

Responsive Web Applications Using Ajax

Matthew J. Travi

Abstract

This paper is a survey of the Ajax methodology and how it can be used to improve the user experience for web applications. The Ajax web application model effectively removes the breaks in workflow from the user interface through intelligent use of only existing web technologies. This methodology can be applied to improve existing applications or used to create powerful new applications.

I. Introduction

The web has become an important platform for application development. With its worldwide reach, the internet has been recognized as a means to deploy applications to a large number of users without actually installing the applications on the users' computers. A major drawback of web applications has been the speed and responsiveness of the interface used to interact with these applications. Ajax has recently become popular as a means to improve the user experience of web applications by removing the requirement of loading a new page when accessing data on the web server. Web applications have been an effective means of accomplishing tasks for some time, but the improvements added by Ajax allow users to be more productive and can make the interfaces friendly enough to use for long periods of time.

Section II of this paper describes the technologies used in the Ajax methodology and shows how a more responsive user interface can be achieved with the use of Ajax. It is assumed that the reader has a basic understanding of these technologies; therefore, implementation details of each will not be described. Section III highlights some areas that developers and design teams should be aware of that can be different than traditional web applications. Knowledge of these issues can help to prevent unexpected problems in the development of Ajax applications.

II. Overview of Ajax

Ajax presents a new web application model which enables the development of responsive web applications. When the internet was first developed, it was intended to provide a means for many users to share documents without much consideration for full applications. The process of loading the entire page was very logical when a user wanted to view a new document. However, web applications have continued to follow this model as the internet has evolved beyond sharing documents.

As web developers create more complicated applications, the web browser cannot respond to user interactions fast enough using the traditional web model. The traditional web model requires the server to respond to every request for updates with a complete web page, including the HTML markup and any images, stylesheets, JavaScript, or other items that are required for that page. When a full page refresh occurs with each user interaction, the workflow is interrupted as shown in Figure 1 and the user wastes time waiting for the numerous pages to load. In order to make the web interface more responsive, the web application model must be modified so that it no longer requires the entire page to be downloaded with every update.

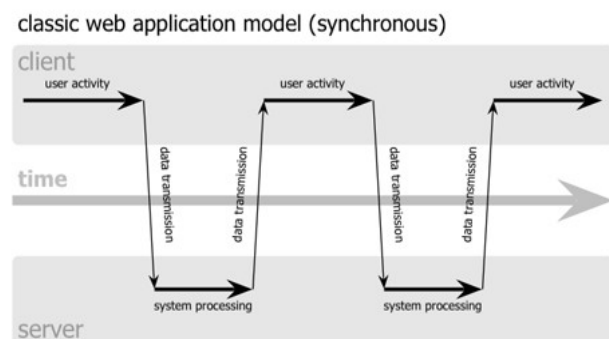


Figure 1: A sequence diagram of the traditional web application model with breaks in the workflow [1]

Asynchronous JavaScript and XML, or Ajax, is a new methodology for making the web interface more responsive. Ajax is not a new technology, but rather a new technique named by Jesse James Garrett of Adaptive Path [1] that uses existing technologies already in use in traditional web applications. Using these technologies together in a specified manner allows Ajax to request new information from the server without interrupting the interface presented to the user in the browser. The technologies used by this methodology are:

- Standard Extensible Hyper-text Markup Language (HTML) and Cascading Style Sheets (CSS) to build the user interface (UI)
- A client-side scripting language, such as JavaScript, to create what can be referred to as an Ajax engine, which becomes a new layer between the UI and the server
- An XMLHttpRequest object to expose HTTP transport functionality to the Ajax engine
- A standard data format, such as XML, to delimit the data that is returned from the server since the response body is sent as a single text string

Although JavaScript and XML are commonly used in Ajax applications, other scripting languages and data formats can be used to accomplish the same tasks. Because of this fact, Garrett explains that Ajax is not meant to be an acronym, but rather a name that establishes a common vocabulary for the technique [2]. JavaScript and XML may be used throughout this document to refer to components of Ajax rather than a more general term; however, these specific languages are not required to implement this methodology.

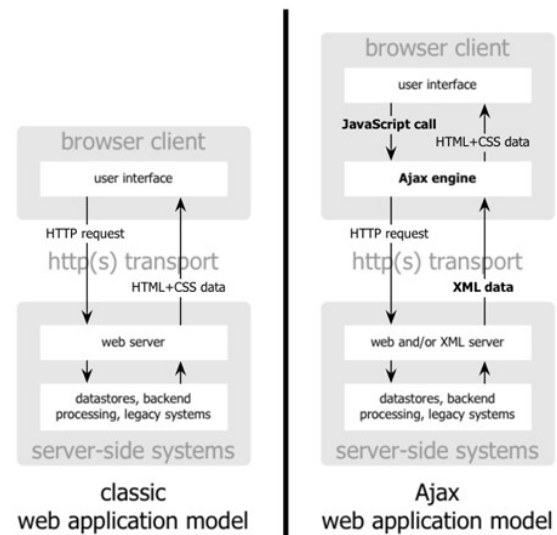


Figure 2: A comparison of the traditional and Ajax web application models showing the additional layer in the Ajax model [1]

The Ajax methodology coordinates the technologies in such a way that a new layer of abstraction, shown in Figure 2, is added to the client side of the traditional web application model. Rather than generating a Hyper-text Transfer Protocol (HTTP) request for each interaction with the user interface, a JavaScript call is made to the Ajax Engine to invoke a specified JavaScript function. This function uses the XMLHttpRequest object to make a request to the server if necessary and alters the user interface with the updated data by manipulating the Document Object Model (DOM).

The DOM is an Object Oriented representation of the HTML markup in a hierarchical tree structure. Document elements can be directly accessed through the DOM and manipulated as needed with JavaScript. Once the DOM has been updated, the browser renders these changes to present them to the user.

Manipulating the DOM to modify the user interface was common even before Ajax was practiced. The process is known as Dynamic HTML (DHTML) and is commonly used to create such things as drop down menus and sortable tables. The major distinguishing factor

between DHTML and Ajax is the fact that server communication is possible with Ajax.

Ajax requests are sent to the server over the standard HTTP transport. An XMLHttpRequest object is used to expose the JavaScript function to this functionality, and the server processes the request as a standard HTTP request. Since this request can be asynchronous and the Ajax engine is handling the request rather than the browser, the user interface is not interrupted as shown in Figure 3, and the user can continue working.

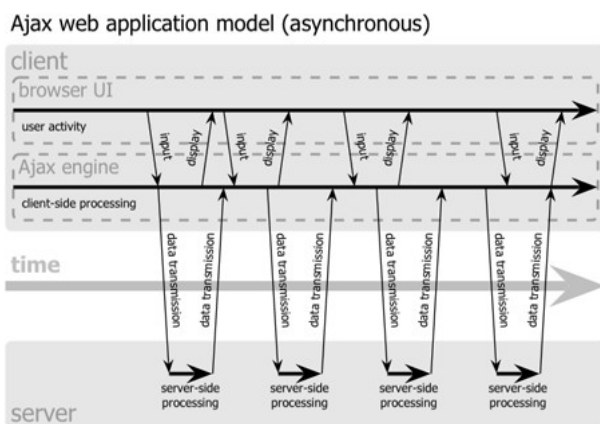


Figure 3: Sequence diagram of the Ajax web application model showing the continuous workflow and asynchronous nature of this methodology [1]

Once the server has received the request from the Ajax engine, the request is processed as a standard HTTP request. Server software is independent of the Ajax methodology, so Java, PHP, etc. can all be used with as much success as with a standard web application. The distinguishing factor with the server side of an Ajax application is the body of the HTTP response. An Ajax application needs only the requested data to be returned, without wrapping it in an HTML document. This can make the response body leaner, reducing the load on the server and allowing better performance.

With only data in the response body, care must be taken so that the data is usable once it has been downloaded to the client. The response body contains only a single text string. A single value can be returned quite

easily, but returning multiple values in a single response becomes slightly more complicated. A standard data format must be used to delimit multiple variable names and values from each other. Although XML is part of the Ajax name and is very commonly used in Ajax, it is by no means the only data format that can be used to send response values in an Ajax application. The format can be as simple as a pipe-delimited string as long as the client side is set up to parse the string correctly.

JavaScript Object Notation (JSON) is another popular data format in use in many Ajax applications. This format is understood by JavaScript as a standard array. This saves processing time on the client side because there is no parsing to be done. This format is also slightly leaner than XML, so it can be sent between the server and the client faster than XML.

Since the Ajax engine does not wait for the response from the server, something must be done to inform the engine that the response has arrived. When creating the Ajax request, a callback function is assigned to handle the HTTP response once it is returned from the server. This function is called when the ready state of the response changes and is responsible for parsing the response body and updating the UI appropriately.

Since there are multiple changes in ready-state before the response is completely downloaded, the callback function must check to see that the ready state is 4 (Completed).

The ready states are:

- 0: Uninitialized
- 1: Loading
- 2: Loaded
- 3: Interactive
- 4: Completed

Once the ready state has been verified as complete, the callback function should also check the status of the response from the server. A HTTP status code of "200" (OK)

means that the request was successfully processed and did not encounter a server error. With this status, the function can continue on with the processing necessary to parse the data and update the user interface correctly. It is a good practice to also alert the user in some way if a status other than 200 was received. The function can check for other specific status codes and handle each individually or together with a generic alert. It is up to the developer to determine what is most appropriate for the application.

Many of the low level details of common Ajax functions can be abstracted from the developers through the use of one of the many frameworks that have been developed. It is beyond the scope of this document to give instruction on the use of these frameworks, but a few that are worth looking into further are Prototype [3], Rico [4], Dojo [5], and Direct Web Remoting (DWR) [6]. DWR could be especially useful with Deere's extensive use of Java Servlets because it provides Remote Procedure Call (RPC) functionality to directly access methods in the servlet from the JavaScript on the client side.

III. Concerns of Ajax Use

There is more to writing Ajax applications than understanding the methodology. There are several issues that a development team must consider. Ajax can be incrementally added to existing applications or built from scratch, and each approach has its advantages and problem areas.

As with all web applications, the browser acts as an interpreter for the code sent from the server. Since not all browsers interpret the code in the same way, web applications are less predictable than a precompiled application that has been installed on the client machine. Ajax applications require more of the logic to execute on the client side, making this a much bigger concern.

The first step to accomplish cross-browser compatibility is to ensure that the HTML and CSS files used for each page are validated

against the World Wide Web Consortium (W3C) standards. Since these are not compiled files, the validators made available by the W3C can point out the syntax errors that would normally be caught by a compile time debugger in a compiled language. Although the web page may appear to render correctly, there can be errors that are easily corrected by validation but could lead to problems later if not corrected.

Beyond basic rendering of the web page in the browser, there can be differences in how the JavaScript code is executed. This adds additional testing time to the project to ensure expected results from the application. In addition, it is possible for users to disable JavaScript within their browser even though the current major browsers have JavaScript support enabled by default. It is the responsibility of the development team to ensure that the Ajax application is degradable to a version that does not require the use of the scripting language in these cases. If an alternate version is not presented in these cases, the users are not alerted that the JavaScript portion of the Ajax application has been disabled in their browser, but are instead simply presented with a broken application.

On a similar note, it is very important to leave a space between the opening and closing `<script>` tags in the HTML file that references a script file with the `src` attribute. Without this space, some browsers will not even load the JavaScript file. Tracking down the error in that browser can be quite difficult without knowledge of this issue [7].

The browser introduces other unexpected behavior that may not be evident until the first attempts are made to use the newly developed Ajax application. Issues such as client-side caching and back button functionality require great care to ensure expected results. Neglecting issues like these will cause the user to become frustrated when the UI does not respond to his action as expect.

Web browsers cache the GET requests made with the XMLHttpRequest object [8]. Because of this attempt by the browser to conserve repetitive downloads, developers must take additional steps to ensure that new information is returned from the server with each request to the same URL. There are two approaches that effectively correct this issue. One requires action on the server side, and the other requires action on the client side. The server side approach involves setting the cache control headers, such as "Cache-Control" and "Pragma," to require the browser to send a new request to the server each time. The client side approach adds a URL variable to the request that is changed with each request so that the browser interprets the request as going to a new URL each time.

History functionality is handled by the browser by making an entry to a history file each time that it loads a new web page. Since an Ajax application does not need to load a new page each time that it requests new data from the server, the URL does not change and the browser's history is not updated. Therefore, when the user attempts to use the browser's back button to return to a previous state, they are instead taken back to a point before the page containing the Ajax application was originally loaded.

There are various techniques to improve the functionality of the back button with Ajax applications. Each technique only works with certain browsers, however. Hidden inline frames can be useful in certain browsers, while an anchor hash or fragment identifier (#) works well with another set of browsers. Complicated solutions like this can be difficult to accomplish for all situations and can be simplified greatly with the various frameworks that have already been developed. One open source solution suggested in [9] is a JavaScript library named "Really Simple History" (RSH) [10]. This framework returns both back button functionality and bookmarking to the many states of an Ajax application. The developer simply instructs the RSH framework to add a history entry and the best method for the

current browser is used. This can even make application history more useful in properly developed Ajax applications because the developer can choose exactly when history entries are made.

Code management becomes a big issue as Ajax applications are developed. One of the nicest things about Ajax functionality is that it can easily be added to existing web applications. As additional Ajax features are added, however, the client-side code becomes more complex and better organization is warranted. The authors of [11] give a good explanation of how client-side code can be refactored and how design patterns should be applied to make the client-side code more maintainable and reusable.

IV. Related Work

Another web application model, proposed by Alex Russell, is called Comet and further improves the updating abilities of the user interface of web applications [12]. An application using Comet maintains the connection between the server and client, allowing updates to be pushed to the user immediately rather than waiting for a request from the user for these updates. This can be especially useful in applications that use polling to request updates from the server in close to real-time.

V. Conclusion

Ajax is a methodology that improves the responsiveness of web applications while only requiring a standard, JavaScript enabled browser. Ajax applications can be easily created from existing web applications. Although building from the ground up can produce a powerful Ajax application, it is not necessary to rebuild an existing application to add Ajax functionality. With the correct planning and management, Ajax can be a very effective method for web development, providing responsive and friendly interfaces to users of the application.

Acknowledgments

This research was supported by Deere & Company as a summer intern project. Special thanks are extended to Dan Murphy for his help and input throughout the project and to the numerous reviewers for their helpful comments.

References

- [1] J. J. Garrett. "Ajax: A New Approach to Web Applications." Internet: www.adaptivepath.com/publications/essays/archives/000385.php, Feb. 18, 2005 [Jul. 6, 2006].
- [2] "Ajax (programming)." Internet: en.wikipedia.org/wiki/AJAX, Jul. 6, 2006 [Jul. 6, 2006].
- [3] "Prototype Javascript Framework: Class-style OO, Ajax, and more." Internet: prototype.conio.net, [Jul. 6, 2006].
- [4] "Rico." Internet: www.openrico.org/rico/demos.page, [Jul. 6, 2006].
- [5] "dojo, the Javascript Toolkit: brought to you by the Dojo Foundation." Internet: www.dojotoolkit.org, [Jul 6, 2006].
- [6] "DWR – Easy AJAX for JAVA | Getahead." Internet: getahead.ltd.uk/DWR, [Jul. 6, 2006].
- [7] B. McLaughlin. *Head Rush Ajax*. Sebastopol, CA: O'Reilly Media, Inc., 2006, p. 403.
- [8] J. Gehrtland, B. Galiraith, D. Almer. *Pragmatic Ajax*. Raleigh, NC: The Pragmatic Bookshelf, 2006, p. 77.
- [9] B. W. Perry. *Ajax Hacks*. Sebastopol, CA: O'Reilly Media, Inc., 2006, pp. 335-354.
- [10] B. Newburg. "Really Simple History." Internet: codingparadise.org/projects/html_history/README.html, [Jul. 6, 2006].
- [11] D. Crane, E. Pascarello, D. James. "Introducing Order to Ajax." *Ajax in Action*. Greenwich, CT: Manning Publications Co., 2006, pp. 69-11
- [12] A. Russell. "COMET: the next stage of AJAX." Internet: www.irishdev.com/NewsArticle.aspx?id=2166, Mar. 24, 2006 [Jul. 7, 2006].