Ajax Tutorial
(Asynchronous Javascript And XML)
Creating client-side dynamic web pages

Brief History

Ajax is only a name given to a set of tools that were previously existing.

The main part is XMLHttpRequest, a server-side object usable in JavaScript , that was implemented into Internet Explorer since the 4.0 version.

It is in Internet Explorer an ActiveX object that was first named XMLHTTP some times, before to be generalized on all browser under the name XMLHttpRequest, when the Ajax technologie becomes commonly used.

The use of XMLHttpRequest in 2005 by Google, in Gmail and GoogleMaps has contributed to the success of this format. But this is the name Ajax itself that made the technology so popular.

Why to use Ajax?

Mainly to build a fast, dynamic website, but also to save resources.

For improving sharing of resources, it is better to use the power of all the client computers rather than just an unique server and network. Ajax allows to perform processing on client computer (in JavaScript) with data taken from the server.

The processing of web page formerly was only server-side, using web services or PHP scripts, before the whole page was sent within the network.

But Ajax can selectively modify a part of a page displayed by the browser, and update it without the need to reload the whole document with all images, menus, etc...

For example, fields of forms, choices of user, may be processed and the result displayed immediately into the same page.

What is Ajax in depth?

Ajax is a set of technologies, supported by a web browser, including these elements:

- HTML and CSS for presenting.

- JavaScript (ECMAScript) for local processing, and DOM (Document Object Model) to access data inside the page or to access elements of XML file read on the server (with the getElementByTagName method for example)

- The XMLHttpRequest class read or send data on the server asynchronously.

optionally...

- The DomParser class may be used
- PHP or another scripting language may be used on the server.

Ajax Tutorial
(Asynchronous Javascript And XML)
Creating client-side dynamic web pages

- XML and XSLT to process the data if returned in XML form.
- SOAP may be used to dialog with the server.

The "Asynchronous" word, means that the response of the server while be processed when available, without to wait and to freeze the display of the page.

How does it works?

Ajax uses a programming model with display and events. These events are user actions, they call functions associated to elements of the web page.

Interactivity is achieved with forms and buttons. DOM allows to link elements of the page with actions and also to extract data from XML files provided by the server.

To get data on the server, XMLHttpRequest provides two methods:

- open: create a connection.
- send: send a request to the server.

Data furnished by the server will be found in the attributes of the XMLHttpRequest object:

- **responseXml** for an XML file or
- **responseText** for a plain text.

Take note that a new XMLHttpRequest object has to be created for each new file to load.

We have to wait for the data to be available to process it, and in this purpose, the state of availability of data is given by the **readyState** attribute of XMLHttpRequest.

States of **readyState** follow (only the last one is really useful):

0: not initialized.
1: connection established.
2: request received.
3: answer in process.
4: finished.

Ajax and DHTML

DHTML has same purpose and is also, as Ajax, a set of standards:

- HTML
- CSS
- JavaScript.

DHTML allows to change the display of the page from user commands or from text typed by the user. Ajax allows also to send requests asynchronously and load data from the server.

The XMLHttpRequest class

Allows to interact with the servers, thanks to its methods and attributes.

Attributes

readyState	the code successively changes value from 0 to 4 that means for "ready".
status	200 is OK 404 if the page is not found.
responseText	holds loaded data as a string of characters.
responseXml	holds an XML loaded file, DOM's method allows to extract data.
onreadystatechange	property that takes a function as value that is invoked when the readystatechange event is dispatched.

Methods

open(mode, url, boolean)	mode: type of request, GET or POST url: the location of the file, with a path. boolean: true (asynchronous) / false (synchronous). optionally, a login and a password may be added to arguments.
send("string")	null for a GET command.

Building a request, step by step

First step: create an instance

This is just a classical instance of class, but two options must be tried, for browser compatibility.

```
if (window.XMLHttpRequest)     // Object of the current windows
{
   xhr = new XMLHttpRequest();     // Firefox, Safari, ...
}
else
 if (window.ActiveXObject)   // ActiveX version
 {
   xhr = new ActiveXObject("Microsoft.XMLHTTP"); // Internet Explorer
 }
or exceptions may be used instead:
try {
  xhr = new ActiveXObject("Microsoft.XMLHTTP");    // Trying Internet Explorer
}
catch(e)    // Failed
{
  xhr = new XMLHttpRequest();    // Other browsers.
}
```

Second step: wait for the response

The response and further processing are included in a function and the return of the function will be assigned to the **onreadystatechange** attribute of the object previously created.
xhr.onreadystatechange = function() { // instructions to process the response };
if (xhr.readyState == 4)
{
  // Received, OK
} else
{
  // Wait...
}

Third step: make the request itself

Two methods of XMLHttpRequest are used:
- **open**: command GET or POST, URL of the document, true for asynchronous.
- **send**: with POST only, the data to send to the server.
The request below read a document on the server.
xhr.open('GET', 'http://www.xul.fr/somefile.xml', true);
xhr.send(null);

Examples

Get a text

```
<html>
<head>
<script>
function submitForm()
{
    var xhr;
    try {  xhr = new ActiveXObject('Msxml2.XMLHTTP');  }
    catch (e)
    {
       try {   xhr = new ActiveXObject('Microsoft.XMLHTTP');   }
       catch (e2)
       {
        try {  xhr = new XMLHttpRequest();    }
        catch (e3) {  xhr = false;   }
       }
    }

    xhr.onreadystatechange  = function()
    {
        if(xhr.readyState  == 4)
```

```
      {
         if(xhr.status  == 200)
            document.ajax.dyn="Received:"  + xhr.responseText;
         else
            document.ajax.dyn="Error code " + xhr.status;
      }
   };

   xhr.open(GET, "data.txt",  true);
   xhr.send(null);
}
</script>
</head>

<body>
   <FORM method="POST" name="ajax" action="">
      <INPUT type="BUTTON" value="Submit"  ONCLICK="submitForm()">
      <INPUT type="text" name="dyn"  value="">
   </FORM>
 </body>
 </html>
```

Syntax of form using Ajax

View a demo of the example in action.
Download the source.

Comments on the code:

new ActiveXObject(Microsoft.XMLHTTP)
This constructor is for Internet Explorer.

new XMLHttpRequest()
This constructor is for any other browser including Firefox.

http.onreadystatechange
An anonymous function is assigned to the event indicator.

http.readyState == 4
The 4 state means for the response is ready and sent by the server.

http.status == 200
This status means ok, otherwise some error code is returned, 404 for example.

http.open("POST", "data.xml", true);
POST or GET
URL of the script to execute.
true for asynchronous (false for synchronous).

http.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
This is for POST only.

http.send(document.getElementById("TYPEDTEXT").value);
Send data to the server. Data comes from the "TYPEDTEXT" variable filled throught the form by the user.

Get from XML

To get data from an XML file we have just to replace this line:

```
document.ajax.dyn="Received:" + xhr.responseText;
by this code:
var doc = xhr.responseXML;   // Assign the XML file to a var
var element = doc.getElementsByTagName('root').item(0);   // Read the first element
document.ajax.dyn.value= element.firstChild.data;    // Assign the content to the form
```
View a demo of the get from XML example in action.

Write to body

In this demo, the text read is put into the body of the page, and not into a textfield. The code below replaces the textfield form object and the second part replaces the assignment into the JavaScript function.

```
<div id="zone">
   ... some text to replace ...
 </div>
document.getElementById("zone").innerHTML = "Received:" + xhr.responseText;
```

Post a text

In this demo, a text is sent to the server and is written into a file. The call to the "open" method changes, the argument is POST, the url is the name of a file or script that receives the data sent, and that must process it. And the "send" method has now a value as argument that is a string of parameters.

```
xhr.open("POST", "ajax-post-text.php", true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.send(data);
```

The parameter of the send method is in format of the HTML POST method. When several values are sent, they are separated by the ampersand symbol:

```
var data = "file=" + url + "&content=" + content;
```

The "file" parameter is the name of a file created to store the content. The filename must be checked by the server to prevent any other file to be modified.

Ajax Tutorial
(Asynchronous Javascript And XML)
Creating client-side dynamic web pages

Using an external file

It is simpler to include a JavaScript file. This line will be included into the head section of the HTML page:

<script src="ajax.js" type="text/javascript"></script>
And the function is called with this statement:

var xhr = createXHR();

View the script in the ajax.js file.

How to build an Ajax website?

You need for some wrapper. A short list of frameworks is provided below.

Your JavaScript program, integrated into a web page, sends request to the server to load files for rebuilding of pages. The received documents are processed with Dom's methods or XML parsers and the data are used to update the pages.

Drawbacks of Ajax

- If JavaScript is not activated, Ajax can't works. The user must be asked to set JavaScript from within options of the browser, with the "noscript" tag.

- Since data to display are loaded dynamically, they are not part of the page, and the keywords inside are not used by search engines.

- The asynchronous mode may change the page with delays (when the processing on the server take some times), this may be disturbing.

- The back button may be deactivated (this is not the case in examples provided here). This may be overcomed.

Specifications

Ajax is based on these specifications:

- XML 1, HTML 4, DOM 2, CSS 2 from W3C
- ECMAScript 1.5. Standard for JavaScript from ECMA.
- W3C draft specification for XMLHttpRequest.
- HTTP 1.1. Status codes: 404 etc

Articles and references

- JavaScript. The programming language of Ajax.
- Directory. Sites on the open directory about Ajax.
- JSON. The second data format for Ajax.
- Ajax offline. Ajax and Gears, tutorial and demo.
- Ajax, a New Approach to Web applications. The article that coined the term.

Tools

- Firebug. Firefox extension to debug Ajax.
- Anaa. A simple framework based on this tutorial.
- Ajax Toolkit Framework. Toolbox to build Ajax IDEs.
- Eclipse Webtool. Platform for web applications.