Ajax (programming)

From Wikipedia, the free encyclopedia

Ajax (also **AJAX**; /ˈeɪdʒæks/; short for **asynchronous JavaScript and XML**)[1][2][3] is a group of interrelated Web development techniques used on the client-side to create asynchronous Web applications. With Ajax, web applications can send data to and retrieve from a server asynchronously (in the background) without interfering with the display and behavior of the existing page. Data can be retrieved using the `XMLHttpRequest` object. Despite the name, the use of XML is not required (JSON is often used in the AJAJ variant), and the requests do not need to be asynchronous.[4]

Ajax is not a single technology, but a group of technologies. HTML and CSS can be used in combination to mark up and style information. The DOM is accessed with JavaScript to dynamically display – and allow the user to interact with – the information presented. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads.

Contents

History

In the early 1990s, most Web sites were based on complete HTML pages. Each user action required that a complete page be loaded from the server. This process was inefficient, as reflected by the user experience: all page content disappeared, then reappeared. Each time the browser reloaded a page because of a partial change, all of the content had to be re-sent, even though only some of the information had changed. This placed additional load on the server and used excessive bandwidth.

In 1996, the iframe tag was introduced by Internet Explorer to load or to fetch content asynchronously.

In 1998, Microsoft Outlook Web App team implemented the first component XMLHTTP by client script.

In 1999, Microsoft used its iframe technology to dynamically update the news stories and stock quotes on the default page for Internet Explorer,[5] and created the XMLHTTP ActiveX control in Internet Explorer 5, which was later adopted by Mozilla, Safari, Opera and other browsers as the XMLHttpRequest JavaScript object.[6] Microsoft has adopted the native XMLHttpRequest model as of Internet Explorer 7, though the

ActiveX version is still supported. The utility of background HTTP requests to the server and asynchronous Web technologies remained fairly obscure until it started appearing in full scale online applications such as Outlook Web App (2000)[7] and Oddpost (2002).

Google made a wide deployment of standards-compliant, cross browser Ajax with Gmail (2004) and Google Maps (2005).[8] In October 2004 Kayak.com's public beta release was among the first large-scale e-commerce uses of what their developers at that time called "the xml http thing".[9]

The term "Ajax" was publicly stated on 18 February 2005 by Jesse James Garrett in an article titled "Ajax: A New Approach to Web Applications (http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/)", based on techniques used on Google pages.[3]

On 5 April 2006, the World Wide Web Consortium (W3C) released the first draft specification for the XMLHttpRequest object in an attempt to create an official Web standard.[8]

Technologies

The term *Ajax* has come to represent a broad group of Web technologies that can be used to implement a Web application that communicates with a server in the background, without interfering with the current state of the page. In the article that coined the term Ajax,[3] Jesse James Garrett explained that the following technologies are incorporated:

- HTML (or XHTML) and CSS for presentation
- The Document Object Model (DOM) for dynamic display of and interaction with data
- XML for the interchange of data, and XSLT for its manipulation
- The XMLHttpRequest object for asynchronous communication
- JavaScript to bring these technologies together

Since then, however, there have been a number of developments in the technologies used in an Ajax application, and the definition of the term Ajax. XML is not required for data interchange and, therefore, XSLT is not required for the manipulation of data. JavaScript Object Notation (JSON) is often used as an alternative format for data interchange,[10] although other formats such as preformatted HTML or plain text can also be used.[11]

Asynchronous HTML and HTTP (http://microformats.org/wiki/rest/ahah) (AHAH) involves using XMLHTTPRequest to retrieve (X)HTML fragments, which are then inserted directly into the Web page.

Drawbacks

- In pre-HTML5 browsers, pages dynamically created using successive Ajax requests did not automatically register themselves with the browser's history engine, so clicking the browser's "back" button may not have returned the browser to an earlier state of the Ajax-enabled page, but may have instead returned to the last full page visited before it. Such behavior — navigating between pages instead of navigating between page states — may be desirable, but if fine-grained tracking of page state is required, then a pre-HTML5 workaround was to use invisible iframes to trigger changes in the browser's history. A workaround implemented by Ajax techniques is to change the URL fragment identifier (the part of a URL after the "#") when an Ajax-enabled page is accessed and monitor it for

changes.[12][13] HTML5 provides an extensive API standard for working with the browser's history engine.[14]

- Dynamic Web page updates also make it difficult to bookmark and return to a particular state of the application. Solutions to this problem exist, many of which again use the URL fragment identifier.[12][13] The solution provided by HTML5 for the above problem also applies for this.[14]
- Depending on the nature of the Ajax application, dynamic page updates may interfere disruptively with user interactions, especially if working on an unstable Internet connection. For instance, editing a search field may trigger a query to the server for search completions, but the user may not know that a search completion popup is forthcoming, and if the internet connection is slow, the popup list may show up at an inconvenient time, when the user has already proceeded to do something else.
- Excluding Google,[15] most major Web crawlers do not execute JavaScript code,[16] so in order to be indexed by search engines, a Web application must provide an alternative means of accessing the content that would normally be retrieved with Ajax. It has been suggested that a headless browser may be used to index content provided by Ajax-enabled websites.[17]
- Any user whose browser does not support JavaScript or `XMLHttpRequest`, or simply has this functionality disabled, will not be able to properly use pages which depend on Ajax. Devices such as smartphones and PDAs may not have support for the required technologies, though this is becoming less of a problem. The only way to let the user carry out functionality is to fall back to non-JavaScript methods. This can be achieved by making sure links and forms can be resolved properly and not relying solely on Ajax.[18]
- Similarly, some Web applications that use Ajax are built in a way that cannot be read by screen-reading technologies, such as JAWS. The WAI-ARIA standards provide a way to provide hints in such a case.[19]
- Screen readers that are able to use Ajax may still not be able to properly read the dynamically generated content.[20]
- The same origin policy prevents some Ajax techniques from being used across domains,[8] although the W3C has a draft of the XMLHttpRequest object that would enable this functionality.[21] Methods exist to sidestep this security feature by using a special Cross Domain Communications channel embedded as an iframe within a page,[22] or by the use of JSONP.
- The asynchronous callback-style of programming required can lead to complex code that is hard to maintain, to debug[23] and to test.[24]

Example

Here is an example of a simple Ajax request using the GET method, written in JavaScript.

get-ajax-data.js:

```
// This is the client-side script.

// Initialize the Ajax request.
var xhr = new XMLHttpRequest();
xhr.open('get', 'send-ajax-data.php');

// Track the state changes of the request.
xhr.onreadystatechange = function () {
    var DONE = 4; // readyState 4 means the request is done.
    var OK = 200; // status 200 is a successful return.
    if (xhr.readyState === DONE) {
        if (xhr.status === OK) {
            alert(xhr.responseText); // 'This is the returned text.'
```

```
        } else {
            alert('Error: ' + xhr.status); // An error occurred during the request.
        }
    }
};

// Send the request to send-ajax-data.php
xhr.send(null);
```

send-ajax-data.php:

```
<?php
// This is the server-side script.

// Set the content type.
header('Content-Type: text/plain');

// Send the data back.
echo "This is the returned text.";
?>
```

jQuery example

This example uses the popular JavaScript library jQuery, to do the same thing as the example above.

```
$.get('send-ajax-data.php', function(data) {
    alert(data);
});
```

AngularJS example

AngularJS is another popular library which provides Ajax calls.

```
var app = angular.module("someApp", []);

app.controller("ctrl", ["$http", function($http){
    $http.get('/send-ajax-data.php').success(function(data){
        // do something with data
    });
}
```

See also

- Ajax framework
- List of Ajax frameworks
- AJAX Service Bus
- Comet (programming) (also known as Reverse Ajax)
- Rich Internet application
- WebSocket
- ActionScript

References

1. http://shiflett.org/blog/2007/apr/ajax-is-not-an-acronym
2. https://blogs.oracle.com/arungupta/entry/ajax_vs_ajax_ajax_ofcourse
3. Jesse James Garrett (18 February 2005). "Ajax: A New Approach to Web Applications" (https://web.archive.org/web/20080702075113/http://www.adaptivepath.com/ideas/essays/archives/000385.php). AdaptivePath.com. Archived from the original (http://www.adaptivepath.com/ideas/essays/archives/000385.php) on 2 July 2008. Retrieved 19 June 2008.
4. Ullman, Chris (March 2007). *Beginning Ajax* (http://www.wrox.com/WileyCDA/Section/id-303217.html). wrox. ISBN 978-0-470-10675-4. Archived (https://web.archive.org/web/20080705101852/http://www.wrox.com/WileyCDA/Section/id-303217.html) from the original on 5 July 2008. Retrieved 24 June 2008.
5. MSN.com (http://home.microsoft.com). Home.microsoft.com (1999-12-31). Retrieved on 2013-07-13.
6. "Dynamic HTML and XML: The XMLHttpRequest Object" (http://developer.apple.com/internet/webcontent/xmlhttpreq.html). Apple Inc. Retrieved 25 June 2008.
7. Hopmann, Alex. "Story of XMLHTTP" (http://www.alexhopmann.com/story-of-xmlhttp/). *Alex Hopmann's Blog*. Retrieved 17 May 2010.
8. "A Brief History of Ajax" (http://www.aaronsw.com/weblog/ajaxhistory). Aaron Swartz. 22 December 2005. Retrieved 4 August 2009.
9. English, Paul. "Kayak User Interface" (http://kayaklabs.blogspot.com/2006/04/kayak-user-interface.html). *OFFICIAL KAYAK.COM TECHNOBLOG*. Retrieved 22 May 2014.
10. "JavaScript Object Notation" (http://tapestry.apache.org/tapestry4.1/ajax/json.html). Apache.org. Archived (https://Web.archive.org/Web/20080616113121/http://tapestry.apache.org/tapestry4.1/ajax/json.html) from the original on 16 June 2008. Retrieved 4 July 2008.
11. "Speed Up Your Ajax-based Apps with JSON" (http://www.devx.com/Webdev/Article/32651). DevX.com. Archived (https://Web.archive.org/Web/20080704134131/http://www.devx.com/Webdev/Article/32651) from the original on 4 July 2008. Retrieved 4 July 2008.
12. "Why use Ajax?" (http://www.interaktonline.com/support/articles/Details/Ajax:+Asynchronously+Moving+Forward-Why+use+Ajax%3F.html?id_art=36&id_asc=309). InterAKT. 10 November 2005. Archived (https://web.archive.org/web/20080529080808/http://www.interaktonline.com/support/articles/Details/AJAX:+Asynchronously+Moving+Forward-Why+use+AJAX%3F.html?id_art=36&id_asc=309) from the original on 29 May 2008. Retrieved 26 June 2008.
13. "Deep Linking for AJAX" (http://blog.onthewings.net/2009/04/08/deep-linking-for-ajax/).
14. "HTML5 specification" (http://www.whatwg.org/specs/web-apps/current-work/multipage/history.html). Retrieved 21 October 2011.
15. Hendriks, Erik (23 May 2014). "Official news on crawling and indexing sites for the Google index" (http://googlewebmastercentral.blogspot.ca/2014/05/understanding-web-pages-better.html?m=1). Google. Retrieved 24 May 2015.
16. Prokoph, Andreas (8 May 2007). "Help Web crawlers efficiently crawl your portal sites and Web sites" (http://www.ibm.com/developerworks/library/x-sitemaps/index.html). IBM. Retrieved 22 April 2009.
17. http://googlewebmastercentral.blogspot.com.au/2009/10/proposal-for-making-ajax-crawlable.html
18. Quinsey, Peter. "User-proofing Ajax" (http://www.alistapart.com/articles/userproofingajax).
19. "WAI-ARIA Overview" (http://www.w3.org/WAI/intro/aria.php). http://www.w3.org/. Archived (https://web.archive.org/web/20101026062837/http://www.w3.org/WAI/intro/aria.php) from the original on 26 October 2010. Retrieved 21 October 2010.
20. Edwards, James (5 May 2006). "Ajax and Screenreaders: When Can it Work?" (http://articles.sitepoint.com/article/ajax-screenreaders-work). sitepoint.com. Retrieved 27 June 2008.
21. "Access Control for Cross-Site Requests" (http://dev.w3.org/2006/waf/access-control/). World Wide Web Consortium. Archived (https://web.archive.org/web/20080514034546/http://dev.w3.org/2006/waf/access-control/) from the original on 14 May 2008. Retrieved 27 June 2008.
22. "Secure Cross-Domain Communication in the Browser" (http://msdn.microsoft.com/en-us/library/bb735305.aspx). The Architecture Journal (MSDN). Archived (https://web.archive.org/web/20100329064533/http://msdn.microsoft.com/en-us/library/bb735305.aspx) from the original on 29 March 2010. Retrieved 27 April 2010.

23. Cuthbertson, Tim. "What is asynchronous programming, and why is it so damn awkward?" (http://gfxmonk.net/2010/07/04/defer-taming-asynchronous-javascript-with-coffeescript.html#1). http://gfxmonk.net/. Retrieved 19 October 2010.
24. "Selenium documentation: Fetching a Page" (http://seleniumhq.org/docs/03_webdriver.html#fetching-a-page). http://seleniumhq.org/. Retrieved 6 October 2011.
It is worth noting that if your page uses a lot of Ajax on load then WebDriver may not know when it has completely loaded. If you need to ensure such pages are fully loaded, then you can use Explicit and Implicit Waits.

External links

- Ajax: A New Approach to Web Applications (http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/) — Article that coined the AJAX term and Q&A
- Ajax (programming)

Wikimedia Commons has media related to ***AJAX (programming)***.

Wikibooks has a book on the topic of: ***AJAX***

(https://www.dmoz.org/Computers/Programming/Languages/JavaScript/Ajax) at DMOZ
- Ajax Tutorial (http://www.xul.fr/en-xml-ajax.html) with GET, POST, text and XML examples.

Retrieved from "https://en.wikipedia.org/w/index.php?title=Ajax_(programming)&oldid=668744130"

Categories: Ajax (programming) | Cloud standards | Inter-process communication | JavaScript | Web development | Web 2.0 neologisms

---