http://www.w3schools.com/ajax/ajax_intro.asp

AJAX Introduction

« Previous

Next Chapter »

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
- JavaScript

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 misleading name. You don't have to understand XML to use AJAX.

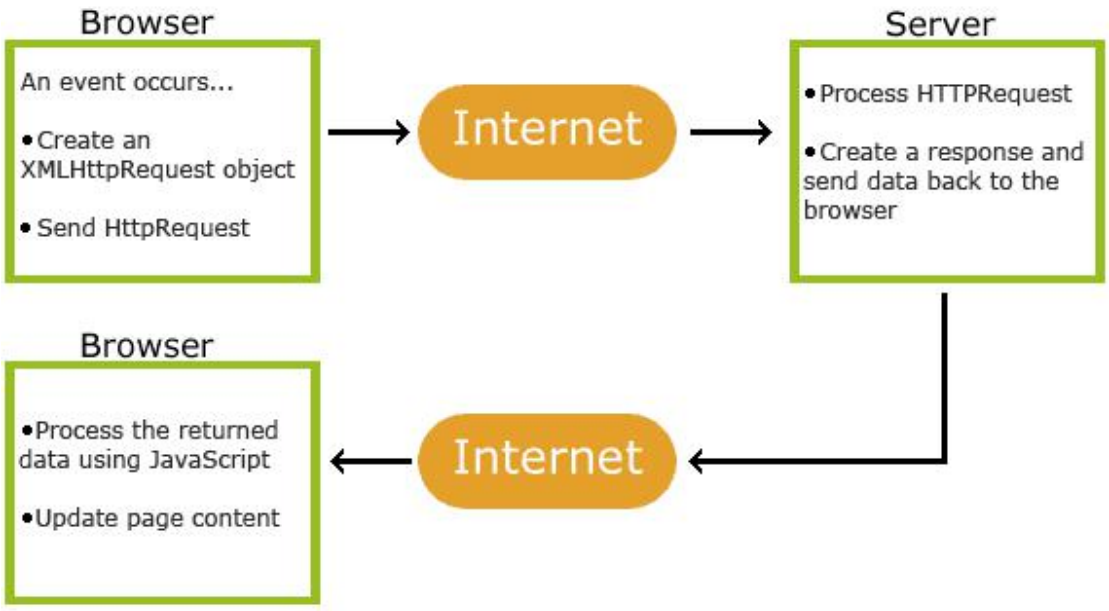
AJAX is a technique for creating fast and dynamic web pages.

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.

How AJAX Works



AJAX is Based on Internet Standards

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

- XMLHttpRequest object (to retrieve data from a web server)
- JavaScript/DOM (to display/use the data)

XMLHttpRequest is a misleading name. You don't have to understand XML to use AJAX.

Google Suggest

AJAX was made popular in 2005 by Google, with Google Suggest.

Google Suggest is using AJAX to create a very dynamic web interface: When you start typing in Google's search box, a JavaScript sends the letters off to a server and the server returns a list of suggestions.

Start Using AJAX Today

AJAX is based on existing standards. These standards have been used by developers for several years. Read our next chapters to see how it works!

AJAX - Create an XMLHttpRequest Object

« Previous

Next Chapter »

The keystone of AJAX is the XMLHttpRequest object.

The XMLHttpRequest Object

All modern browsers support the XMLHttpRequest object.

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 (Chrome, IE7+, Firefox, 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) use an ActiveX Object:

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

To handle all 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 xhttp;
if (window.XMLHttpRequest) {
    xhttp = new XMLHttpRequest();
    } else {
    // code for IE6, IE5
    xhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

Try it Yourself »

In the next chapter you will learn about sending server requests.

AJAX - Send a Request To a Server

« Previous

Next Chapter »

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:

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

Method	Description
open(*method, url, async*)	Specifies the type of request *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(*string*)	Sends the request to the server (used for POST)

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

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

Try it Yourself »

In the example above, you may get a cached result. To avoid this, add a unique ID to the URL:

Example

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

Try it Yourself »

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

Example

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

Try it Yourself »

POST Requests

A simple POST request:

Example

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

Try it Yourself »

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

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

Try it Yourself »

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:

```
xhttp.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:

```
xhttp.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

```
xhttp.onreadystatechange = function() {
  if (xhttp.readyState == 4 && xhttp.status == 200) {
    document.getElementById("demo").innerHTML =
```

```
xhttp.responseText;
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Try it Yourself »

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:

```
xhttp.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

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

Try it Yourself »

AJAX - Server Response

« Previous

Next Chapter »

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("demo").innerHTML = xhttp.responseText;
```

Try it Yourself »

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

Try it Yourself »

cd_catalog.xml

```
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
```

```
  </CD>
  <CD>
    <TITLE>Greatest Hits</TITLE>
    <ARTIST>Dolly Parton</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>RCA</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1982</YEAR>
  </CD>
  <CD>
    <TITLE>Still got the blues</TITLE>
    <ARTIST>Gary Moore</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>Virgin records</COMPANY>
    <PRICE>10.20</PRICE>
    <YEAR>1990</YEAR>
  </CD>
</CATALOG>
```

AJAX -

The onreadystatechange Event

« Previous

Next Chapter »

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 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

```
function loadDoc() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (xhttp.readyState == 4 && xhttp.status == 200) {
            document.getElementById("demo").innerHTML =
xhttp.responseText;
        }
```

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

Try it Yourself »

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

```
function loadDoc(url, cfunc) {
    var xhttp;
    xhttp=new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (xhttp.readyState == 4 && xhttp.status == 200) {
            cfunc(xhttp);
        }
    };
xhttp.open("GET", url, true);
xhttp.send();
}
```

Try it Yourself »

AJAX PHP Example

« Previous

Next Chapter »

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:

Example Explained

In the example above, when a user types a character in the input field, a function called "showHint()" is executed.

The function is triggered by the onkeyup event.

Here is the HTML code:

Example

```
<html>
<head>
<script>
function showHint(str) {
    if (str.length == 0) {
        document.getElementById("txtHint").innerHTML = "";
        return;
    } else {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
            if (xmlhttp.readyState == 4 && xmlhttp.status == 200)
{
                document.getElementById("txtHint").innerHTML =
xmlhttp.responseText;
            }
        };
        xmlhttp.open("GET", "gethint.php?q=" + str, true);
        xmlhttp.send();
    }
}
</script>
</head>
<body>

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

Try it Yourself »

Code explanation:

First, check if the input field is empty (str.length == 0). If it is, clear the content of the txtHint placeholder and exit the function.

However, if the input field is not empty, do the following:

- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a PHP file (gethint.php) on the server

- Notice that q parameter is added gethint.php?q="+str
- The str variable holds the content of the input field

The PHP File - "gethint.php"

The PHP file checks an array of names, and returns the corresponding name(s) to the browser:

```
<?php
// 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 = $_REQUEST["q"];

$hint = "";

// lookup all hints from array if $q is different from ""
if ($q !== "") {
    $q = strtolower($q);
    $len=strlen($q);
    foreach($a as $name) {
        if (stristr($q, substr($name, 0, $len))) {
            if ($hint === "") {
                $hint = $name;
            } else {
                $hint .= ", $name";
            }
        }
    }
}

// Output "no suggestion" if no hint was found or output correct
values
echo $hint === "" ? "no suggestion" : $hint;
?>
```

AJAX Database Example

« Previous

Next Chapter »

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

Customer info will be listed here...

Try it Yourself »

Example Explained - The showCustomer() Function

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

showCustomer

```
function showCustomer(str) {
  var xhttp;
  if (str == "") {
    document.getElementById("txtHint").innerHTML = "";
    return;
  }
  xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
    document.getElementById("txtHint").innerHTML =
xhttp.responseText;
```

```
    }
  };
  xhttp.open("GET", "getcustomer.asp?q="+str, true);
  xhttp.send();
}
```

The showCustomer() function does the following:

- Check if a customer 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 AJAX Server Page

The page on the server called by the JavaScript above is an ASP file called "getcustomer.asp".

The server file could easily be rewritten in PHP, or some other server languages.

Look at a corresponding example in PHP.

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

```
<%
response.expires=-1
sql="SELECT * FROM CUSTOMERS WHERE CUSTOMERID="
sql=sql & "'" & request.querystring("q") & "'"

set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/datafolder/northwind.mdb"))
set rs=Server.CreateObject("ADODB.recordset")
rs.Open sql,conn

response.write("<table>")
```

```
do until rs.EOF
  for each x in rs.Fields
    response.write("<tr><td><b>" & x.name & "</b></td>")
    response.write("<td>" & x.value & "</td></tr>")
  next
  rs.MoveNext
loop
response.write("</table>")
%>
```

AJAX XML Example

« Previous

Next Chapter »

AJAX can be used for interactive communication with an XML file.

AJAX XML Example

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

Example

Get CD info

Try it Yourself »

Example Explained

When a user clicks on the "Get CD info" button above, the loadDoc() function is executed.

The loadDoc() function creates an XMLHttpRequest object, adds the function to be executed when the server response is ready, and sends the request off to the server.

When the server response is ready, an HTML table is built, nodes (elements) are extracted from the XML file, and it finally updates the txtCDInfo placeholder with the HTML table filled with XML data:

LoadXMLDoc()

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
    myFunction(xhttp);
    }
  };
  xhttp.open("GET", "cd_catalog.xml", true);
  xhttp.send();
}
function myFunction(xml) {
  var i;
  var xmlDoc = xml.responseXML;
  var table="<tr><th>Artist</th><th>Title</th></tr>";
  var x = xmlDoc.getElementsByTagName("CD");
  for (i = 0; i <x.length; i++) {
    table += "<tr><td>" +
    x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeVa
lue +
    "</td><td>" +
    x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeVal
ue +
    "</td></tr>";
  }
  document.getElementById("demo").innerHTML = table;
}
```

The XML File

The XML file used in the example above looks like this: "cd_catalog.xml".