XMLHttpRequest

From Wikipedia, the free encyclopedia

XMLHttpRequest (**XHR**) is an API available in web browser scripting languages such as JavaScript. It is used to send HTTP or HTTPS requests directly to a web server and load the server response data directly back into the script.[1] The data might be received from the server as XML text[2] or as plain text.[3] Data from the response can be used directly to alter the DOM of the currently active document in the browser window without loading a new web page document. The response data can also be evaluated by the client-side scripting. For example, if it was formatted as JSON by the web server, it can easily be converted into a client-side data object for further use.

XMLHttpRequest has an important role in the Ajax web development technique. It is currently used by many websites to implement responsive and dynamic web applications. Examples of these web applications include Gmail, Google Maps, Facebook, and many others.

History and support [edit]

The concept behind the *XMLHttpRequest* object was originally created by the developers of Outlook Web Access for Microsoft Exchange Server 2000.[4] An interface called *IXMLHTTPRequest* was developed and implemented into the second version of the MSXML library using this concept.[4][5] The second version of the MSXML library was shipped with Internet Explorer 5.0 in March 1999, allowing access, via ActiveX, to the *IXMLHTTPRequest* interface using the ***XMLHTTP*** wrapper of the MSXML library.[6]

The Mozilla Foundation developed and implemented an interface called *nsIXMLHttpRequest* into the Gecko layout engine. This interface was modelled to work as closely to Microsoft's *IXMLHTTPRequest* interface as possible.[7][8] Mozilla created a wrapper to use this interface through a JavaScript object which they called ***XMLHttpRequest***.[9] The *XMLHttpRequest* object was accessible as early as Gecko version 0.6 released on December 6 of 2000,[10][11] but it was not completely functional until as late as version 1.0 of Gecko released on June 5, 2002.[10][11] The *XMLHttpRequest* object became a de facto standard amongst other major user agents, implemented in Safari 1.2 released in February 2004,[12] Konqueror, Opera 8.0 released in April 2005,[13] and iCab 3.0b352 released in September 2005.[14]

The World Wide Web Consortium published a *Working Draft* specification for the *XMLHttpRequest* object on April 5, 2006, edited by Anne van Kesteren of Opera Software and Dean Jackson of W3C.[15] Its goal is "to document a minimum set of interoperable features based on existing implementations, allowing Web developers to use these features without platform-specific code." The last revision to the XMLHttpRequest object specification was on November 19 of 2009, being a last call working draft.[16] [17]

Microsoft added the *XMLHttpRequest* object identifier to its scripting languages in Internet Explorer 7.0 released in October 2006.[6]

With the advent of cross-browser JavaScript libraries such as jQuery and the Prototype JavaScript Framework, developers can invoke XMLHttpRequest functionality without coding directly to the API. Prototype provides an asynchronous requester object called `Ajax.Request` that wraps the browser's underlying implementation and provides access to it.[18] jQuery objects represent or wrap elements from the current client-side DOM. They all have a `.load()` method that takes a URI parameter and makes an XMLHttpRequest to that URI, then by default places any returned HTML into the HTML element represented by the jQuery object.[19][20]

The W3C has since published another *Working Draft* specification for the *XMLHttpRequest* object, "XMLHttpRequest Level 2", on February 25 of 2008.[21] Level 2 consists of extended functionality to the ***XMLHttpRequest*** object, including, but not currently limited to, progress events, support for cross-site requests, and the handling of byte streams. The latest revision of the XMLHttpRequest Level 2 specification is that of 20th August 2009, which is still a working draft.[22]

Support in Internet Explorer versions 5, 5.5 and 6 [edit]

Internet Explorer versions 5 and 6 did not define the XMLHttpRequest object identifier in their scripting languages as the XMLHttpRequest identifier itself was not standard at the time of their releases.[6] Backward compatibility can be achieved through object detection if the XMLHttpRequest identifier does not exist.

An example of how to instantiate an XMLHttpRequest object with support for Internet Explorer

versions 5 and 6 using JScript method ActiveXObject is below.[23]

```
/*
   Provide the XMLHttpRequest constructor for IE 5.x-6.x:
   Other browsers (including IE 7.x-8.x) do not redefine
   XMLHttpRequest if it already exists.

   This example is based on findings at:
   http://blogs.msdn.com/xmlteam/archive/2006/10/23/using-the-right-version-of-msxml-in-internet-explorer.aspx
*/
if (typeof XMLHttpRequest == "undefined")
  XMLHttpRequest = function () {
    try { return new ActiveXObject("Msxml2.XMLHTTP.6.0"); }
      catch (e) {}
    try { return new ActiveXObject("Msxml2.XMLHTTP.3.0"); }
      catch (e) {}
    try { return new ActiveXObject("Msxml2.XMLHTTP"); }
      catch (e) {}
    //Microsoft.XMLHTTP points to Msxml2.XMLHTTP.3.0 and is redundant
    throw new Error("This browser does not support XMLHttpRequest.");
  };
```

Web pages that use XMLHttpRequest or XMLHTTP can mitigate the current minor differences in the implementations either by encapsulating the XMLHttpRequest object in a JavaScript wrapper, or by using an existing framework that does so. In either case, the wrapper should detect the abilities of current implementation and work within its requirements.

HTTP request [edit]

The following sections demonstrate how a request using the XMLHttpRequest object functions within a conforming user agent based on the W3C Working Draft. As the W3C standard for the XMLHttpRequest object is still a draft, user agents may not abide by all the functionings of the W3C definition and any of the following is subject to change. Extreme care should be taken into consideration when scripting with the XMLHttpRequest object across multiple user agents. This article will try to list the inconsistencies between the major user agents.

The *open* method [edit]

The HTTP and HTTPS requests of the XMLHttpRequest object must be initialized through the ***open*** method. This method must be invoked prior to the actual sending of a request to validate and resolve the request method, URL, and URI user information to be used for the request. This method does not assure that the URL exists or the user information is correct. This method can accept up to five parameters, but requires only two, to initialize a request.

The first parameter of the method is a text string indicating the HTTP request method to use. The request methods that must be supported by a conforming user agent, defined by the W3C draft for the XMLHttpRequest object, are currently listed as the following.[24]

- GET (Supported by IE7+, Mozilla 1+)
- POST (Supported by IE7+, Mozilla 1+)
- HEAD (Supported by IE7+)
- PUT
- DELETE
- OPTIONS (Supported by IE7+)

However, request methods are not limited to the ones listed above. The W3C draft states that a browser may support additional request methods at their own discretion.

The second parameter of the method is another text string, this one indicating the URL of the HTTP request. The W3C recommends that browsers should raise an error and not allow the request of a URL with either a different port or *ihost* URI component from the current document.[25]

The third parameter, a boolean value indicating whether or not the request will be asynchronous, is not a required parameter by the W3C draft. The default value of this parameter should be assumed to be true by a W3C conforming user agent if it is not provided. An asynchronous request ("true") will not wait on a server response before continuing on with the execution of the current script. It will instead invoke the ***onreadystatechange*** event listener of the XMLHttpRequest object throughout the various stages of the request. A synchronous request ("false") however will block execution of the current script until the request has been completed, thus not invoking the ***onreadystatechange*** event listener.

The fourth and fifth parameters are the URI user and password, respectively. These parameters are not required and should default to the current user and password of the document if not supplied, as defined by the W3C draft.

The *setRequestHeader* method [edit]

Upon successful initialization of a request, the ***setRequestHeader*** method of the XMLHttpRequest object can be invoked to send HTTP headers with the request. The first parameter of this method is the text string name of the header. The second parameter is the text string value. This method must be invoked for each header that needs to be sent with the request. Any headers attached here will be removed the next time the *open* method is invoked in a W3C conforming user agent.

The *send* method [edit]

To send an HTTP request, the ***send*** method of the XMLHttpRequest must be invoked. This method accepts a single parameter containing the content to be sent with the request. This parameter may be omitted if no content needs to be sent. The W3C draft states that this parameter may be any type available to the scripting language as long as it can be turned into a text string, with the exception of the DOM ***document*** object. If a user agent cannot stringify the parameter, then the parameter

should be ignored.

If the parameter is a DOM *document* object, a user agent should assure the document is turned into well-formed XML using the encoding indicated by the *inputEncoding* property of the *document* object. If the **Content-Type** request header was not added through *setRequestHeader* yet, it should automatically be added by a conforming user agent as "application/xml;charset=*charset*," where *charset* is the encoding used to encode the document.

The *onreadystatechange* event listener [edit]

If the **open** method of the XMLHttpRequest object was invoked with the third parameter set to *true* for an asynchronous request, the **onreadystatechange** event listener will be automatically invoked for each of the following actions that change the **readyState** property of the XMLHttpRequest object.

- After the **open** method has been invoked successfully, the **readyState** property of the XMLHttpRequest object should be assigned a value of 1.
- After the **send** method has been invoked and the HTTP response headers have been received, the **readyState** property of the XMLHttpRequest object should be assigned a value of 2.
- Once the HTTP response content begins to load, the **readyState** property of the XMLHttpRequest object should be assigned a value of 3.
- Once the HTTP response content has finished loading, the **readyState** property of the XMLHttpRequest object should be assigned a value of 4.

The major user agents are inconsistent with the handling of the *onreadystatechange* event listener.

The HTTP response [edit]

After a successful and completed call to the **send** method of the XMLHttpRequest, if the server response was valid XML and the ***Content-Type*** header sent by the server is understood by the user agent as an Internet media type for XML, the ***responseXML*** property of the XMLHttpRequest object will contain a DOM document object. Another property, ***responseText*** will contain the response of the server in plain text by a conforming user agent, regardless of whether or not it was understood as XML.

See also [edit]

- Hypertext Transfer Protocol
- Representational State Transfer
- Ajax

References [edit]

1. ^ "XMLHttpRequest object explained by the W3C Working Draft". W3.org. Retrieved 2009-07-14.
2. ^ "The responseXML attribute of the XMLHttpRequest object explained by the W3C Working Draft". W3.org. Retrieved 2009-07-14.
3. ^ "The responseText attribute of the XMLHttpRequest object explained by the W3C Working Draft". W3.org. Retrieved 2009-07-14.
4. ^ *a b* "Article on the history of XMLHTTP by an original developer". Alexhopmann.com. 2007-01-31. Retrieved 2009-07-14.
5. ^ "Specification of the IXMLHTTPRequest interface from the Microsoft Developer Network". Msdn.microsoft.com. Retrieved 2009-07-14.
6. ^ *a b c* Dutta, Sunava (2006-01-23). "Native XMLHTTPRequest object". *IEBlog*. Microsoft. Retrieved 2006-11-30.
7. ^ "Specification of the nsIXMLHttpRequest interface from the Mozilla Developer Center". Developer.mozilla.org. 2008-05-16. Retrieved 2009-07-14.
8. ^ "Specification of the nsIJSXMLHttpRequest interface from the Mozilla Developer Center". Developer.mozilla.org. 2009-05-03. Retrieved 2009-07-14.
9. ^ "Specification of the XMLHttpRequest object from the Mozilla Developer Center". Developer.mozilla.org. 2009-05-03. Retrieved 2009-07-14.
10. ^ *a b* "Version history for the Mozilla Application Suite". Mozilla.org. Retrieved 2009-07-14.
11. ^ *a b* "Downloadable, archived releases for the Mozilla browser". Archive.mozilla.org. Retrieved 2009-07-14.
12. ^ "Archived news from Mozillazine stating the release date of Safari 1.2". Weblogs.mozillazine.org. Retrieved 2009-07-14.
13. ^ "Press release stating the release date of Opera 8.0 from the Opera website". Opera.com. 2005-04-19. Retrieved 2009-07-14.
14. ^ Soft-Info.org. "Detailed browser information stating the release date of iCab 3.0b352 from". Soft-Info.com. Retrieved 2009-07-14.
15. ^ "Specification of the XMLHttpRequest object from the Level 1 W3C Working Draft released on April 5th, 2006". W3.org. Retrieved 2009-07-14.
16. ^ "XMLHttpRequest W3C Working Draft 19 November 2009". W3.org. Retrieved 2009-12-17.
17. ^ "W3C Process Document, Section 7.4.2 Last Call Announcement". W3.org. Retrieved 2009-12-17.
18. ^ Porteneuve, Christophe (2007). "9". in Daniel H Steinberg. Raleigh, North Carolina: Pragmatic Bookshelf. pp. 183. ISBN 1-934356-01-8.
19. ^ Chaffer, Jonathan; Karl Swedberg (2007). *Learning jQuery*. Birmingham: Packt Publishing. pp. 107. ISBN 978-1-847192-50-9.
20. ^ Chaffer, Jonathan; Karl Swedberg (2007). *jQuery Reference Guide*. Birmingham: Packt Publishing. pp. 156. ISBN 978-1-847193-81-0.
21. ^ "Specification of the XMLHttpRequest object from the Level 2 W3C Working Draft released on February 25th, 2008". W3.org. Retrieved 2009-07-14.
22. ^ "XMLHttpRequest Level 2, W3C Working Draft 20 August 2009". W3.org. Retrieved 2010-04-08.
23. ^ "Ajax Reference (XMLHttpRequest object)". JavaScript Kit. 2008-07-22. Retrieved 2009-07-14.
24. ^ "Dependencies of the XMLHttpRequest object explained by the W3C Working Draft". W3.org. Retrieved 2009-07-14.
25. ^ "The "open" method of the XMLHttpRequest object explained by the W3C Working Draft". W3.org. Retrieved 2009-10-13.

External links [edit]

- Level 1 specification of the XMLHttpRequest object from W3C
- Level 2 specification of the XMLHttpRequest object from W3C
- Specification of the XMLHttpRequest object for Apple developers
- Specification of the XMLHttpRequest object for Microsoft developers
- Specification of the XMLHttpRequest object for Mozilla developers
- Specification of the XMLHttpRequest object for Opera developers
- "Attacking AJAX Applications", a presentation given at the Black Hat security conference. Discusses several issues involving XHR and the future of cross-domain AJAX.

Wikibooks has a book on the topic of *XMLHttpRequest*

v · d · e	**Windows Internet Explorer**	[show]
v · d · e	**World Wide Web Consortium**	[show]

Categories: JavaScript | XML | HTTP | HTTP clients | Ajax (programming) | Articles with example code | Microsoft