AJAX Tutorial

AJAX is about updating parts of a web page, without reloading the whole page.

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML / XHTML
- CSS
- JavaScript / DOM

If you want to study these subjects first, find the tutorials on our Home page.

What is AJAX?

AJAX = Asynchronous JavaScript and XML.

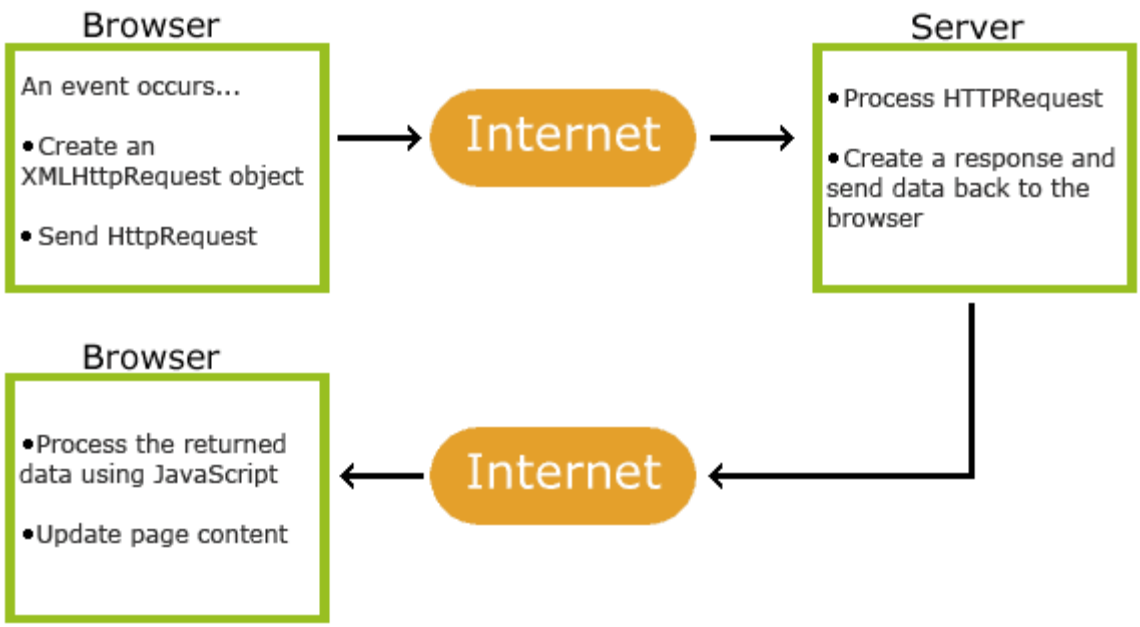
AJAX is a technique for creating fast and dynamic web pages.

AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Classic web pages, (which do not use AJAX) must reload the entire page if the content should change.

Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.

How AJAX Works



AJAX is Based on Internet Standards

AJAX is based on internet standards, and uses a combination of:

- XMLHttpRequest object (to exchange data asynchronously with a server)
- JavaScript/DOM (to display/interact with the information)
- CSS (to style the data)
- XML (often used as the format for transferring data)

AJAX applications are browser- and platform-independent!

AJAX Example Explained

The AJAX application above contains one div section and one button.

The div section will be used to display information returned from a server. The button calls a function named loadXMLDoc(), if it is clicked:

```
<!DOCTYPE html>
<html>
<body>
```

```
<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button>

</body>
</html>
```

Next, add a <script> tag to the page's head section. The script section contains the loadXMLDoc() function:

```
<head>
<script>
function loadXMLDoc()
{
.... AJAX script goes here ...
}
</script>
</head>
```

AJAX - Create an XMLHttpRequest Object

The keystone of AJAX is the XMLHttpRequest object.

The XMLHttpRequest Object

All modern browsers support the XMLHttpRequest object (IE5 and IE6 use an ActiveXObject).

The XMLHttpRequest object is used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Create an XMLHttpRequest Object

All modern browsers (IE7+, Firefox, Chrome, Safari, and Opera) have a built-in XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

variable=new XMLHttpRequest();

Old versions of Internet Explorer (IE5 and IE6) uses an ActiveX Object:

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

To handle all modern browsers, including IE5 and IE6, check if the browser supports the XMLHttpRequest object. If it does, create an XMLHttpRequest object, if not, create an ActiveXObject:

Example

```
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");
  }
```

AJAX - Send a Request To a Server

The XMLHttpRequest object is used to exchange data with a server.

Send a Request To a Server

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

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

Method	Description
open(*method,url,async*)	Specifies the type of request, the URL, and if the request should be handled asynchronously or not. *method*: the type of request: GET or POST *url*: the location of the file on the server

	async: true (asynchronous) or false (synchronous)
send(*string*)	Sends the request off to the server. *string*: Only used for POST requests

---

GET or POST?

GET is simpler and faster than POST, and can be used in most cases.

However, always use POST requests when:

- A cached file is not an option (update a file or database on the server)
- Sending a large amount of data to the server (POST has no size limitations)
- Sending user input (which can contain unknown characters), POST is more robust and secure than GET

GET Requests

A simple GET request:

Example

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


<!DOCTYPE html>

<html>

<head>

<script>

function loadXMLDoc()

{

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");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","demo_get.asp",true);
xmlhttp.send();
}
</script>
</head>
<body>

<h2>AJAX</h2>
<button type="button" onclick="loadXMLDoc()">Request data</button>
<div id="myDiv"></div>
```

```
</body>

</html>
```

In the example above, you may get a cached result.

To avoid this, add a unique ID to the URL:

Example

```
xmlhttp.open("GET","demo_get.asp?t=" + Math.random(),true);
xmlhttp.send();


<!DOCTYPE html>

<html>

<head>

<script>

function loadXMLDoc()

{

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");

  }

xmlhttp.onreadystatechange=function()

  {

  if (xmlhttp.readyState==4 && xmlhttp.status==200)
```

```
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","demo_get.asp?t=" + Math.random(),true);
xmlhttp.send();
}
</script>
</head>
<body>

<h2>AJAX</h2>
<button type="button" onclick="loadXMLDoc()">Request data</button>
<p>Click the button several times to see if the time changes, or if the file is cached.</p>
<div id="myDiv"></div>

</body>
</html>
```

If you want to send information with the GET method, add the information to the URL:

Example

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

<!DOCTYPE html>
<html>
<head>
<script>
```

```
function loadXMLDoc()
{
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");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","demo_get2.asp?fname=Henry&lname=Ford",true);
xmlhttp.send();
}
</script>
</head>
<body>
```

```
<h2>AJAX</h2>

<button type="button" onclick="loadXMLDoc()">Request data</button>

<div id="myDiv"></div>


</body>

</html>
```

POST Requests

A simple POST request:

Example

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

```
<!DOCTYPE html>

<html>

<head>

<script>

function loadXMLDoc()

{

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");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("POST","demo_post.asp",true);
xmlhttp.send();
}
</script>
</head>
<body>

<h2>AJAX</h2>
<button type="button" onclick="loadXMLDoc()">Request data</button>
<div id="myDiv"></div>

</body>
</html>
```

To POST data like an HTML form, add an HTTP header with setRequestHeader(). Specify the data you want to send in the send() method:

Example

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

Method	Description
setRequestHeader(*header,value*)	Adds HTTP headers to the request. *header*: specifies the header name *value*: specifies the header value

---

The url - A File On a Server

The url parameter of the open() method, is an address to a file on a server:

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

The file can be any kind of file, like .txt and .xml, or server scripting files like .asp and .php (which can perform actions on the server before sending the response back).

Asynchronous - True or False?

AJAX stands for Asynchronous JavaScript and XML, and for the XMLHttpRequest object to behave as AJAX, the async parameter of the open() method has to be set to true:

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

Sending asynchronous requests is a huge improvement for web developers. Many of the tasks performed on the server are very time consuming. Before AJAX, this operation could cause the application to hang or stop.

With AJAX, the JavaScript does not have to wait for the server response, but can instead:

- execute other scripts while waiting for server response
- deal with the response when the response ready

Async=true

When using async=true, specify a function to execute when the response is ready in the onreadystatechange event:

Example

```
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
```

```
<!DOCTYPE html>

<html>

<head>

<script>

function loadXMLDoc()

{

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");
```

```
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
}
</script>
</head>
<body>

<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button>

</body>
</html>
```

You will learn more about onreadystatechange in a later chapter.

Async=false

To use async=false, change the third parameter in the open() method to false:

```
xmlhttp.open("GET","ajax_info.txt",false);
```

Using async=false is not recommended, but for a few small requests this can be ok.

Remember that the JavaScript will NOT continue to execute, until the server response is ready. If the server is busy or slow, the application will hang or stop.

Note: When you use async=false, do NOT write an onreadystatechange function - just put the code after the send() statement:

Example

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

<!DOCTYPE html>

<html>

<head>

<script>

function loadXMLDoc()

{

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");

  }

xmlhttp.open("GET","ajax_info.txt",false);

xmlhttp.send();

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

```
}

</script>

</head>

<body>


<div id="myDiv"><h2>Let AJAX change this text</h2></div>

<button type="button" onclick="loadXMLDoc()">Change Content</button>


</body>

</html>
```

AJAX - Server Response

Server Response

To get the response from a server, use the responseText or responseXML property of the XMLHttpRequest object.

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

The responseText Property

If the response from the server is not XML, use the responseText property.

The responseText property returns the response as a string, and you can use it accordingly:

Example

```
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

```
<!DOCTYPE html>
<html>
<head>
<script>
function loadXMLDoc()
{
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");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
}
```

```
</script>

</head>

<body>


<div id="myDiv"><h2>Let AJAX change this text</h2></div>

<button type="button" onclick="loadXMLDoc()">Change Content</button>


</body>

</html>
```

The responseXML Property

If the response from the server is XML, and you want to parse it as an XML object, use the responseXML property:

Example

Request the file cd_catalog.xml and parse the response:

```
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;
```

EXPLAIN

```
<!DOCTYPE html>

<html>

<head>

<script>
```

```
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");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    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;
    }
```

```
  }
xmlhttp.open("GET","cd_catalog.xml",true);
xmlhttp.send();
}
</script>
</head>

<body>

<h2>My CD Collection:</h2>
<div id="myDiv"></div>
<button type="button" onclick="loadXMLDoc()">Get my CD collection</button>

</body>
</html>
```

AJAX - The onreadystatechange Event

The onreadystatechange event

When a request to a server is sent, we want to perform some actions based on the response.

The onreadystatechange event is triggered every time the readyState changes.

The readyState property holds the status of the XMLHttpRequest.

Three important properties of the XMLHttpRequest object:

Property	Description

onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 404: Page not found

In the onreadystatechange event, we specify what will happen when the server response is ready to be processed.

When readyState is 4 and status is 200, the response is ready:

Example

```
<!DOCTYPE html>

<html>

<head>

<script>

function loadXMLDoc()

{

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");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
}
</script>
</head>
<body>

<div id="myDiv">><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button>

</body>
</html>
```

Note: The onreadystatechange event is triggered five times (0-4), one time for each change in readyState.

Using a Callback Function

A callback function is a function passed as a parameter to another function.

If you have more than one AJAX task on your website, you should create ONE standard function for creating the XMLHttpRequest object, and call this for each AJAX task.

The function call should contain the URL and what to do on onreadystatechange (which is probably different for each call):

Example

```
<!DOCTYPE html>

<html>

<head>

<script>

var xmlhttp;

function loadXMLDoc(url,cfunc)

{

if (window.XMLHttpRequest)

  {// code for IE7+, Firefox, Chrome, Opera, Safari

  xmlhttp=new XMLHttpRequest();

  }

else

  {// code for IE6, IE5

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

  }

xmlhttp.onreadystatechange=cfunc;

xmlhttp.open("GET",url,true);

xmlhttp.send();
```

```
}
function myFunction()
{
loadXMLDoc("ajax_info.txt",function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  });
}
</script>
</head>
<body>

<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="myFunction()">Change Content</button>

</body>
</html>
```

AJAX PHP Example

AJAX is used to create more interactive applications.

AJAX PHP Example

The following example will demonstrate how a web page can communicate with a web server while a user type characters in an input field:

Example

Start typing a name in the input field below:

First name:

Suggestions:

```
<!DOCTYPE html>

<html>

<head>

<script>

function showHint(str)

{

var xmlhttp;

if (str.length==0)

  {

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

  return;

  }

if (window.XMLHttpRequest)

  {// code for IE7+, Firefox, Chrome, Opera, Safari
```

```
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","gethint.asp?q="+str,true);
xmlhttp.send();
}
</script>
</head>
<body>

<h3>Start typing a name in the input field below:</h3>
<form action=" gethint.php ">
First name: <input type="text" id="txt1" onkeyup="showHint(this.value)" />
</form>
<p>Suggestions: <span id="txtHint"></span></p>
```

```
</body>

</html>
```

Example Explained - The showHint() Function

When a user types a character in the input field above, the function "showHint()" is executed. The function is triggered by the "onkeyup" event:

```
function showHint(str)
{
var xmlhttp;
if (str.length==0)
  {
  document.getElementById("txtHint").innerHTML="";
  return;
  }
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","gethint.asp?q="+str,true);
xmlhttp.send();
}
```

Source code explanation:

If the input field is empty (str.length==0), the function clears the content of the txtHint placeholder and exits the function.

If the input field is not empty, the showHint() function executes the following:

- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server
- Notice that a parameter (q) is added to the URL (with the content of the input field)

The PHP File

Below is the code above rewritten in PHP.

Note: To run the example in PHP, change the value of the url variable (in the HTML file) "gethint.php".

```
<?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;
?>
```

PHP - AJAX and MySQL

AJAX can be used for interactive communication with a database.

AJAX Database Example

The following example will demonstrate how a web page can fetch information from a database with AJAX:

Example

Person info will be listed here...

Example Explained - The MySQL Database

The database table we use in the example above looks like this:

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

Example Explained - The HTML Page

When a user selects a user in the dropdown list above, a function called "showUser()" is executed. The function is triggered by the "onchange" event:

```
<html>
<head>
```

```
<script>
function showUser(str)
{
if (str==="")
  {
  document.getElementById("txtHint").innerHTML="";
  return;
  }
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","getuser.php?q="+str,true);
xmlhttp.send();
}
</script>
</head>
<body>

<form>
<select name="users" onchange="showUser(this.value)">
<option value="">Select a person:</option>
<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>
<br>
<div id="txtHint"><b>Person info will be listed here.</b></div>
```

```
</body>
</html>
```

The showUser() function does the following:

- Check if a person is selected
- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server
- Notice that a parameter (q) is added to the URL (with the content of the dropdown list)

The PHP File

The page on the server called by the JavaScript above is a PHP file called "getuser.php".

The source code in "getuser.php" runs a query against a MySQL database, and returns the result in an HTML table:

```
<?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);
?>
```

Explanation: When the query is sent from the JavaScript to the PHP file, the following happens:

1. PHP opens a connection to a MySQL server
2. The correct person is found
3. An HTML table is created, filled with data, and sent back to the "txtHint" placeholder

PHP Example - AJAX Live Search

« Previous

Next Chapter »

AJAX can be used to create more user-friendly and interactive searches.

AJAX Live Search

The following example will demonstrate a live search, where you get search results while you type.

Live search has many benefits compared to traditional searching:

- Results are shown as you type

- Results narrow as you continue typing
- If results become too narrow, remove characters to see a broader result

Search for a W3Schools page in the input field below:

no suggestion

The results in the example above are found in an XML file (links.xml). To make this example small and simple, only six results are available.

Example Explained - The HTML Page

When a user types a character in the input field above, the function "showResult()" is executed. The function is triggered by the "onkeyup" event:

```
<html>
<head>
<script>
function showResult(str)
{
if (str.length==0)
  {
  document.getElementById("livesearch").innerHTML="";
  document.getElementById("livesearch").style.border="0px";
  return;
  }
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("livesearch").innerHTML=xmlhttp.responseText;
```

```
  document.getElementById("livesearch").style.border="1px solid #A5ACB2";
  }
 }
xmlhttp.open("GET","livesearch.php?q="+str,true);
xmlhttp.send();
}
</script>
</head>
<body>

<form>
<input type="text" size="30" onkeyup="showResult(this.value)">
<div id="livesearch"></div>
</form>

</body>
</html>
```

Source code explanation:

If the input field is empty (str.length==0), the function clears the content of the livesearch placeholder and exits the function.

If the input field is not empty, the showResult() function executes the following:

- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server
- Notice that a parameter (q) is added to the URL (with the content of the input field)

The PHP File

The page on the server called by the JavaScript above is a PHP file called "livesearch.php".

The source code in "livesearch.php" searches an XML file for titles matching the search string and returns the result:

```
<?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;
?>
```

If there is any text sent from the JavaScript (strlen($q) > 0), the following happens:

- Load an XML file into a new XML DOM object
- Loop through all <title> elements to find matches from the text sent from the JavaScript
- Sets the correct url and title in the "$response" variable. If more than one match is found, all matches are added to the variable
- If no matches are found, the $response variable is set to "no suggestion"

PHP Example - AJAX RSS Reader

An RSS Reader is used to read RSS Feeds.

AJAX RSS Reader

The following example will demonstrate an RSS reader, where the RSS-feed is loaded into a webpage without reloading:

Example Explained - The HTML Page

When a user selects an RSS-feed in the dropdown list above, a function called "showResult()" is executed. The function is triggered by the "onchange" event:

```
<html>
<head>
<script>
function showRSS(str)
{
if (str.length==0)
  {
```

```
  document.getElementById("rssOutput").innerHTML="";
  return;
  }
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("rssOutput").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","getrss.php?q="+str,true);
xmlhttp.send();
}
</script>
</head>
<body>

<form>
<select onchange="showRSS(this.value)">
<option value="">Select an RSS-feed:</option>
<option value="Google">Google News</option>
<option value="MSNBC">MSNBC News</option>
</select>
</form>
<br>
<div id="rssOutput">RSS-feed will be listed here...</div>
</body>
</html>
```

The showResult() function does the following:

- Check if an RSS-feed is selected
- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready

- Send the request off to a file on the server
- Notice that a parameter (q) is added to the URL (with the content of the dropdown list)

The PHP File

The page on the server called by the JavaScript above is a PHP file called "getrss.php":

```
<?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>");
 }
?>
```

When a request for an RSS feed is sent from the JavaScript, the following happens:

- Check which feed was selected
- Create a new XML DOM object
- Load the RSS document in the xml variable
- Extract and output elements from the channel element
- Extract and output elements from the item elements

PHP Example - AJAX Poll

AJAX Poll

The following example will demonstrate a poll where the result is shown without reloading.

Do you like PHP and AJAX so far?

Yes:

No:

Example Explained - The HTML Page

When a user choose an option above, a function called "getVote()" is executed. The function is triggered by the "onclick" event:

```
<html>
<head>
<script>
function getVote(int)
{
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("poll").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","poll_vote.php?vote="+int,true);
xmlhttp.send();
}
</script>
</head>
<body>

<div id="poll">
<h3>Do you like PHP and AJAX so far?</h3>
<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>
```

The getVote() function does the following:

- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server
- Notice that a parameter (vote) is added to the URL (with the value of the yes or no option)

The PHP File

The page on the server called by the JavaScript above is a PHP file called "poll_vote.php":

```
<?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>
```

The value is sent from the JavaScript, and the following happens:

1. Get the content of the "poll_result.txt" file
2. Put the content of the file in variables and add one to the selected variable
3. Write the result to the "poll_result.txt" file
4. Output a graphical representation of the poll result

The Text File

The text file (poll_result.txt) is where we store the data from the poll.

It is stored like this:

```
0||0
```

The first number represents the "Yes" votes, the second number represents the "No" votes.

Note: Remember to allow your web server to edit the text file. Do **NOT** give everyone access, just the web server (PHP).