**AJAX 简介**

* [Previous Page](http://www.w3school.com.cn/php/php_xml_simplexml.asp)
* [Next Page](http://www.w3school.com.cn/php/php_ajax_xmlhttprequest.asp)

**AJAX = Asynchronous JavaScript And XML（异步 JavaScript 及 XML）**

AJAX 是 *A*synchronous *J*avaScript *A*nd *X*ML 的首字母缩写。

AJAX 并不是一种新的编程语言，而仅仅是一种新的技术，它可以创建更好、更快且交互性更强的 web 应用程序。

AJAX 使用 JavaScript 在 web 浏览器与 web 服务器之间来发送和接收数据。

通过在幕后与 web 服务器交换数据，而不是每当用户作出改变时重载整个 web 页面，AJAX 技术可以使网页更迅速地响应。

**AJAX 基于开放的标准**

AJAX 基于以下开放的标准：

* *JavaScript*
* *XML*
* *HTML*
* *CSS*

在 AJAX 中使用的开放标准被良好地定义，并得到所有主要浏览器的支持。AJAX 应用程序独立于浏览器和平台。（可以说，它是一种跨平台跨浏览器的技术）。

**AJAX 事关更好的 Internet 应用程序**

与桌面应用程序相比，Web 应用程序有很多优势：

* 可拥有更多用户
* 更容易安装和维护
* 更容易开发

但是， 应用程序不总是象传统应用程序那样强大和友好。

通过 AJAX，可以使 Internet 应用程序更加强大（更轻巧、更快速，且更易使用）。

**今天您就可以开始使用 AJAX**

没有什么新知识需要学习。

AJAX 基于开放的标准。而这些标准已被大多数开发者使用多年。

大多数 web 应用程序可通过使用 AJAX 技术进行重写，来替代传统的 HTML 表单。

**AJAX 使用 XML 和 HTTP 请求**

传统的 web 应用程序会把数据提交到 web 服务器（使用 HTML 表单）。在 web 服务器把数据处理完毕之后，会向用户返回一张完整的新网页。

由于每当用户提交输入，服务器就会返回新网页，传统的 web 应用程序往往运行缓慢，且越来越不友好。

通过 AJAX，web 应用程序无需重载网页，就可以发送并取回数据。完成这项工作，需要通过向服务器发送 HTTP 请求（在幕后），并通过当服务器返回数据时使用 JavaScript 仅仅修改网页的某部分。

一般使用 XML 作为接收服务器数据的格式，尽管可以使用任何格式，包括纯文本。

您将在本教程接下来的章节学习到如何完成这些工作。

**PHP 和 AJAX**

不存在什么 AJAX 服务器。

AJAX 是一种在浏览器运行的技术。它使用浏览器与 web 服务器之间的异步数据传输，使网页从服务器请求少量的信息，而不是整张页面。

AJAX 是一种独立于 web 服务器软件的 web 浏览器技术。

但是，在本教程中，我们将集中在运行在 PHP 服务器上的实际案例，而不是 AJAX 的*工作原理*。

如需阅读更多有关 AJAX 如何工作的知识，请访问我们的 [AJAX 教程](http://www.w3school.com.cn/ajax/index.asp)。

# AJAX XMLHttpRequest

* [Previous Page](http://www.w3school.com.cn/php/php_ajax_intro.asp)
* [Next Page](http://www.w3school.com.cn/php/php_ajax_suggest.asp)

**XMLHttpRequest 对象使 AJAX 成为可能。**

## XMLHttpRequest

XMLHttpRequest 对象是 AJAX 的关键。

该对象在 Internet Explorer 5.5 与 2000 年 7 月发布之后就已经可用了，但是在 2005 人们开始讨论 AJAX 和 Web 2.0 之前，这个对象并没有得到充分的认识。

## 创建 XMLHttpRequest 对象

不同的浏览器使用不同的方法来创建 XMLHttpRequest 对象。

Internet Explorer 使用 ActiveXObject。

其他浏览器使用名为 XMLHttpRequest 的 JavaScript 内建对象。

要克服这个问题，可以使用这段简单的代码：

var XMLHttp=null

if (window.XMLHttpRequest)

{

XMLHttp=new XMLHttpRequest()

}

else if (window.ActiveXObject)

{

XMLHttp=new ActiveXObject("Microsoft.XMLHTTP")

}

### 代码解释：

1. 首先创建一个作为 XMLHttpRequest 对象使用的 XMLHttp 变量。把它的值设置为 null。
2. 然后测试 window.XMLHttpRequest 对象是否可用。在新版本的 Firefox, Mozilla, Opera 以及 Safari 浏览器中，该对象是可用的。
3. 如果可用，则用它创建一个新对象：XMLHttp=new XMLHttpRequest()
4. 如果不可用，则检测 window.ActiveXObject 是否可用。在 Internet Explorer version 5.5 及更高的版本中，该对象是可用的。
5. 如果可用，使用它来创建一个新对象：XMLHttp=new ActiveXObject()

## 改进的例子

一些程序员喜欢使用最新最快的版本的 XMLHttpRequest 对象。

下面的例子试图加载微软最新版本的 "Msxml2.XMLHTTP"，在 Internet Explorer 6 中可用，如果无法加载，则后退到 "Microsoft.XMLHTTP"，在 Internet Explorer 5.5 及其后版本中可用。

function GetXmlHttpObject()

{

var xmlHttp=null;

try

{

// Firefox, Opera 8.0+, Safari

xmlHttp=new XMLHttpRequest();

}

catch (e)

{

// Internet Explorer

try

{

xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");

}

catch (e)

{

xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");

}

}

return xmlHttp;

}

### 代码解释：

1. 首先创建用作 XMLHttpRequest 对象的 XMLHttp 变量。把它的值设置为 null。
2. 按照 web 标准创建对象 (Mozilla, Opera 以及 Safari)：XMLHttp=new XMLHttpRequest()
3. 按照微软的方式创建对象，在 Internet Explorer 6 及更高的版本可用：XMLHttp=new ActiveXObject("Msxml2.XMLHTTP")
4. 如果捕获错误，则尝试更老的方法 (Internet Explorer 5.5) ：XMLHttp=new ActiveXObject("Microsoft.XMLHTTP")

## 更多有关 XMLHttpRequest 对象的信息

如果您希望阅读更多有关 XMLHttpRequest 的内容，请访问我们的 [AJAX 教程](http://www.w3school.com.cn/ajax/index.asp)。

# PHP 和 AJAX 请求

* [Previous Page](http://www.w3school.com.cn/php/php_ajax_xmlhttprequest.asp)
* [Next Page](http://www.w3school.com.cn/php/php_ajax_xml.asp)

## AJAX 请求

在下面的 AJAX 例子中，我们将演示当用户向 web 表单中输入数据时，网页如何与在线的 web 服务器进行通信。

## 在下面的文本框中输入名字：

窗体顶端

First Name：

窗体底端

Suggestions：

这个例子包括三张页面：

* 一个简单的 HTML 表单
* 一段 JavaScript
* 一张 PHP 页面

## HTML 表单

这是 HTML 表单。它包含一个简单的 HTML 表单和指向 JavaScript 的链接：

<html>

<head>

<script src="clienthint.js"></script>

</head>

<body>

<form>

First Name:

<input type="text" id="txt1"

onkeyup="showHint(this.value)">

</form>

<p>Suggestions: <span id="txtHint"></span></p>

</body>

</html>

### 例子解释 - HTML 表单

正如您看到的，上面的 HTML 页面含有一个简单的 HTML 表单，其中带有一个名为 "txt1" 的输入字段。

该表单是这样工作的：

1. 当用户在输入域中按下并松开按键时，会触发一个事件
2. 当该事件被触发时，执行名为 showHint() 的函数
3. 表单的下面是一个名为 "txtHint" 的 <span>。它用作 showHint() 函数所返回数据的占位符。

## JavaScript

JavaScript 代码存储在 "clienthint.js" 文件中，它被链接到 HTML 文档：

var xmlHttp

function showHint(str)

{

if (str.length==0)

{

document.getElementById("txtHint").innerHTML=""

return

}

xmlHttp=GetXmlHttpObject()

if (xmlHttp==null)

{

alert ("Browser does not support HTTP Request")

return

}

var url="gethint.php"

url=url+"?q="+str

url=url+"&sid="+Math.random()

xmlHttp.onreadystatechange=stateChanged

xmlHttp.open("GET",url,true)

xmlHttp.send(null)

}

function stateChanged()

{

if (xmlHttp.readyState==4 || xmlHttp.readyState=="complete")

{

document.getElementById("txtHint").innerHTML=xmlHttp.responseText

}

}

function GetXmlHttpObject()

{

var xmlHttp=null;

try

{

// Firefox, Opera 8.0+, Safari

xmlHttp=new XMLHttpRequest();

}

catch (e)

{

// Internet Explorer

try

{

xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");

}

catch (e)

{

xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");

}

}

return xmlHttp;

}

### 例子解释：

#### showHint() 函数

每当在输入域中输入一个字符，该函数就会被执行一次。

如果文本框中有内容 (str.length > 0)，该函数这样执行：

1. 定义要发送到服务器的 URL（文件名）
2. 把带有输入域内容的参数 (q) 添加到这个 URL
3. 添加一个随机数，以防服务器使用缓存文件
4. 调用 GetXmlHttpObject 函数来创建 XMLHTTP 对象，并在事件被触发时告知该对象执行名为 stateChanged 的函数
5. 用给定的 URL 来打开打开这个 XMLHTTP 对象
6. 向服务器发送 HTTP 请求

如果输入域为空，则函数简单地清空 txtHint 占位符的内容。

#### stateChanged() 函数

每当 XMLHTTP 对象的状态发生改变，则执行该函数。

在状态变成 4 （或 "complete"）时，用响应文本填充 txtHint 占位符 txtHint 的内容。

#### GetXmlHttpObject() 函数

AJAX 应用程序只能运行在完整支持 XML 的 web 浏览器中。

上面的代码调用了名为 GetXmlHttpObject() 的函数。

该函数的作用是解决为不同浏览器创建不同 XMLHTTP 对象的问题。

这一点在上一节中已经解释过了。

## PHP 页面

被 JavaScript 代码调用的服务器页面是一个名为 "gethint.php" 的简单服务器页面。

"gethint.php" 中的代码会检查名字数组，然后向客户端返回对应的名字：

<?php

// Fill up array with names

$a[]="Anna";

$a[]="Brittany";

$a[]="Cinderella";

$a[]="Diana";

$a[]="Eva";

$a[]="Fiona";

$a[]="Gunda";

$a[]="Hege";

$a[]="Inga";

$a[]="Johanna";

$a[]="Kitty";

$a[]="Linda";

$a[]="Nina";

$a[]="Ophelia";

$a[]="Petunia";

$a[]="Amanda";

$a[]="Raquel";

$a[]="Cindy";

$a[]="Doris";

$a[]="Eve";

$a[]="Evita";

$a[]="Sunniva";

$a[]="Tove";

$a[]="Unni";

$a[]="Violet";

$a[]="Liza";

$a[]="Elizabeth";

$a[]="Ellen";

$a[]="Wenche";

$a[]="Vicky";

//get the q parameter from URL

$q=$\_GET["q"];

//lookup all hints from array if length of q>0

if (strlen($q) > 0)

{

$hint="";

for($i=0; $i<count($a); $i++)

{

if (strtolower($q)==strtolower(substr($a[$i],0,strlen($q))))

{

if ($hint=="")

{

$hint=$a[$i];

}

else

{

$hint=$hint." , ".$a[$i];

}

}

}

}

//Set output to "no suggestion" if no hint were found

//or to the correct values

if ($hint == "")

{

$response="no suggestion";

}

else

{

$response=$hint;

}

//output the response

echo $response;

?>

如果存在从 JavaScript 送来的文本 (strlen($q) > 0)，则：

1. 找到与 JavaScript 所传送的字符相匹配的名字
2. 如果找到多个名字，把所有名字包含在 response 字符串中
3. 如果没有找到匹配的名字，把 response 设置为 "no suggestion"
4. 如果找到一个或多个名字，把 response 设置为这些名字
5. 把 response 发送到 "txtHint" 占位符

# PHP 和 AJAX XML 实例

* [Previous Page](http://www.w3school.com.cn/php/php_ajax_suggest.asp)
* [Next Page](http://www.w3school.com.cn/php/php_ajax_database.asp)

**AJAX 可与 XML 文件进行交互式通信。**

## AJAX XML 实例

在下面的 AJAX 实例中，我们将演示网页如何使用 AJAX 技术从 XML 文件中读取信息。

## 在下面的下列列表中选择一个 CD

窗体顶端

选择 CD： 

窗体底端

在此列出 CD 信息。

本例包括三张页面：

* 一个简单 HTML 表单
* 一个 XML 文件
* 一个 JavaScript 文件
* 一张 PHP 页面

## HTML 表单

上面的例子包含了一张简单的 HTML 表单，以及指向 JavaScript 的链接：

<html>

<head>

<script src="selectcd.js"></script>

</head>

<body>

<form>

Select a CD:

<select name="cds" onchange="showCD(this.value)">

<option value="Bob Dylan">Bob Dylan</option>

<option value="Bee Gees">Bee Gees</option>

<option value="Cat Stevens">Cat Stevens</option>

</select>

</form>

<p>

<div id="txtHint"><b>CD info will be listed here.</b></div>

</p>

</body>

</html>

### 例子解释：

正如您看到的，它仅仅是一张简单的 HTML 表单，其中带有名为 "cds" 的下拉列表。

表单下面的段落包含了一个名为 "txtHint" 的 div。这个 div 用作从 web 服务器检索到的数据的占位符。

当用户选择数据时，会执行名为 "showCD" 的函数。这个函数的执行是由 "onchange" 事件触发的。

换句话说，每当用户改变了下拉列表中的值，就会调用 showCD 函数。

## XML 文件

XML 文件是 "[cd\_catalog.xml](http://www.w3school.com.cn/php/cd_catalog.xml)"。该文件中包含了有关 CD 收藏的数据。

## JavaScript

这是存储在 "selectcd.js" 文件中的 JavaScript 代码：

var xmlHttp

function showCD(str)

{

xmlHttp=GetXmlHttpObject()

if (xmlHttp==null)

{

alert ("Browser does not support HTTP Request")

return

}

var url="getcd.php"

url=url+"?q="+str

url=url+"&sid="+Math.random()

xmlHttp.onreadystatechange=stateChanged

xmlHttp.open("GET",url,true)

xmlHttp.send(null)

}

function stateChanged()

{

if (xmlHttp.readyState==4 || xmlHttp.readyState=="complete")

{

document.getElementById("txtHint").innerHTML=xmlHttp.responseText

}

}

function GetXmlHttpObject()

{

var xmlHttp=null;

try

{

// Firefox, Opera 8.0+, Safari

xmlHttp=new XMLHttpRequest();

}

catch (e)

{

// Internet Explorer

try

{

xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");

}

catch (e)

{

xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");

}

}

return xmlHttp;

}

### 例子解释：

stateChanged() 和 GetXmlHttpObject 函数与上一节中的相同，您可以参阅上一页中的相关解释。

#### showCD() 函数

假如选择了下拉列表中的某个项目，则函数执行：

1. 调用 GetXmlHttpObject 函数来创建 XMLHTTP 对象
2. 定义发送到服务器的 URL（文件名）
3. 向 URL 添加带有下拉列表内容的参数 (q)
4. 添加一个随机数，以防服务器使用缓存的文件
5. 当触发事件时调用 stateChanged
6. 通过给定的 URL 打开 XMLHTTP 对象
7. 向服务器发送 HTTP 请求

## PHP 页面

这个被 JavaScript 调用的服务器页面，是一个名为 "getcd.php" 的简单 PHP 文件。

这张页面是用 PHP 编写的，使用 XML DOM 来加载 XML 文档 "[cd\_catalog.xml](http://www.w3school.com.cn/php/cd_catalog.xml)"。

代码运行针对 XML 文件的查询，并以 HTML 返回结果：

<?php

$q=$\_GET["q"];

$xmlDoc = new DOMDocument();

$xmlDoc->load("cd\_catalog.xml");

$x=$xmlDoc->getElementsByTagName('ARTIST');

for ($i=0; $i<=$x->length-1; $i++)

{

//Process only element nodes

if ($x->item($i)->nodeType==1)

{

if ($x->item($i)->childNodes->item(0)->nodeValue == $q)

{

$y=($x->item($i)->parentNode);

}

}

}

$cd=($y->childNodes);

for ($i=0;$i<$cd->length;$i++)

{

//Process only element nodes

if ($cd->item($i)->nodeType==1)

{

echo($cd->item($i)->nodeName);

echo(": ");

echo($cd->item($i)->childNodes->item(0)->nodeValue);

echo("<br />");

}

}

?>

### 例子解释

当请求从 JavaScript 发送到 PHP 页面时，发生：

1. PHP 创建 "cd\_catalog.xml" 文件的 XML DOM 对象
2. 循环所有 "artist" 元素 (nodetypes = 1)，查找与 JavaScript 所传数据向匹配的名字
3. 找到 CD 包含的正确 artist
4. 输出 album 的信息，并发送到 "txtHint" 占位符

* [Previous Page](http://www.w3school.com.cn/php/php_ajax_suggest.asp)
* [Next Page](http://www.w3school.com.cn/php/php_ajax_database.asp)

# PHP 和 AJAX MySQL 数据库实例

* [Previous Page](http://www.w3school.com.cn/php/php_ajax_xml.asp)
* [Next Page](http://www.w3school.com.cn/php/php_ajax_responsexml.asp)

**AJAX 可用来与数据库进行交互式通信。**

## AJAX 数据库实例

在下面的 AJAX 实例中，我们将演示网页如何使用 AJAX 技术从 MySQL 数据库中读取信息。

## 在下拉列表中选择一个名字 （测试说明：该实例功能未实现）

窗体顶端

Select a User: 

窗体底端

在此列出用户信息。

此列由四个元素组成：

* MySQL 数据库
* 简单的 HTML 表单
* JavaScript
* PHP 页面

## 数据库

将在本例中使用的数据库看起来类似这样：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **FirstName** | **LastName** | **Age** | **Hometown** | **Job** |
| 1 | Peter | Griffin | 41 | Quahog | Brewery |
| 2 | Lois | Griffin | 40 | Newport | Piano Teacher |
| 3 | Joseph | Swanson | 39 | Quahog | Police Officer |
| 4 | Glenn | Quagmire | 41 | Quahog | Pilot |

## HTML 表单

上面的例子包含了一个简单的 HTML 表单，以及指向 JavaScript 的链接：

<html>

<head>

<script src="selectuser.js"></script>

</head>

<body>

<form>

Select a User:

<select name="users" onchange="showUser(this.value)">

<option value="1">Peter Griffin</option>

<option value="2">Lois Griffin</option>

<option value="3">Glenn Quagmire</option>

<option value="4">Joseph Swanson</option>

</select>

</form>

<p>

<div id="txtHint"><b>User info will be listed here.</b></div>

</p>

</body>

</html>

### 例子解释 - HTML 表单

正如您看到的，它仅仅是一个简单的 HTML 表单，其中带有名为 "users" 的下拉列表，这个列表包含了姓名，以及与数据库的 "id" 对应的选项值。

表单下面的段落包含了名为 "txtHint" 的 div。这个 div 用作从 web 服务器检索到的信息的占位符。

当用户选择数据时，执行名为 "showUser()" 的函数。该函数的执行由 "onchange" 事件触发。

换句话说：每当用户改变下拉列表中的值，就会调用 showUser() 函数。

## JavaScript

这是存储在 "selectuser.js" 文件中的 JavaScript 代码：

var xmlHttp

function showUser(str)

{

xmlHttp=GetXmlHttpObject()

if (xmlHttp==null)

{

alert ("Browser does not support HTTP Request")

return

}

var url="getuser.php"

url=url+"?q="+str

url=url+"&sid="+Math.random()

xmlHttp.onreadystatechange=stateChanged

xmlHttp.open("GET",url,true)

xmlHttp.send(null)

}

function stateChanged()

{

if (xmlHttp.readyState==4 || xmlHttp.readyState=="complete")

{

document.getElementById("txtHint").innerHTML=xmlHttp.responseText

}

}

function GetXmlHttpObject()

{

var xmlHttp=null;

try

{

// Firefox, Opera 8.0+, Safari

xmlHttp=new XMLHttpRequest();

}

catch (e)

{

//Internet Explorer

try

{

xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");

}

catch (e)

{

xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");

}

}

return xmlHttp;

}

### 例子解释：

stateChanged() 和 GetXmlHttpObject 函数与 [PHP AJAX 请求](http://www.w3school.com.cn/php/php_ajax_suggest.asp) 那一节中的相同，您可以参阅其中的相关解释。

#### showUser() 函数

假如下拉列表中的项目被选择，函数执行：

1. 调用 GetXmlHttpObject 函数来创建 XMLHTTP 对象
2. 定义发送到服务器的 URL（文件名）
3. 向 URL 添加带有下拉列表内容的参数 (q)
4. 添加一个随机数，以防服务器使用缓存的文件
5. 当触发事件时调用 stateChanged
6. 通过给定的 URL 打开 XMLHTTP 对象
7. 向服务器发送 HTTP 请求

## PHP 页面

由 JavaScript 调用的服务器页面，是名为 "getuser.php" 的简单 PHP 文件。

该页面用 PHP 编写，并使用 MySQL 数据库。

其中的代码执行针对数据库的 SQL 查询，并以 HTML 表格返回结果：

<?php

$q=$\_GET["q"];

$con = mysql\_connect('localhost', 'peter', 'abc123');

if (!$con)

{

die('Could not connect: ' . mysql\_error());

}

mysql\_select\_db("ajax\_demo", $con);

$sql="SELECT \* FROM user WHERE id = '".$q."'";

$result = mysql\_query($sql);

echo "<table border='1'>

<tr>

<th>Firstname</th>

<th>Lastname</th>

<th>Age</th>

<th>Hometown</th>

<th>Job</th>

</tr>";

while($row = mysql\_fetch\_array($result))

{

echo "<tr>";

echo "<td>" . $row['FirstName'] . "</td>";

echo "<td>" . $row['LastName'] . "</td>";

echo "<td>" . $row['Age'] . "</td>";

echo "<td>" . $row['Hometown'] . "</td>";

echo "<td>" . $row['Job'] . "</td>";

echo "</tr>";

}

echo "</table>";

mysql\_close($con);

?>

### 例子解释：

当查询从 JavaScript 被发送到这个 PHP 页面，会发生：

1. PHP 打开到达 MySQL 服务器的连接
2. 找到拥有指定姓名的 "user"
3. 创建表格，插入数据，然后将其发送到 "txtHint" 占位符

# PHP 和 AJAX responseXML 实例

* [Previous Page](http://www.w3school.com.cn/php/php_ajax_database.asp)
* [Next Page](http://www.w3school.com.cn/php/php_ajax_livesearch.asp)

**AJAX 可用于以 XML 返回数据库信息。**

## AJAX Database 转 XML 实例 （测试说明：该实例功能未实现）

在下面的 AJAX 实例中，我们将演示网页如何从 MySQL 数据库中读取信息，把数据转换为 XML 文档，并在不同的地方使用这个文档来显示信息。

本例与上一节中的 "PHP AJAX Database" 这个例子很相似，不过有一个很大的不同：在本例中，我们通过使用 responseXML 函数从 PHP 页面得到的是 XML 形式的数据。

把 XML 文档作为响应来接收，使我们有能力更新页面的多个位置，而不仅仅是接收一个 PHP 输出并显示出来。

在本例中，我们将使用从数据库接收到的信息来更新多个 <span> 元素。

## 在下拉列表中选择一个名字

窗体顶端

Select a User: 

窗体底端

### 

此列由四个元素组成：

* MySQL 数据库
* 简单的 HTML 表单
* JavaScript
* PHP 页面

## 数据库

将在本例中使用的数据库看起来类似这样：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **FirstName** | **LastName** | **Age** | **Hometown** | **Job** |
| 1 | Peter | Griffin | 41 | Quahog | Brewery |
| 2 | Lois | Griffin | 40 | Newport | Piano Teacher |
| 3 | Joseph | Swanson | 39 | Quahog | Police Officer |
| 4 | Glenn | Quagmire | 41 | Quahog | Pilot |

## HTML 表单

上面的例子包含了一个简单的 HTML 表单，以及指向 JavaScript 的链接：

<html>

<head>

<script src="responsexml.js"></script>

</head>

<body>

<form>

Select a User:

<select name="users" onchange="showUser(this.value)">

<option value="1">Peter Griffin</option>

<option value="2">Lois Griffin</option>

<option value="3">Glenn Quagmire</option>

<option value="4">Joseph Swanson</option>

</select>

</form>

<h2>

<span id="firstname"></span>&nbsp;<span id="lastname"></span>

</h2>

<span id="job"></span>

<div style="text-align: right">

<span id="age\_text"></span>

<span id="age"></span>

<span id="hometown\_text"></span>

<span id="hometown"></span>

</div>

</body>

</html>

### 例子解释 - HTML 表单

* HTML 表单是一个下拉列表，其 name 属性的值是 "users"，可选项的值与数据库的 id 字段相对应
* 表单下面有几个 <span> 元素，它们用作我们所接收到的不同的值的占位符
* 当用户选择了具体的选项，函数 "showUser()" 就会执行。该函数的执行由 "onchange" 事件触发

换句话说，每当用户在下拉列表中改变了值，函数 showUser() 就会执行，并在指定的 <span> 元素中输出结果。

## JavaScript

这是存储在文件 "responsexml.js" 中的 JavaScript 代码：

var xmlHttp

function showUser(str)

{

xmlHttp=GetXmlHttpObject()

if (xmlHttp==null)

{

alert ("Browser does not support HTTP Request")

return

}

var url="responsexml.php"

url=url+"?q="+str

url=url+"&sid="+Math.random()

xmlHttp.onreadystatechange=stateChanged

xmlHttp.open("GET",url,true)

xmlHttp.send(null)

}

function stateChanged()

{

if (xmlHttp.readyState==4 || xmlHttp.readyState=="complete")

{

xmlDoc=xmlHttp.responseXML;

document.getElementById("firstname").innerHTML=

xmlDoc.getElementsByTagName("firstname")[0].childNodes[0].nodeValue;

document.getElementById("lastname").innerHTML=

xmlDoc.getElementsByTagName("lastname")[0].childNodes[0].nodeValue;

document.getElementById("job").innerHTML=

xmlDoc.getElementsByTagName("job")[0].childNodes[0].nodeValue;

document.getElementById("age\_text").innerHTML="Age: ";

document.getElementById("age").innerHTML=

xmlDoc.getElementsByTagName("age")[0].childNodes[0].nodeValue;

document.getElementById("hometown\_text").innerHTML="<br/>From: ";

document.getElementById("hometown").innerHTML=

xmlDoc.getElementsByTagName("hometown")[0].childNodes[0].nodeValue;

}

}

function GetXmlHttpObject()

{

var objXMLHttp=null

if (window.XMLHttpRequest)

{

objXMLHttp=new XMLHttpRequest()

}

else if (window.ActiveXObject)

{

objXMLHttp=new ActiveXObject("Microsoft.XMLHTTP")

}

return objXMLHttp

}

### 例子解释：

showUser() 与 GetXmlHttpObject 函数与 [PHP 和 AJAX MySQL 数据库实例](http://www.w3school.com.cn/php/php_ajax_database.asp) 这一节中的例子是相同的。您可以参阅其中的相关解释。

#### stateChanged() 函数

如果选择了下拉列表中的项目，该函数执行：

1. 通过使用 responseXML 函数，把 "xmlDoc" 变量定义为一个 XML 文档
2. 从这个 XML 文档中取回数据，把它们放在正确的 "span" 元素中

## PHP 页面

这个由 JavaScript 调用的服务器页面，是一个名为 "responsexml.php" 的简单的 PHP 文件。

该页面由 PHP 编写，并使用 MySQL 数据库。

代码会运行一段针对数据库的 SQL 查询，并以 XML 文档返回结果：

<?php

header('Content-Type: text/xml');

header("Cache-Control: no-cache, must-revalidate");

//A date in the past

header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");

$q=$\_GET["q"];

$con = mysql\_connect('localhost', 'peter', 'abc123');

if (!$con)

{

die('Could not connect: ' . mysql\_error());

}

mysql\_select\_db("ajax\_demo", $con);

$sql="SELECT \* FROM user WHERE id = ".$q."";

$result = mysql\_query($sql);

echo '<?xml version="1.0" encoding="ISO-8859-1"?>

<person>';

while($row = mysql\_fetch\_array($result))

{

echo "<firstname>" . $row['FirstName'] . "</firstname>";

echo "<lastname>" . $row['LastName'] . "</lastname>";

echo "<age>" . $row['Age'] . "</age>";

echo "<hometown>" . $row['Hometown'] . "</hometown>";

echo "<job>" . $row['Job'] . "</job>";

}

echo "</person>";

mysql\_close($con);

?>

### 例子解释：

当查询从 JavaScript 送达 PHP 页面时，会发生：

* PHP 文档的 content-type 被设置为 "text/xml"
* PHP 文档被设置为 "no-cache"，以防止缓存
* 用 HTML 页面送来的数据设置 $q 变量
* PHP 打开与 MySQL 服务器的连接
* 找到带有指定 id 的 "user"
* 以 XML 文档输出数据

# PHP 和 AJAX Live Search

* [Previous Page](http://www.w3school.com.cn/php/php_ajax_responsexml.asp)
* [Next Page](http://www.w3school.com.cn/php/php_ajax_rss_reader.asp)

**AJAX 可为用户提供更友好、交互性更强的搜索体验。**

## AJAX Live Search

在下面的 AJAX 例子中，我们将演示一个实时的搜索。

实时的搜索与传统搜索相比，具有很多优势：

* 当键入数据时，就会显示出匹配的结果
* 当继续键入数据时，对结果进行过滤
* 如果结果太少，删除字符就可以获得更宽的范围

## 在下面的文本框中搜索 W3School 的页面

窗体顶端



窗体底端

本例包括四个元素：

* 简单的 HTML 表单
* JavaScript
* PHP 页面
* XML 文档

在本例中，结果在一个 XML 文档 ([links.xml](http://www.w3school.com.cn/php/links.xml)) 中进行查找。为了让这个例子小而简单，我们只提供 8 个结果。

## HTML 表单

这是 HTML 页面。它包含一个简单的 HTML 表单，针对此表单的 CSS 样式，以及指向 JavaScript 的链接：

<html>

<head>

<script src="livesearch.js"></script>

<style type="text/css">

#livesearch

{

margin:0px;

width:194px;

}

#txt1

{

margin:0px;

}

</style>

</head>

<body>

<form>

<input type="text" id="txt1" size="30"

onkeyup="showResult(this.value)">

<div id="livesearch"></div>

</form>

</body>

</html>

### 例子解释 - HTML 表单

正如你看到的，HTML 页面包含一个简单的 HTML 表单，其中的文本框名为 "txt1"。

表单是这样工作的：

1. 当用户在文本框中按键并松开按键时，会触发一个事件
2. 当事件触发时，会执行名为 showResult() 的函数
3. 表单下面是名为 "livesearch" 的 <div> 元素。它用作 showResult() 所返回数据的占位符

## JavaScript

JavaScript 代码存储在与 HTML 文档连接的 "livesearch.js" 中：

var xmlHttp

function showResult(str)

{

if (str.length==0)

{

document.getElementById("livesearch").

innerHTML="";

document.getElementById("livesearch").

style.border="0px";

return

}

xmlHttp=GetXmlHttpObject()

if (xmlHttp==null)

{

alert ("Browser does not support HTTP Request")

return

}

var url="livesearch.php"

url=url+"?q="+str

url=url+"&sid="+Math.random()

xmlHttp.onreadystatechange=stateChanged

xmlHttp.open("GET",url,true)

xmlHttp.send(null)

}

function stateChanged()

{

if (xmlHttp.readyState==4 || xmlHttp.readyState=="complete")

{

document.getElementById("livesearch").

innerHTML=xmlHttp.responseText;

document.getElementById("livesearch").

style.border="1px solid #A5ACB2";

}

}

function GetXmlHttpObject()

{

var xmlHttp=null;

try

{

// Firefox, Opera 8.0+, Safari

xmlHttp=new XMLHttpRequest();

}

catch (e)

{

// Internet Explorer

try

{

xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");

}

catch (e)

{

xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");

}

}

return xmlHttp;

}

### 例子解释：

GetXmlHttpObject 与 [PHP 和 AJAX 请求](http://www.w3school.com.cn/php/php_ajax_suggest.asp) 中的例子相同。

#### showResult() 函数

该函数每当一个字符输入文本框就会执行一次。

如果文本域中没有输入 (str.length == 0)，该函数把返回字段设置为空，并删除周围的任何边框。

不过，如果文本域中存在输入，则函数执行：

1. 定义发送到服务器的 url （文件名）
2. 把带有输入框内容的参数 (q) 添加到 url
3. 添加一个随机数，以防止服务器使用缓存文件
4. 调用 GetXmlHttpObject 函数来创建 XMLHTTP 对象，并在触发一个变化时告知此函数执行名为 stateChanged 的一个函数
5. 使用给定的 url 来打开 XMLHTTP 对象
6. 向服务器发送 HTTP 请求

#### stateChanged() 函数

每当 XMLHTTP 对象的状态发生变化时，该函数就会执行。

当状态变为 4 (或 "complete") 时，就会使用响应文本来填充 txtHint 占位符的内容，并在返回字段周围设置一个边框。

## PHP 页面

由 JavaScript 代码调用的服务器页面是名为 "livesearch.php" 的 PHP 文件。

"livesearch.php" 中的代码检查那个 XML 文档 "links.xml"。该文档 w3school.com.cn 上的一些页面的标题和 URL。

这些代码会搜索 XML 文件中匹配搜索字符串的标题，并以 HTML 返回结果：

<?php

$xmlDoc = new DOMDocument();

$xmlDoc->load("links.xml");

$x=$xmlDoc->getElementsByTagName('link');

//get the q parameter from URL

$q=$\_GET["q"];

//lookup all links from the xml file if length of q>0

if (strlen($q) > 0)

{

$hint="";

for($i=0; $i<($x->length); $i++)

{

$y=$x->item($i)->getElementsByTagName('title');

$z=$x->item($i)->getElementsByTagName('url');

if ($y->item(0)->nodeType==1)

{

//find a link matching the search text

if (stristr($y->item(0)->childNodes->item(0)->nodeValue,$q))

{

if ($hint=="")

{

$hint="<a href='" .

$z->item(0)->childNodes->item(0)->nodeValue .

"' target='\_blank'>" .

$y->item(0)->childNodes->item(0)->nodeValue . "</a>";

}

else

{

$hint=$hint . "<br /><a href='" .

$z->item(0)->childNodes->item(0)->nodeValue .

"' target='\_blank'>" .

$y->item(0)->childNodes->item(0)->nodeValue . "</a>";

}

}

}

}

}

// Set output to "no suggestion" if no hint were found

// or to the correct values

if ($hint == "")

{

$response="no suggestion";

}

else

{

$response=$hint;

}

//output the response

echo $response;

?>

### 例子解释：

如果从 JavaScript 送来了任何文本 (strlen($q) > 0)，会发生：

1. PHP 创建 "links.xml" 文件的一个 XML DOM 对象
2. 遍历所有 "title" 元素 (nodetypes = 1)，以便找到匹配 JavaScript 所传数据的 name
3. 找到包含正确 title 的 link，并设置为 "$response" 变量。如果找到多于一个匹配，所有的匹配都会添加到变量
4. 如果没有找到匹配，则把 $response 变量设置为 "no suggestion"
5. $result 是送往 "livesearch" 占位符的输出

# PHP 和 AJAX RSS 阅读器

* [Previous Page](http://www.w3school.com.cn/php/php_ajax_livesearch.asp)
* [Next Page](http://www.w3school.com.cn/php/php_ajax_poll.asp)

**RSS 阅读器用于阅读 RSS Feed。**

**RSS 允许对新闻和更新进行快速浏览。**

## AJAX RSS 阅读器

在下面的 AJAX 实例中，我们将演示一个 RSS 阅读器，通过它，来自 RSS 的内容在不进行刷新的情况下载入网页。

## 在下面的列表框中选择一个 RSS 新闻订阅

窗体顶端

Select an RSS-Feed: 

窗体底端

在此列出 RSS Feed。

本例包括三个元素：

* 简单的 HTML 表单
* JavaScript
* PHP 页面

## HTML 表单

这是 HTML 页面。它包含一个简单的 HTML 表单和执行一个 JavaScript 文件的链接：

<html>

<head>

<script type="text/javascript" src="getrss.js"></script>

</head>

<body>

<form>

Select an RSS-Feed:

<select onchange="showRSS(this.value)">

<option value="Google">Google News</option>

<option value="MSNBC">MSNBC News</option>

</select>

</form>

<p><div id="rssOutput">

<b>RSS Feed will be listed here.</b></div></p>

</body>

</html>

### 例子解释 - HTML 表单

正如您看到的，上面的 HTML 页面包含一个简单的 HTML 表单，其中带有一个下拉列表框。

表单是这样工作的：

1. 当用户选择下拉框中的选项时，会触发一个事件
2. 当事件触发时，执行 showRSS() 函数

表单下面是名为 "rssOutput" 的一个 <div>。它用作 showRSS() 函数所返回的数据的占位符。

## JavaScript

JavaScript 代码存储在 "getrss.js" 中，它与 HTML 文档相连接：

var xmlHttp

function showRSS(str)

{

xmlHttp=GetXmlHttpObject()

if (xmlHttp==null)

{

alert ("Browser does not support HTTP Request")

return

}

var url="getrss.php"

url=url+"?q="+str

url=url+"&sid="+Math.random()

xmlHttp.onreadystatechange=stateChanged

xmlHttp.open("GET",url,true)

xmlHttp.send(null)

}

function stateChanged()

{

if (xmlHttp.readyState==4 || xmlHttp.readyState=="complete")

{

document.getElementById("rssOutput")

.innerHTML=xmlHttp.responseText

}

}

function GetXmlHttpObject()

{

var xmlHttp=null;

try

{

// Firefox, Opera 8.0+, Safari

xmlHttp=new XMLHttpRequest();

}

catch (e)

{

// Internet Explorer

try

{

xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");

}

catch (e)

{

xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");

}

}

return xmlHttp;

}

### 例子解释：

stateChanged() 和 GetXmlHttpObject 函数与 [PHP 和 AJAX 请求](http://www.w3school.com.cn/php/php_ajax_suggest.asp) 这一节中的例子相同。

#### showRSS() 函数

每当在下拉框中选择选择时，该函数就会执行：

1. 定义发送到服务器的 url （文件名）
2. 把参数 (q) 添加到 url，参数内容是下拉框中的被选项
3. 添加一个随机数，以防止服务器缓存文件
4. 调用 GetXmlHttpObject 函数来创建 XMLHTTP 对象，并告知该对象在触发一个改变时去执行 stateChanged 函数
5. 通过给定的 url 来打开 XMLHTTP
6. 把 HTTP 请求发动到服务器

## PHP 页面

调用 JavaScript 代码的服务器页面是名为 "getrss.php" 的 PHP 文件：

<?php

//get the q parameter from URL

$q=$\_GET["q"];

//find out which feed was selected

if($q=="Google")

{

$xml=("http://news.google.com/news?ned=us&topic=h&output=rss");

}

elseif($q=="MSNBC")

{

$xml=("http://rss.msnbc.msn.com/id/3032091/device/rss/rss.xml");

}

$xmlDoc = new DOMDocument();

$xmlDoc->load($xml);

//get elements from "<channel>"

$channel=$xmlDoc->getElementsByTagName('channel')->item(0);

$channel\_title = $channel->getElementsByTagName('title')

->item(0)->childNodes->item(0)->nodeValue;

$channel\_link = $channel->getElementsByTagName('link')

->item(0)->childNodes->item(0)->nodeValue;

$channel\_desc = $channel->getElementsByTagName('description')

->item(0)->childNodes->item(0)->nodeValue;

//output elements from "<channel>"

echo("<p><a href='" . $channel\_link

. "'>" . $channel\_title . "</a>");

echo("<br />");

echo($channel\_desc . "</p>");

//get and output "<item>" elements

$x=$xmlDoc->getElementsByTagName('item');

for ($i=0; $i<=2; $i++)

{

$item\_title=$x->item($i)->getElementsByTagName('title')

->item(0)->childNodes->item(0)->nodeValue;

$item\_link=$x->item($i)->getElementsByTagName('link')

->item(0)->childNodes->item(0)->nodeValue;

$item\_desc=$x->item($i)->getElementsByTagName('description')

->item(0)->childNodes->item(0)->nodeValue;

echo ("<p><a href='" . $item\_link

. "'>" . $item\_title . "</a>");

echo ("<br />");

echo ($item\_desc . "</p>");

}

?>

### 例子解释：

当一个选项从 JavaScript 发送时，会发生：

1. PHP 找出哪个 RSS feed 被选中
2. 为选中的 RSS feed 创建 XML DOM 对象
3. 找到并输出来自 RSS 频道的元素
4. 遍历前三个 RSS 项目中的元素，并进行输出

# PHP 和 AJAX 投票

* [Previous Page](http://www.w3school.com.cn/php/php_ajax_rss_reader.asp)
* [Next Page](http://www.w3school.com.cn/php/php_ref_array.asp)

## AJAX 投票

在这个 AJAX 实例中，我们将演示一个投票程序，网页在不重新加载的情况下，就可以获得结果。

## 到目前为止，您喜欢 PHP 和 AJXA 吗？

窗体顶端

Yes:   
No: 

窗体底端

本例包括四个元素：

* HTML 表单
* JavaScript
* PHP 页面
* 存放结果的文本文件

## HTML 表单

这是 HTML 页面。它包含一个简单的 HTML 表单，以及一个与 JavaScript 文件的连接：

<html>

<head>

<script src="poll.js"></script>

</head>

<body>

<div id="poll">

<h2>Do you like PHP and AJAX so far?</h2>

<form>

Yes:

<input type="radio" name="vote"

value="0" onclick="getVote(this.value)">

<br />

No:

<input type="radio" name="vote"

value="1" onclick="getVote(this.value)">

</form>

</div>

</body>

</html>

### 例子解释 - HTML 表单

正如您看到的，上面的 HTML 页面包含一个简单的 HTML 表单，其中的 <div> 元素带有两个单选按钮。

表单这样工作：

* 当用户选择 "yes" 或 "no" 时，会触发一个事件
* 当事件触发时，执行 getVote() 函数
* 围绕该表单的是名为 "poll" 的 <div>。当数据从 getVote() 函数返回时，返回的数据会替代该表单。

## 文本文件

文本文件 (poll\_result.txt) 中存储来自投票程序的数据。

它类似这样：

0||0

第一个数字表示 "Yes" 投票，第二个数字表示 "No" 投票。

注释：记得只允许您的 web 服务器来编辑该文本文件。不要让其他人获得访问权，除了 web 服务器 (PHP)。

## JavaScript

JavaScript 代码存储在 "poll.js" 中，并于 HTML 文档相连接：

var xmlHttp

function getVote(int)

{

xmlHttp=GetXmlHttpObject()

if (xmlHttp==null)

{

alert ("Browser does not support HTTP Request")

return

}

var url="poll\_vote.php"

url=url+"?vote="+int

url=url+"&sid="+Math.random()

xmlHttp.onreadystatechange=stateChanged

xmlHttp.open("GET",url,true)

xmlHttp.send(null)

}

function stateChanged()

{

if (xmlHttp.readyState==4 || xmlHttp.readyState=="complete")

{

document.getElementById("poll").

innerHTML=xmlHttp.responseText;

}

}

function GetXmlHttpObject()

{

var objXMLHttp=null

if (window.XMLHttpRequest)

{

objXMLHttp=new XMLHttpRequest()

}

else if (window.ActiveXObject)

{

objXMLHttp=new ActiveXObject("Microsoft.XMLHTTP")

}

return objXMLHttp

}

### 例子解释：

stateChanged() 和 GetXmlHttpObject 函数与 [PHP 和 AJAX 请求](http://www.w3school.com.cn/php/php_ajax_suggest.asp) 这一节中的例子相同。

#### getVote() 函数

当用户在 HTML 表单中选择 "yes" 或 "no" 时，该函数就会执行。

1. 定义发送到服务器的 url （文件名）
2. 向 url 添加参数 (vote)，参数中带有输入字段的内容
3. 添加一个随机数，以防止服务器使用缓存的文件
4. 调用 GetXmlHttpObject 函数来创建 XMLHTTP 对象，并告知该对象当触发一个变化时执行 stateChanged 函数
5. 用给定的 url 来打开 XMLHTTP 对象
6. 向服务器发送 HTTP 请求

## PHP 页面

由 JavaScript 代码调用的服务器页面是名为 "poll\_vote.php" 的一个简单的 PHP 文件。

<?php

$vote = $\_REQUEST['vote'];

//get content of textfile

$filename = "poll\_result.txt";

$content = file($filename);

//put content in array

$array = explode("||", $content[0]);

$yes = $array[0];

$no = $array[1];

if ($vote == 0)

{

$yes = $yes + 1;

}

if ($vote == 1)

{

$no = $no + 1;

}

//insert votes to txt file

$insertvote = $yes."||".$no;

$fp = fopen($filename,"w");

fputs($fp,$insertvote);

fclose($fp);

?>

<h2>Result:</h2>

<table>

<tr>

<td>Yes:</td>

<td>

<img src="poll.gif"

width='<?php echo(100\*round($yes/($no+$yes),2)); ?>'

height='20'>

<?php echo(100\*round($yes/($no+$yes),2)); ?>%

</td>

</tr>

<tr>

<td>No:</td>

<td>

<img src="poll.gif"

width='<?php echo(100\*round($no/($no+$yes),2)); ?>'

height='20'>

<?php echo(100\*round($no/($no+$yes),2)); ?>%

</td>

</tr>

</table>

### 例子解释：

所选的值从 JavaScript 传来，然后会发生：

1. 获取 "poll\_result.txt" 文件的内容
2. 把文件内容放入变量，并向被选变量累加 1
3. 把结果写入 "poll\_result.txt" 文件
4. 输出图形化的投票结果