# Asynchronous JavaScript and XML (AJAX)

* AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS and Java Script.
* Ajax uses XHTML for content and CSS for presentation, as well as the Document Object Model and JavaScript for dynamic content display.
* Conventional web application trasmit information to and from the sever using synchronous requests. This means you fill out a form, hit submit, and get directed to a new page with new information from the server.
* With AJAX when submit is pressed, JavaScript will make a request to the server, interpret the results and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
* XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
* AJAX is a web browser technology independent of web server software.
* A user can continue to use the application while the client program requests information from the server in the background
* Intuitive and natural user interaction. No clicking required only Mouse movement is a sufficient event trigger.
* Data-driven as opposed to page-driven

**Rich Internet Application (RIA) Technology**

AJAX is most viable RIA technology so far. Its getting tremendous industry momentum and several toolkit and framworks are emerging. But same time JAX has browser incompatibility and it is supported by Java Script which is hard to maintain and debug.

**AJAX Is Based On Open Standards**

AJAX is based on the following open standards:

* Browser-based presentation using HTML and Cascading Style Sheets (CSS)
* Data stored in XML format and fetched from the server
* Behind-the-scenes data fetches using XMLHttpRequest objects in the browser
* JavaScript to make everything happen

# Technologies Used in AJAX

## JavaScript

* Loosely typed scripting language
* JavaScript function is called when an event in a page occurs
* Glue for the whole AJAX operation

## DOM

* API for accessing and manipulating structured documents
* Represents the structure of XML and HTML documents

## CSS

* Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript

## XMLHttpRequest

* JavaScript object that performs asynchrous interaction with the server

# AJAX Browser Support

All the available browsers cannot support AJAX. Here is the list of major browsers which support AJAX.

* Mozilla Firefox 1.0 and above
* Netscape version 7.1 and above
* Apple Safari 1.2 and above.
* Microsoft Internet Exporer 5 and above
* Konqueror
* Opera 7.6 and above

So now when you write your application then you would have to take care of the browsers who do not support AJAX.

**NOTE:** When we are saying that browser does not support AJAX it simply means that browser does not support creation of Javascript object XMLHttpRequest object.

# AJAX in Action

This section will give you clear picture of the exact steps of AJAX operation.

**Steps of AJAX Operation**

1. A client event occurs
2. An XMLHttpRequest object is created
3. The XMLHttpRequest object is configured
4. The XMLHttpRequest object makes an asynchronous request to the Webserver.
5. Webserver returns the result containing XML document.
6. The XMLHttpRequest object calls the callback() function and processes the result.
7. The HTML DOM is updated

Let’s take these steps one by one

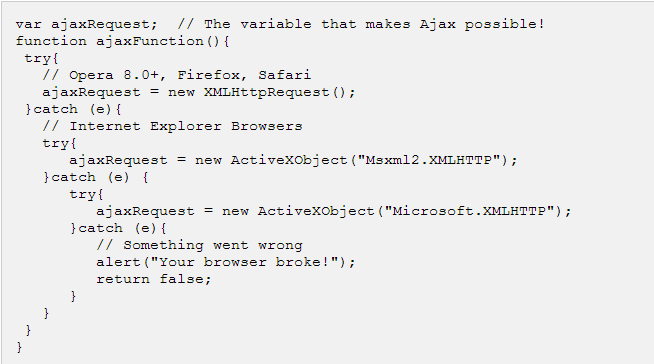
**1. A client event occurs**

A JavaScript function is called as the result of an event

Example: *validateUserId()* JavaScript function is mapped as a event handler to a *onkeyup*event on input form field whose id is set to *"userid"*

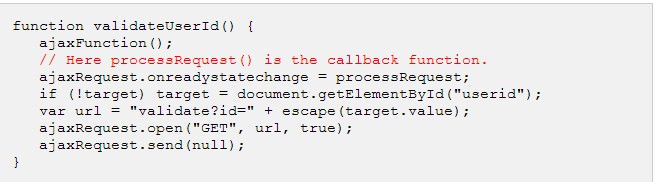
<input type="text" size="20" id="userid" name="id" onkeyup="validateUserId();">

**2. The XMLHttpRequest object is created**



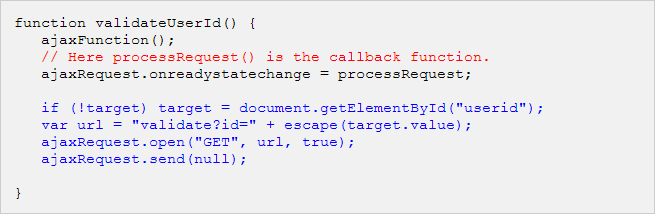
## 3. The XMLHttpRequest object is Configured

In this step we will write a function which will be triggered by the client event and a callback function processRequest() will be registered



## 4. Making Asynchornous Request to the Webserver

Source code is available in the above piece of code. Code written in blue color is responsible to make a request to the web server. This is all being done using XMLHttpRequest object*ajaxRequest*



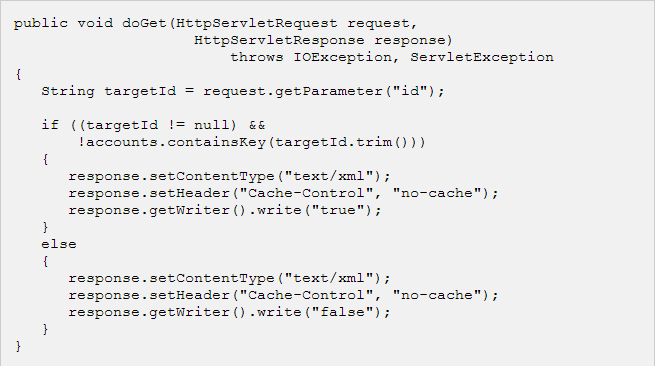
Assume if you enter *mohammad* in userid box then in the above request URL is set to validate?id=mohammad

## 5. Webserver returns the result containing XML document

You can implement your server side script in any language. But logic should be as follows

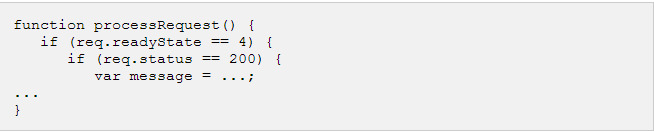
* Get a request from the client
* Parse the input from the client
* Do required processing.
* Send the output to the client.

If we assume that you are going to write a servlet then here is the piece of code



## 6. Callback function processRequest() is called

The XMLHttpRequest object was configured to call the processRequest() function when there is a state change to the *readyState* of the *XMLHttpRequest* object. Now this function will receive the result from the server and will do required processing. As in the following example it sets a variable message on true or false based on returned value from the Webserver.

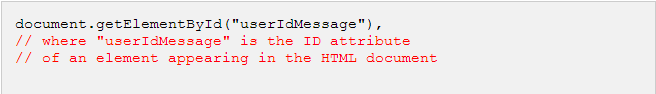


**7. The HTML DOM is updated**

This is the final step and in this step your HTML page will be updated. It happens in the following way

JavaScript technology gets a reference to any element in a page using DOM API

* The recommended way to gain a reference to an element is to call.



* JavaScript technology may now be used to modify the element's attributes; modify the element's style properties; or add, remove, or modify child elements. Here is the example



thats it...if you understood above mentioned seven steps then you are almost done with AJAX. In next chapter we will see *XMLHttpRequest* object in more detail.

# What is XMLHttpRequest

The XMLHttpRequest object is the key to AJAX. It has been available ever since Internet Explorer 5.5 was released in July 2000, but not fully discovered before people started to talk about AJAX and Web 2.0 in 2005.

XMLHttpRequest (XHR) is an API that can be used by JavaScript, JScript, VBScript and other web browser scripting languages to transfer and manipulate XML data to and from a web server using HTTP, establishing an independent connection channel between a web page's Client-Side and Server-Side.

The data returned from XMLHttpRequest calls will often be provided by back-end databases. Besides XML, XMLHttpRequest can be used to fetch data in other formats, e.g. JSON or even plain text.

You already have seen couple of examples on how to create a XMLHttpRequest object.

Below is listed some of the methods and properties you have to become familiar with.

XMLHttpRequest Methods:

* **abort()**  
  Cancels the current request.
* **getAllResponseHeaders()**  
  Returns the complete set of HTTP headers as a string.
* **getResponseHeader( headerName )**  
  Returns the value of the specified HTTP header.
* **open( method, URL )  
  open( method, URL, async )  
  open( method, URL, async, userName )  
  open( method, URL, async, userName, password )**Specifies the method, URL, and other optional attributes of a request.  
    
  The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods, such as "PUT" and "DELETE" (primarily used in REST applications), may be possible  
    
  The "async" parameter specifies whether the request should be handled asynchronously or not. "True" means that script processing carries on after the send() method, without waiting for a response, and "false" means that the script waits for a response before continuing script processing.
* **send( content )**  
  Sends the request.
* **setRequestHeader( label, value )**  
  Adds a label/value pair to the HTTP header to be sent.

XMLHttpRequest Properties:

* **onreadystatechange**  
  An event handler for an event that fires at every state change.
* **readyState**

The readyState property defines the current state of the XMLHttpRequest object.

Here are the possible values for the readyState propery:

|  |  |
| --- | --- |
| **State** | **Description** |
| 0 | The request is not initialized |
| 1 | The request has been set up |
| 2 | The request has been sent |
| 3 | The request is in process |
| 4 | The request is completed |

**readyState=0** after you have created the XMLHttpRequest object, but before you have called the open() method.

**readyState=1** after you have called the open() method, but before you have called send().

**readyState=2** after you have called send().

**readyState=3** after the browser has established a communication with the server, but before the server has completed the response.

**readyState=4** after the request has been completed, and the response data have been completely received from the server.

* **responseText**  
  Returns the response as a string.
* **responseXML**  
  Returns the response as XML. This property returns an XML document object, which can be examined and parsed using W3C DOM node tree methods and properties.
* **status**  
  Returns the status as a number (e.g. 404 for "Not Found" and 200 for "OK").
* **statusText**  
  Returns the status as a string (e.g. "Not Found" or "OK").

# AJAX Security

## Ajax Security: Server Side

* AJAX-based Web applications use the same serverside security schemes of regular Web applications
* You specify authentication, authorization, and data protection requirements in your web.xml file (declarative) or in your program (programatic)
* AJAX-based Web applications are subject to the same security threats as regular Web applications

## Ajax Security: Client Side

* JavaScript code is visible to a user/hacker. Hacker can use the JavaScript code for inferring server side weaknesses
* JavaScript code is downloaded from the server and executed ("eval") at the client and can compromise the client by mal-intended code
* Downloaded JavaScript code is constrained by sand-box security model and can be relaxed for signed JavaScript

# Current Issues with AJAX

AJAX is growing very fast and that is the reason that it contains many issues with it. We hope with the passes of time they will be resolved ab AJAX will be ideal for web applications. We are listing down few issues which AJAX has as a challenge.

Complexity is increased:

* Server side developers will need to understand that presentation logic will be required in the HTML client pages as well as in the server-side logic
* Page developers must have JavaScript technology skills

AJAX-based applications can be difficult to debug, test, and maintain:

* JavaScript is hard to test - automatic testing is hard
* Weak modularity in JavaScript
* Lack of design patterns or best practice guidelines yet

**Toolkits/Frameworks are not mature yet**

* Most of them are in beta phase

**No standardization of the XMLHttpRequest yet**

* Future version of IE will address this

**No support of XMLHttpRequest in old browsers**

* Iframe will help

**JavaScript technology dependency & incompatibility**

* Must be enabled for applications to function
* Still some browser incompatibilities

**JavaScript code is visible to a hacker**

* Poorly designed JavaScript code can invite security problem