Running head: AJAX

AJAX: Highly Interactive Web Applications

Jason Giglio

jgiglio@netmar.com

Abstract

AJAX stands for Asynchronous JavaScript and XML. AJAX has recently been gaining attention as a way to make web applications more interactive. While it can reduce apparent latency between user interaction and application response, it can cause user interface, maintainability, and accessibility issues.

AJAX: Highly Interactive Web Applications

AJAX is a hot topic lately in web development. AJAX stands for Asynchronous JavaScript and XML. Some popular new web applications such as Google's Gmail and Google Maps are written in an AJAX style. AJAX is not a new technology; JavaScript and XML have been used for many years. It is instead a programming technique using these older technologies to create highly interactive web applications that operate similar to the way local applications operate (Garrett, 2005).

Even though AJAX has great potential to solve some of the problems that arise when developing for a web platform, it does have some drawbacks. Web applications designed using AJAX must pay special attention to usability, as AJAX techniques break some usability guidelines "by default." Workarounds must be put into place to restore usability (Bosworth, 2005).

AJAX also presents new challenges to the development process. Testing is complicated by offloading a good portion of the web application to the client side. Since the web developer has little control over which client is used, strict adherence to web standards and testing with many clients is necessary. Developers must also face learning and testing JavaScript, XML, CSS, XHTML, and the browser DOM, in addition to whatever server side language they choose to use (Hinchcliffe, 2005).

Traditional Techniques

Traditional web applications use GET and POST request methods to submit data to the server. The server then formulates a reply and sends the user an entirely new page with the results on it. Because of the nature of this transaction, the system is not stateful without external tracking. To track state, cookies are used. Cookies are small text files stored on the client side that are used to maintain state data from page load to page load (Snook, 2005).

This technique has some drawbacks. Users must wait after each interaction for the server to process their request and for the new page to render. Slow response time is a common user

complaint when asked about the usability of a web site (Nielsen, 1999).

Slow response time can be addressed in part by ensuring that the server side application uses resources in an efficient way. Database or mainframe access, disk access, or heavy CPU calculations can cause delays in page response. These problems can often be addressed by the programmer. The other main cause of slow page loads is the delay caused by the user's connection to the Internet. This cannot be controlled by the web designer directly; however the designer can reduce the bandwidth required to accommodate users on slow connections (Nielsen, 1999).

Advantages of AJAX

In an AJAX system, the same underlying methods are used, but the actual requests are disconnected from direct user interaction. Instead the JavaScript, which is loaded on the client side, presents one persistent view to the user. As the user interacts with the application, it makes requests as needed to the web server to fetch new data using the browser's Document Object Model to modify the existing page the user is already viewing (Garrett, 2005).

Because of this, the user does not experience as many delays when making requests. Data that is likely to be accessed can be prefetched by the JavaScript on the client side. Requests are fetched asynchronously; the application no longer has to wait for the user to click on a hyperlink or a submit button to update the display, thus feedback can be immediate, much like a desktop application (Singel, 2005).

The user is freed from the technical limitations of the request-reply-request loop that happens in traditional web applications. One example is Google Suggest, which provides a sort of "autocomplete" for search terms as you type them. Through AJAX, the browser does not need to have the entire list fetched ahead of time, it can make asynchronous requests as the user types to fill in the needed entries (Garrett, 2005).

Usability disadvantages of AJAX

Applications developed using AJAX can easily break several accepted web usability guidelines. Bosworth (2005) considers these to be the top ten usability guidelines that AJAX applications often violate:

Not giving immediate visual cues for clicking widgets

AJAX makes requests asynchronously, so when the user performs an action in an AJAX application that must be synchronous, the browser will not give them feedback that anything is happening. In a normal web application, the browser will show a spinner or a progress bar on the status bar when a request is pending. Thus the AJAX developer must provide this feedback themselves as part of the AJAX application (Bosworth, 2005).

Breaking the back button

Users expect standard navigation tools such as “forward” and “back” to work properly. Nielsen (1999) counts breaking the “back” button as one of the most common usability errors that web designers commit. AJAX applications often break the “back” button because the application appears on a single page that is updated with new information.

Changing state with GET requests

W3C Technical Architecture Group (2004) has published guidelines on the appropriate use of the GET and POST methods in a web application. In general, they recommend that GET be used for simple requests that do not alter the state of a resource on the server side and that POST be used for complex operations or operations that change the state of a resource. For example, a simple search query should be GET, but submitting a form to create an account on a web site should be POST.

AJAX applications must avoid using GET requests that alter the state of resources on the server side. It is often convenient to use such requests to facilitate the development of an AJAX

application, but such practices should be avoided as they can cause resources to be inadvertently modified if the user visits or revisits a URL that contains such a GET statement.

Blinking and changing parts of the page unexpectedly

The asynchronous nature of AJAX can cause unexpected updates to page elements that are not part of what the user is currently concentrating on. Nielsen (1999) has coined the phrase "animation avoidance" to describe how users generally ignore areas of a web page that blink or animate unexpectedly. This could confuse the user or cause them to miss important information.

Not using links that be communicated or bookmarked

Because AJAX applications often display on a single page, the user will not be able to easily communicate or bookmark any particular snapshot of data unless the developer takes special measures to restore this functionality. This also presents problems for search engines and other automated web "bots," who may not be able to index the data on the site (Bosworth, 2005).

Too much client side code slowing down the browser

JavaScript is not a high performance language. Even as CPUs become faster, site performance is still a concern with AJAX applications that include more JavaScript code than ever before (Bosworth, 2005).

Inventing new user interface (UI) conventions

Nielsen (1999) counted the non-standard use of UI widgets as the number three most common usability problem on the web, even before AJAX was even conceived. Users expect a consistent UI that works in similar ways no matter which application they are using. UI elements communicate information about the types of input that are required of the user. For example, radio buttons communicate that the user must make a mutually exclusive choice.

Not cascading local changes to other parts of the page

AJAX applications are often confined to a section of the screen. If the user makes a change within the AJAX application that should update all page elements, then the developer must remember to update all elements, even those outside the AJAX application (Bosworth, 2005).

Asynchronously performing batch operations

Much like the radio buttons above, users derive much information from UI elements. If an AJAX application used radio buttons perform an action that altered a resource asynchronously, then the user may become disoriented. Users are accustomed to the ability to change their mind before submitting. The asynchronous nature of AJAX can encourage bad practice in this regard (Bosworth, 2005).

Scrolling the page and disorienting the user

Because AJAX is asynchronous, operations that would cause a text reflow can happen at any time. If the user is reading a section of text and the AJAX application inserts more content above the text, then the entire page may scroll down, which may push the text the user was reading off the screen completely (Bosworth, 2005).

Technical disadvantages of AJAX

Obasanjo (2005) raises several points regarding the technical challenges facing an AJAX application. One problem is the slightly differing implementations of JavaScript between various browsers. While standards compliant code largely solves this problem, sometimes browser capability detection is necessary. Detecting the browser capabilities on each page is inefficient, and a method must be developed where detection can be done once per user.

Along those same lines, some browsers will not have JavaScript enabled at all, so the AJAX application will not function. Some allowance must be made for users that do not have JavaScript

enabled. This could mean twice the development work if a good abstraction cannot be found that allows for both AJAX and non-AJAX front ends.

Some clients may create excessive connections to the server due to the asynchronous nature of AJAX. Since the developer does not have direct control over the client, it may be hard to control the server load. Hinchcliffe (2005) also points out that AJAX creates a need for fast handling of many small back end XML messages, an area where traditional web service backends are lacking. Combine this with the concerns of Bosworth (2005) regarding client speed, and AJAX can potentially be a very slow platform.

Conclusions

If AJAX was proposed 2 years ago in a magazine article or journal, then I do not believe anyone would have taken it seriously. The real-world, working applications that Google has developed prove that it is possible and can work well for at least some web applications. Lacking their leadership in this area, it is doubtful this paper would have ever been written.

Questions remain as to how widely applicable AJAX techniques will be. It is not clear how many problem domains lend themselves well to AJAX implementations. Google has proven that it can be used to manage very large graphical datasets with Google Maps and can make a decent mail client with Gmail. Microsoft, always an imitator of the successful, is working on a new version of Hotmail based on AJAX and a sort of web portal based on AJAX called start.com (Obasanjo, 2005).

It also remains to be seen how well developers will overcome the severe usability problems that AJAX techniques can create. Usability is an often neglected area of web design, and techniques that encourage bad usability such as AJAX walk a dangerous line (Bosworth, 2005).

AJAX may seem to have a long list of disadvantages, caveats, and seemingly insurmountable problems, with a short list of advantages. What makes it so compelling is that the primary advantage is one that has been sought for many years: The potential to turn the web into a

full-fledged application platform, suitable for nearly any application.

References

Bosworth, A. (2005, 18 May). *Ajax Mistakes.* Retrieved December 26, 2005 from http://sourcelabs.com/ajb/archives/2005/05/ajax_mistakes.html.

Garrett, J. J. (2005, 18 February). *Ajax: a new approach to web applications.* Retrieved December 25, 2005 from http://www.adaptivepath.com/publications/essays/archives/000385.

Hinchcliffe, D. (2005, 18 August). *State of Ajax: Progress, Challenges, and Implications for SOAs.* Retrieved December 26, 2005 from http://hinchcliffe.org/archive/2005/08/18/1675.aspx.

Nielsen, J. (1999, 30 May). *Top-10 New Mistakes of Web Design.* Retrieved December 25, 2005 from http://www.useit.com/alertbox/990530.html.

Obasanjo, D. (2005, 16 August). *Moving Beyond the Basics: Scott Isaacs on AJAX Design Patterns.* Retrieved December 26, 2005 from http://www.25hoursaday.com/weblog/PermaLink.aspx?guid=23a58e59-0a8d-43e%4-ab18-a6d64ca5be87.

Singel, R. (2005, 5 August). *You Say You Want a Web Revolution.* Retrived on December 26, 2005 from http://www.wired.com/news/technology/0,1282,68403,00.html.

Snook, J. (2005, 28 June). *Powering the web with HTTP.* Retrieved December 25, 2005 from http://digital-web.com/articles/powering_the_web_with_http/.

W3C Technical Architecture Group. (2004, 21 March). *URIs, Addressability, and the use of HTTP GET and POST.* Retrieved December 26, 2005 from http://www.w3.org/2001/tag/doc/whenToUseGet.html.