用：
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

responseXML 属性：如果来自服务器的响应是 XML，而且需要作为 XML 对象进行解析，使用 responseXML 属性(详见后面完整例子）

```
xmlDoc=xmlhttp.responseXML;
txt="";
x=xmlDoc.getElementsByTagName("ARTIST");
for (i=0;i<x.length;i++) {
txt=txt + x[i].childNodes[0].nodeValue + "<br />"; }
document.getElementById("myDiv").innerHTML=txt;
```

AJAX - onreadystatechange 事件

当请求被发送到服务器时，我们需要执行一些基于响应的任务。
每当 readyState 改变时，就会触发 onreadystatechange 事件。
readyState 属性存有 XMLHttpRequest 的状态信息。
下面是 XMLHttpRequest 对象的三个重要的属性：

属性	描述
onreadystatechange	存储函数（或函数名），每当 readyState 属性改变时，就会调用该函数。
readyState	存有 XMLHttpRequest 的状态。从 0 到 4 发生变化。 • 0: 请求未初始化 • 1: 服务器连接已建立 • 2: 请求已接收 • 3: 请求处理中 • 4: 请求已完成，且响应已就绪
status	200: "OK" 404: 未找到页面

AJAX - onreadystatechange 事件

Gower Point

For those who have seen the Earth from space, and for the hundreds and perhaps thousands

在 onreadystatechange 事件中，我们规定当服务器响应已做好被处理的准备时所执行的任务。当 readyState 等于 4 且状态为 200 时，表示响应已就绪：

```
xmlhttp.onreadystatechange=function()
 {
 if (xmlhttp.readyState==4 && xmlhttp.status==200)
  {
          执行触发事件
  }
 }
```

/books.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Copyright w3school.com.cn -->
<!-- W3School.com.cn bookstore example -->
- <bookstore>
  - <book category="children">
      <title lang="en">Harry Potter</title>
      <author>J K. Rowling</author>
      <year>2005</year>
      <price>29.99</price>
    </book>
  - <book category="cooking">
      <title lang="en">Everyday Italian</title>
      <author>Giada De Laurentiis</author>
      <year>2005</year>
      <price>30.00</price>
    </book>
  - <book category="web" cover="paperback">
      <title lang="en">Learning XML</title>
      <author>Erik T. Ray</author>
      <year>2003</year>
      <price>39.95</price>
    </book>
  + <book category="web">
</bookstore>
```

AJAX – 实例

Gower Point

For those who have seen the Earth from space, and for the hundreds and perhaps thousands

1.

```
function loadXMLDoc()
{
var xmlhttp;
var txt,x,i;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
```

2.

```
xmlhttp.open("GET","/example/xmle/books.xml",true);
xmlhttp.send();
```

3.

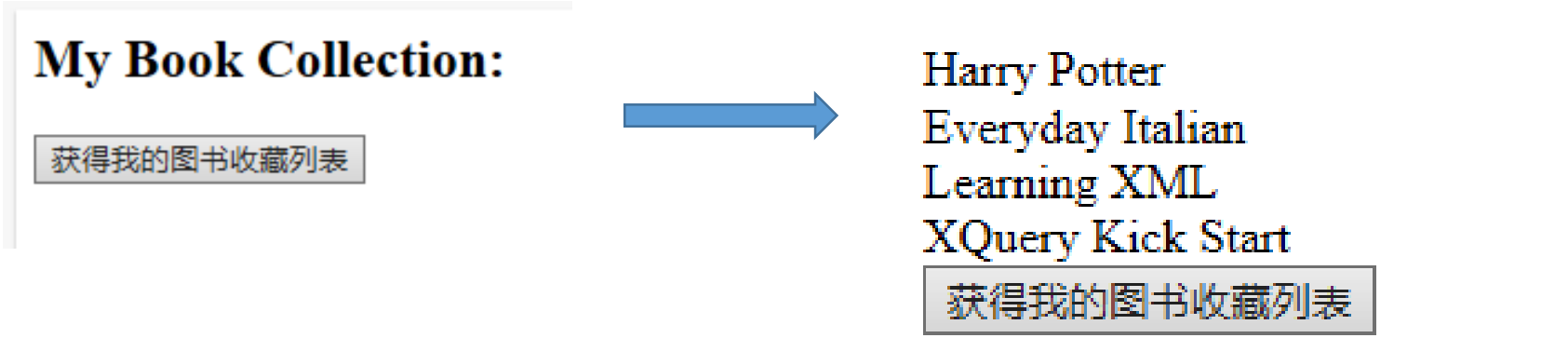
```
xmlhttp.onreadystatechange=function()
 {
 if (xmlhttp.readyState==4 && xmlhttp.status==200)
  {
  xmlDoc=xmlhttp.responseXML;
  txt="";
  x=xmlDoc.getElementsByTagName("title");
  for (i=0;i<x.length;i++)
   {
   txt=txt + x[i].childNodes[0].nodeValue + "<br />";
   }
  document.getElementById("myDiv").innerHTML=txt;
  }
 }
```

AJAX - onreadystatechange 事件

Gower Point

For those who have seen the Earth from space, and for the hundreds and perhaps thousands

```
<body>
<h2>My Book Collection:</h2>
<div id="myDiv"></div>
<button type="button" onclick="loadXMLDoc()">获得我的图书收藏列表</button>
</body>
```



jQuery - AJAX 之 load() 方法

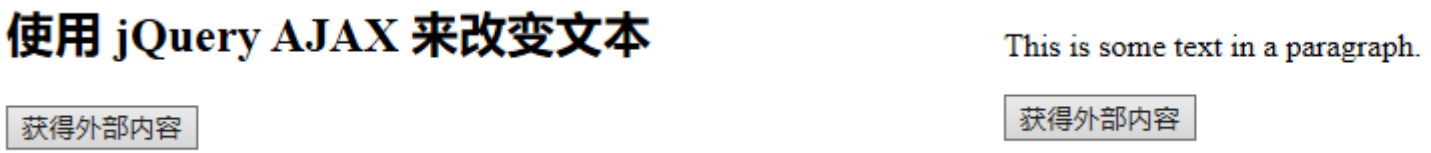
jQuery load() 方法 从服务器加载数据，并把返回的数据放入被选元素中。

语法：
$(selector).load(URL,data,callback);
必需的 URL 参数规定您希望加载的 URL。
可选的 data 参数规定与请求一同发送的查询字符串键/值对集合。
可选的 callback 参数是 load() 方法完成后所执行的函数名称。

实例（把 “demo_test.txt” 文件中 id=“p1” 的元素的内容，加载到指定的 元素中）

```
$("#div1").load("demo_test.txt #p1");
```

```
<div id="div1"><h2>使用 jQuery AJAX 来改变文本</h2></div>
<button>获得外部内容</button>
```



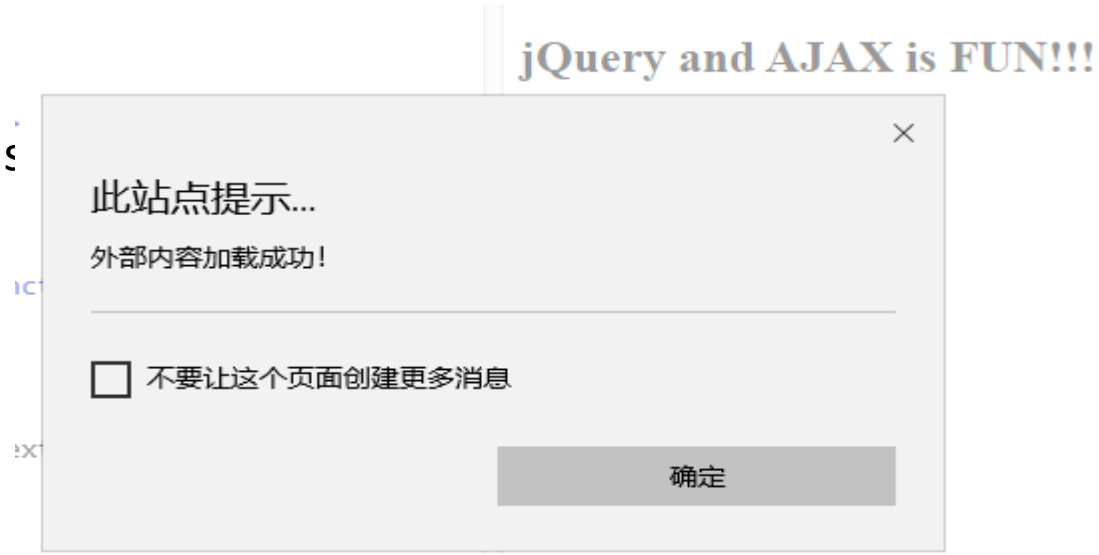
jQuery - AJAX 之 load() 方法

可选的 callback 参数规定当 load() 方法完成后所要允许的回调函数。回调函数可以设置不同的参数：

responseTxt - 包含调用成功时的结果内容
statusTXT - 包含调用的状态
xhr - 包含 XMLHttpRequest 对象

实例（在 load() 方法完成后显示一个提示框。如果 load() 方法已成功，则显示“外部内容加载成功！”，而如果失败，则显示错误消息）

```
$("button").click(function(){
  $("#div1").load("demo_test.txt",function(respons
    if(statusTxt=="success")
      alert("外部内容加载成功！");
    if(statusTxt=="error")
      alert("Error: "+xhr.status+": "+xhr.statusText);
  });
});
```



jQuery $.get() 方法

$.get() 方法通过 HTTP GET 请求从服务器上请求数据。

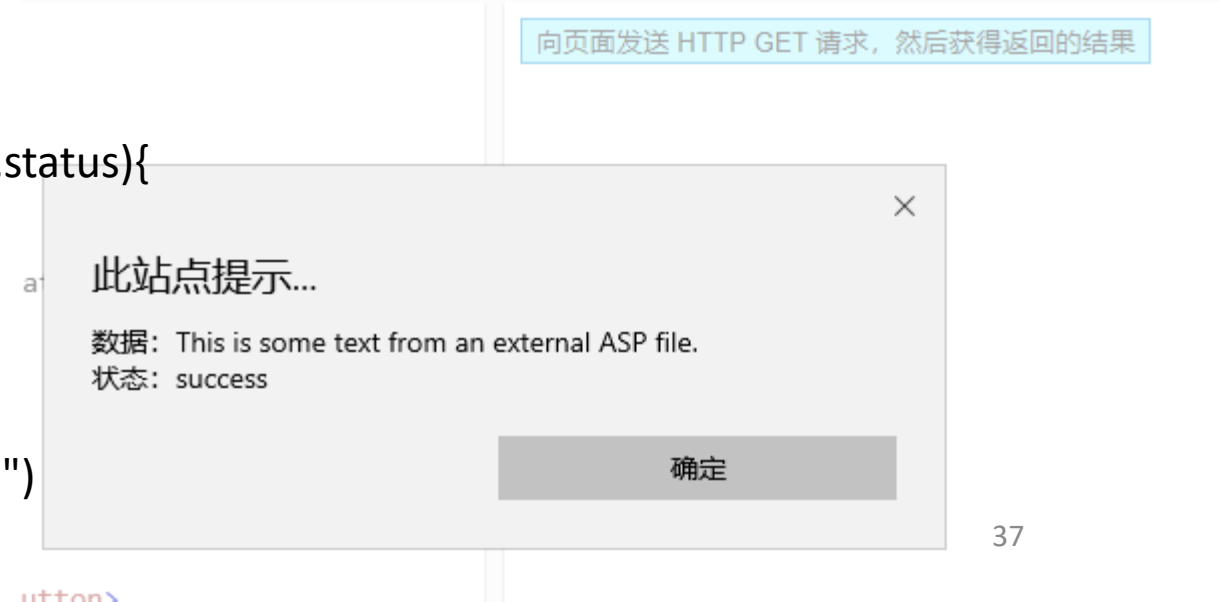
语法：
$.get(URL,callback);
URL **参数规定希望请求的 URL。**
可选的 *callback* 参数是请求成功后所执行的回调函数。第一个回调参数存有被请求页面的内容，第二个回调参数存有请求的状态。

实例：

```
$("button").click(function(){
 $.get("/example/jquery/demo_test.asp",function(data,status){
     alert("数据：" + data + "\n状态：" + status);
   });
```

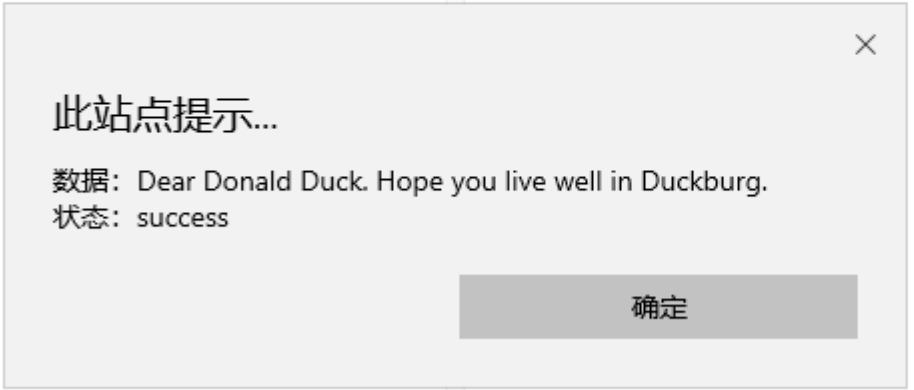
demo_test.asp

```
<%
response.write("This is some text from an external ASP file.")
%>
```



向页面发送 HTTP POST 请求，并获得返回的结果

jQuery $.post() 方法



$.post() 方法通过 HTTP POST 请求从服务器上请求数据

语法：

$.post(URL,data,callback);

必需的 URL 参数规定希望请求的 URL。

可选的 data 参数规定连同请求发送的数据。

可选的 callback 参数是请求成功后所执行的回调函数。第一个回调参数存有被请求页面的内容，而第二个参数存有请求的状态。

实例

```
$("button").click(function(){
  $.post("demo_test_post.asp",
  {
    name:"Donald Duck",
    city:"Duckburg"
  },
  function(data,status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

```
demo_test_post.asp
<%
dim fname,city
fname=Request.Form("name")
city=Request.Form("city")
Response.Write("Dear " & fname & ". ")
Response.Write("Hope you live well in " & city & ".")
%>
```

THANK. YOU