THE EFFECT OF
ASYNCHRONOUS JAVASCRIPT AND XML USAGE
ON USER AND DEVELOPER EXPERIENCE

By

Sarah Esther Williams

Directed Research under
Dr. Michael Crowley

University of Southern California

Spring 2006
# Table of Contents

**TABLE OF CONTENTS**

OVERVIEW

AJAX AND RELATED TECHNOLOGIES

<table>
<thead>
<tr>
<th>Technology</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript</td>
<td>4</td>
</tr>
<tr>
<td>XML</td>
<td>5</td>
</tr>
<tr>
<td>XAML</td>
<td>5</td>
</tr>
<tr>
<td>XUL</td>
<td>6</td>
</tr>
<tr>
<td>Java Applets</td>
<td>6</td>
</tr>
<tr>
<td>JScript</td>
<td>7</td>
</tr>
<tr>
<td>Flash and ActionScript</td>
<td>7</td>
</tr>
<tr>
<td>SVG</td>
<td>7</td>
</tr>
<tr>
<td>DHTML</td>
<td>7</td>
</tr>
<tr>
<td>Industry Standards</td>
<td>8</td>
</tr>
<tr>
<td>Asynchronous JavaScript and XML</td>
<td>9</td>
</tr>
</tbody>
</table>

METHODOLOGY

DEVELOPER EXPERIENCE

USER EXPERIENCE

SUMMARY OF FINDINGS

DEVELOPER EXPERIENCE

USER EXPERIENCE

CONCLUSIONS AND RECOMMENDATIONS

APPENDIX A – SURVEY QUESTIONS

USER SURVEY ON GMAIL AND AJAX

DEVELOPER SURVEY ON AJAX

APPENDIX B – AJAX FRAMEWORKS AND LIBRARIES

SOURCES
OVERVIEW

AJAX, Asynchronous JavaScript and XML, was coined by Jesse James Garrett and his company Adaptive Path to denote the combined use of asynchronous JavaScript and XML to create richer interactive web applications. As the name suggests, AJAX is not in itself a singular tool. Rather, it is a combination of useful technologies each bringing its own advantages and disadvantages to the table. Traditional web applications afford direct communication between the server and the client through consecutive HTTP requests sent from the browser. Server-side, any number of operations can take place including database queries, web service calls etc. With Asynchronous JavaScript and XML, an AJAX engine, written in JavaScript, is inserted between the server and the client to render the UI of the page and to handle asynchronous communication. This dynamic operation is made possible with JavaScript’s XMLHttpRequest Object, and results in updates to the user interface that are seamless and, in most cases, pleasing to the user. AJAX is officially a combined use of XHTML and CSS, the Document Object Model, XML and XSLT, the XMLHttpRequest object and JavaScript. However, many developers often combine asynchronous server calls in scripting language with other web technologies like PHP and Perl to achieve the same benefits of AJAX.

Before AJAX, almost every user interaction with a website resulted in a refresh of the page that informed the user of the results of their action. AJAX introduces minimal page reloading and provides the same feedback to the user transparently thereby improving human computer interaction and even cutting the time taken to do certain tasks in half. Pages are modified incrementally instead of being refreshed on every major user action.

AJAX is still at the stage of public commentary but has earned the approval of most developers. As such, it is rapidly moving towards becoming an industry standard in the web development world. Through online articles and web blogs, those who use the technology discuss its potential and share resources. To my knowledge, there has not yet been an academic study on any aspect of AJAX as much of its impact exists outside of a university setting.

This study aims to

- Enumerate the basic characteristics of AJAX and its most essential components.
- Determine the positive and negative effects of AJAX use on the experience of the programmer especially when compared with the usage of other web technologies.
- Determine the positive and negative effects of AJAX use on the experience of the user.
- Draw conclusions on the potential of AJAX based on its benefits as judged by a subset of its developer and user community.

For developers, AJAX brings a myriad of opportunities to create rich web applications but at the same time can break traditional browser functionality, for example the BACK and FORWARD browser buttons. As far as the user is concerned, even the elimination of constant refreshing can be confusing. Related to this problem is that of user expectancy. A user, who may expect a page to reload, could unnecessarily repeat actions based on the lack thereof. Therefore this study seeks to assess the profits and losses of AJAX as described by persons closest to its implementation: developers and the users themselves.
AJAX and Related Technologies

JavaScript

Because AJAX is based on asynchronous calls to the server done in JavaScript, a discussion of its characteristics is paramount.

JavaScript is a weakly typed interpreted (rather than compiled) language based on the ECMA Script standard (ECMA-262). It is usually embedded into a host environment such as HTML pages where it can take full advantage of the Document Object Model and manipulate the UI of the page. Programmers can code client side JavaScript or server side JavaScript called LiveWire JavaScript. Though JavaScript relies heavily on built-in objects, it has no notion of classes or inheritance. Many companies and programmers, however, have built wrappers and engines around JavaScript that simulate object oriented behavior like inheritance. JavaScript can behave as a procedural or object oriented language. To achieve a crude notion of OOP, methods and variables are linked to empty ‘objects’ at run time. No classes are ever defined. As it is a loosely typed language, errors cannot be detected until the script is actually run. ¹

The foundation of AJAX is a JavaScript object called the XMLHTTPRequest. Though first created by Microsoft as an ActiveX object in IE, it is now supported Safari 1.2, Mozilla 1.0/Firefox and Netscape 7 in addition to by Internet Explorer 5.0+. It is an API built into the browser that enables many scripts including JavaScript, VBScript, Jscript and others to sent asynchronous requests to the server over HTTP. ²

In summary, the advantages and disadvantages of JavaScript are as follows:

Advantages

- Cross Browser support.
- No additional plug-ins necessary to make JavaScript code run on a client machine.
- The web programmer can create sophisticated interfaces with a considerable amount of graphics and animation without the hefty file sizes and prolonged response times usually associated with Flash and Java applets.
- The language is easy to learn and, in theory, does not require expensive development software, though this could be useful.
- JavaScript activity is primarily client side which reduces server load.

Disadvantages

- JavaScript errors can sometimes be hard to find since they cannot be detected until the script it run. Furthermore, since JavaScript could make extensive use of the Document Object Model, perfectly formed JavaScript code could result in bugs because of errors in the DOM.
- Some browsers disable JavaScript activity and a handful simply do not support it. Developers need to program for that case so that the application degrades gracefully. This may take the form of extra server side mechanisms that ensure that client side JavaScript was indeed executed. Other problems may arise from different versions of JavaScript running in different browsers.

http://www.mozilla.org/js/
JavaScript debuggers are extremely expensive while freeware versions lack a rich toolset.

Because JavaScript runs within the browser on the user’s machine itself, it has been exploited in the past by malicious parties. Therefore, the onus is on the programmer to pre-cater for security holes in the application caused by JavaScript.¹

XML

XML is a quick and easy way of describing, storing and transporting data across various systems that may want to represent the data differently.

Advantages

- Readability: XML can be read directly by a human or by a machine.
- As it supports Unicode, almost any kind of data can be represented in XML including traditional data structures like lists and trees in addition to object instances.
- XML can describe data as well as assign values to them using attributes of the XML nodes.
- XML is truly platform independent making it untouchable by changes to technology. It is stored in plain text files so, in theory should be able to function on any machine.
- It is robust and has a long standing history, allowing it to boast accepted international standards and efficient accompanying software.
- Because of its legacy syntax, parsing algorithms are simple, efficient and consistent.

Disadvantages

- Verbose syntax makes for tedious reading and scripting by hand. Furthermore, the redundancy of the language leads to bigger file sizes which could hurt performance in cases where bandwidth is limited, for example cell phones.
- Creating code objects from XML and vice versa causes considerable overhead in some cases for example, in interpreting data types like floating point numbers from attributes within the XML.
- Increased overhead results from an XML parser having to check for malformed XML. This can be particularly hurtful to performance on embedded systems.²

XAML

XAML, Extensible Application Markup Language, formerly known as Extensible Avalon Markup Language, is an XML based, declarative programming language created for implementation with WinFX. It is a user interface markup language capable of describing graphically rich user interfaces such as those in Macromedia Flash pages. XAML files are created using Microsoft Developer tools such as Visual Studio and Microsoft Expression Interactive Designer and are interpreted by the Windows Vista display subsystem.

¹ [http://www.wdvl.com/Authoring/JavaScript/PracticalJS/practicaljs1_3.html](http://www.wdvl.com/Authoring/JavaScript/PracticalJS/practicaljs1_3.html)
Advantages
- The strengths of XML apply here.
- XAML as with other UI markup languages, is quick to build and easy to modify.
- With XAML, there is a separation between application interface and application logic. Program logic is housed in “code- behind” files while UI presentation information is stored in XAML.

Disadvantages
- The weaknesses of XML apply here.
- XAML is designed to be interpreted by a Windows graphics display system only. While there is great potential for interoperability, XAML was made for use in Windows applications and is not a cross platform UI markup language. 

**XUL**
XUL, XML-based user Interface Language is Mozilla’s user interface markup language.
Advantages
- Any UI markup language is quick to build and modify.
- The advantages of XML apply here.

Disadvantages
- The disadvantages of XML apply here.
- Applications that are built on XUL, such as the Mozilla Amazon browser, cannot be run in Internet Explorer, but rather only in browsers that supported are by Mozilla for example Firefox and Netscape.

**Java Applets**
Java applets are applications written in Java that compile to Java byte code to be interpreted and run in a web browser with the help of a Java Virtual Machine.
Advantages
- It is supported by many web browsers much in the same way that JavaScript is supported.
- Though the initial load time may be lengthy, applets will usually cache themselves within the browser for easy retrieval later on.
- Java applets are full blown applications and as such can give the user a seamless desktop like feel to their web experience. The UI is updated incrementally and applets support animation as well.
- Applets will run with any version of Java installed in the servers

Disadvantages
- Load times for applets are considerably longer than other web technologies.
- There is much less flexibility in designing good user interfaces in applets than in HTML.
- A Java Plug in is required which is disabled on some systems for security issues.

---

**Jscript**

Jscript is Microsoft’s implementation of ECMAScript much in the same way as JavaScript is Netscape’s rendition of the same. Most ASP scripts are written in Jscript or VBScript.

**Flash and ActionScript**

Actionscript is another ECMA based scripting language created for implementation within Flash Movies.

Advantages

- Since ActionScript is based on the same standard as JavaScript and JScript, developers can make a seamless transition into working with one or another of these languages.
- Browser compatibility.
- Vast flexibility on the part of the designer in creating engaging, elegant user interfaces rich with animation.

Disadvantages

- Development tools are extremely expensive.
- Can suffer from extended load times.
- Websites based entirely on Flash almost always endure certain usability problems like broken back and forward buttons, and inability to bookmark specific pages.
- Flash scripting and building Flash websites in general is a very specific skill that may be difficult to learn. While other scripting languages are made to manipulate the DOM within a browser window, ActionScript was created to define and manipulate animations, audio, text and event handling within movies.

**SVG**

SVG, scalable vector graphics is an XML markup language used to describe two dimensional graphics and graphical applications. SVG handles vector graphic shapes, digital images and text.

Advantages

- Flexible
- SVG is specifically designed for interoperability with CSS, DOM and SMIL (Synchronized Multimedia Integration Language). It is recommended by W3C and, as such, is a widely accepted standard.

Disadvantages

- A Plug-in required in browsers that don’t provide native support.  

**DHTML**

DHTML is simply the combined use of HTML, a browser-side scripting language (JavaScript or any other such language), cascading style sheets and the Document Object Model. By this definition, DHTML can be considered AJAX without the XML if asynchronous JavaScript is used.

---


Advantages:
- Rapid updates to the user interface. The application is given a professional, smooth look and feel.
- No additional plugs necessary. All technology involved can usually be interpreted by the browser.
- The life of the web designer\(^\text{10}\) is made much easier as he/she can make use of DHTML’s features to design interfaces that are functional, elegant and usable.

Disadvantages:
- The life of the programmer is made harder as coding in DHTML results in longer code and greater care is needed in constructing and manipulating parts of the page.
- Since DHTML relies on JavaScript, it is often very useful to have a good JavaScript debugger. These are often extremely expensive, with freeware versions lacking a rich toolset.

**Industry Standards**

ECMA Script is the accepted basis for many web scripting languages and allows developers to adopt one language or another quickly. Each one has access to the XMLHttpRequest Object via its own code. Open AJAX is an initiative undertaken by various companies and development groups to create industry standards for AJAX. These groups include IBM, Microsoft, Mozilla, the Dojo Foundation, BEA Systems Borland, Novell, Yahoo!, Oracle, Red Hat and many more.\(^\text{11}\) The [World Wide Web Consortium](http://en.wikipedia.org/wiki/Ecmascript) published a Working Draft specification for the XMLHttpRequest object's API on 05 April 2006. While this is still a work in progress, 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".

\(^\text{10}\) For the purpose of this paper, we will define the web designer as one who is concerned with visual appearance, color schemes and consistency throughout a web application. The web programmer is one who is concerned with the programming involved to create these visual elements.

Table 1: AJAX compared with many other web technologies

<table>
<thead>
<tr>
<th></th>
<th>Rapid Interface Updates</th>
<th>Cross Browser Compatibility</th>
<th>Native Support in Browser</th>
<th>Negligible Download Time</th>
<th>No extra bulky files</th>
<th>Flexibility for Designer</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAVASCRIPT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SVG</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XAML</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XUL</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLASH/ ACTIONSCRIPT</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>JAVA APPLET</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHTML</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>AJAX</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Asynchronous JavaScript and XML**

This term was coined as recently as February 18\textsuperscript{th}, 2005 in an article by Jesse James Garrett called *AJAX: a new approach to web applications*.\textsuperscript{13} It is in theory the combined use of asynchronous communication with the server made possible by the XMLHttpRequest object, DOM which defines the structure of the webpage, CSS, XML and XSLT. But in reality, AJAX is used by developers to describe the mingling of asynchronous calls made through JavaScript with any number of web technologies including PHP, VBScript and even just plain HTML.

Table 1: AJAX compared with many other web technologies, shows that AJAX provides just about everything needed for an enriched UI experience when compared with other competing techniques.

Advantages:

\textsuperscript{12} Synonymous here to a desktop application-like look and feel

Final User Interface can boast a Desktop Application-like look and feel.
Response Times are considerably lower than traditional applications
There is little to no page refreshing giving the application a sophisticated appearance and a responsive UI.

Disadvantages:
- In almost all browsers, the BACK and FORWARD buttons are broken because of asynchronous updates to parts of the page. This mechanism does not update the browser’s history tree. The consequences of the lack of this feature vary from browser to browser.
- As with Flash scripting, in many cases, specific pages may not be bookmark-able.
- Since interaction between the server and the client occurs so quickly, some users who expect a page to reload will repeat actions unnecessarily sometimes with detrimental effects.
- In some cases there may be no way to open certain links in new windows.
- Since AJAX from within the browser, web applications designed poorly (poor management of asynchronous calls to the server) could cause sudden increases in CPU usage and hurt performance of all programs on the client’s machine.

Methodology

If AJAX is to define the future of internet web based user interfaces, a bare bones comparison between AJAX and related technologies is just one factor to consider. Another important measure is its effect on its developer and user community.

Developer Experience

To determine the impact of AJAX on developers, a survey was created and made available online in discussion forums related to web technologies, JavaScript programming, AJAX programming, and AJAX development software: S AJAX development at modernmethod.com, DevShed.com, Webdeveloper.com, ASP.net, Backbase.com, DevX.com, InteraktOnline, CodingForums.com, Tek-Tips.com, DigitalPoint.com, and Webmasterworld.com. In addition, links were personally sent to many students with AJAX experience at the University of Southern California. The survey was exposed to developers who had varying experience with AJAX and related tools and libraries. Respondents were not forced to answer any of the questions and could exit the survey at any point. The questionnaire was open from April 26th to May 7th, 2006.

The objectives of the survey were:
- To provide context by gleaning general information from the developer.
- To determine how the AJAX programmer experience compares with traditional web programming and web scripting.
- To relate the programmers’ experience with AJAX to the difficulty level of the projects worked on (because the complexity of a task affects developer experience as much as the technology used itself).
- To determine whether in retrospect, these programmers would reuse AJAX and to find out whether or not they feel that alternatives such as DHTML are more useful.
To enumerate prominent Software development tools for AJAX and to ascertain whether programmers find these tools effective and essential to their experience?

To determine which AJAX weaknesses developers find important to address in their programming.

To compare the developers understanding of a desktop-like UI within a web application with users’ understanding of the same.

Each developer was exposed to the following sections: Basic Information, General Questions, AJAX Experience, AJAX Development Environments, AJAX features, The future of AJAX and Final Comments. The first two sections concerned the developer’s age, gender, proficiency with AJAX in addition to their understanding of desktop applications and the desktop GUIsation of web software. In the AJAX Experience section, respondents were asked about how long they have used AJAX, how difficult it was to learn and how their experience with other web technologies compared to their experience with AJAX. The next section asked developers to evaluate certain development environments while AJAX Features tested how much stress developers put on fixing certain AJAX problems. The Future of AJAX posed questions relating to their opinions on the potential of AJAX. The final section sought to summarize their satisfaction with AJAX and to determine what suggestions for additions, changes, or removals they would give to developers of AJAX development environments. Appendix A contains all questions and options of this survey exactly as presented to the respondent.

**User Experience**

It was a challenge to find ways to quantitatively and qualitatively determine how AJAX has impacted user experience. Most users do not know what AJAX is, let alone how to appreciate the benefits it can bring. What is familiar to users however, are feature sets that make applications stand out from their competitors. For the purposes of this study, Google’s GMAIL email client was chosen as a means of ascertaining users’ reactions to AJAX. A survey was made available online to mostly college aged students. They were not asked direct questions about AJAX but instead, they were asked to rate features of GMAIL according to importance and ease of use. They were also asked to identify features that they appreciated most and least about GMAIL. Finally, they were given the option to share their opinion on the disadvantages of AJAX discussed above. In the analysis of the responses, one will be able to judge whether users respond positively or negatively to features that AJAX provides in GMAIL. The questionnaire was open from April 16th to May 7th, 2006.

The objectives of the survey were:

- To provide context by doing a user analysis which included information about the machines they used, proficiency with using the internet etc.
- To determine whether beneficial AJAX-facilitated features improve user experience in the minds of users themselves namely:
  - No/little Page refreshing
  - Improved response times.
  - Asynchronous calls allowing for immediate feedback for small sections of the page.
  - Perceived Desktop application feel given to web applications
To determine whether disadvantageous AJAX-facilitated features degrade user experience in the minds of users themselves namely:
  o Unnoticed feedback mechanisms
  o Broken BACK and FORWARD buttons in the browser
  o Inability to open certain links in a new window

To determine whether AJAX really does make online applications behave like desktop applications in the minds of users.

To compare the developers understanding of a desktop-like UI within a web application with users’ understanding of the same.

The User Questionnaire was divided into the following sections: Basic Information (age, gender, operating system, connection speed, etc), General Questions About Web Usage (their understanding of desktop applications and their experience with certain disadvantages of AJAX), GMail Use (extent of experience with GMAIL and the nature of GMAIL use), GMail Experience (evaluation of GMAIL’s features and ranking of certain general website features), and Final Comments (Opinions on what should be added to, removed from and changed in GMAIL). Appendix A reproduces all survey questions and options exactly as they were presented to the respondent.

Special Note:
  When the user survey was released, most of the questions forced a response from users. This functionality was turned off after approximately 20 to 30 respondents had submitted surveys.

Summary of Findings

Developer Experience

Demographics
  Developers in general seemed less inclined to stick with the survey to completion. Though there were 56 respondents (after eliminating duplicates and unreadable submissions), many of the questions were only answered by 32 to 36 persons. Out of the 54 respondents who gave their gender, a meager 3 were women. Most respondents were in the 23-25 age range (25%) with the oldest respondents found in the 50 to 54 age range and youngest under 20 years of age. Most of the participants consider themselves beginner or intermediate AJAX programmers (69.3%) according to the definitions given in the question prompt (see Appendix A). All three women who took the survey placed themselves in these categories. 21.8 % of developers work with AJAX as a part of their job.

General Questions
  When asked what features make web applications behave more like desktop GUIs, one user, rather astonishingly, indicated that there were no AJAX features that bring about this effect. Respondents were allowed to choose multiple options and 62.5% chose minimal page refreshing, 53.6% chose Immediate Feedback and 51.8% chose fast user interaction. Open ended responses indicated that the hiding of functions from users (security) is an effect of AJAX features in this context.
As a whole, it seemed that developers who took the survey were attracted to AJAX because of its popularity in the market. The perception that it is trendy and in high demand was enough of a lure for these individuals. One developer used it to combat high load and lag times in his/her website.

The survey was successfully exposed to developers with widely ranging experience with AJAX. The median amount of time these developers have spent with AJAX is at about the 8 month mark. As Table 2 and Table 3 show, the life of AJAX learning is online. Most developers hear about AJAX and built their skills using online blogs, journals and tutorials. Only one respondent learned AJAX in a school setting. Some developers find it helpful to read source code directly as a way of learning AJAX.

**QAJAXStart:** How did you hear about AJAX?

<table>
<thead>
<tr>
<th>Response</th>
<th>20%</th>
<th>40%</th>
<th>60%</th>
<th>80%</th>
<th>100%</th>
<th>Frequency</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>From a friend</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7.1%</td>
<td>4</td>
</tr>
<tr>
<td>From a blog or journal online</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>26.8%</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>From a professor and/or within a school/ university setting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.6%</td>
<td>2</td>
</tr>
<tr>
<td>At your workplace</td>
<td></td>
<td></td>
<td>14.3%</td>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>In a technical book</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.6%</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Other (please specify)</td>
<td></td>
<td>19.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
</tr>
</tbody>
</table>

Table 2

**QRResources:** What steps did you take to become skilled in AJAX programming?

<table>
<thead>
<tr>
<th>Response</th>
<th>20%</th>
<th>40%</th>
<th>60%</th>
<th>80%</th>
<th>100%</th>
<th>Frequency</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>I consulted/ was mentored by someone who knew more about AJAX than I did</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12.5%</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>I consulted online blogs and journals</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>37.5%</td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>I followed online tutorials</td>
<td></td>
<td></td>
<td></td>
<td>42.9%</td>
<td></td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>I attended AJAX/web developer conferences</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.9%</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>I read books/manuals on AJAX</td>
<td></td>
<td>19.6%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>I learned AJAX in school</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.8%</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Other (please specify)</td>
<td></td>
<td>21.4%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12</td>
</tr>
</tbody>
</table>

Table 3

Table 4 further reiterates that developer conferences, human mentors and technical books are not as useful to developers as online resources. 5 respondent claimed to have attended developer conferences however, none of them found their attendance helpful in becoming skilled in AJAX programming. The same is true for the respondent that leaned AJAX in an academic setting.
That said, 33.3% of developers found AJAX decisively easy to learn. 25% of respondents were familiar with AJAX within a few days (but under a week) while 2 Respondents were still uncomfortable with programming in AJAX after over three months of exposure to it. In terms of complexity and time to implement, it most of the participants work on moderately difficult projects (41.7%) From open-ended questions inquiring about easy and difficult aspects of AJAX to learn, one can conclude that for these developers, learning to work with the XMLHttpRequest object was rather easy while gaining a mastery of JavaScript on the whole was extremely difficult. Those who found AJAX easiest to work with were those who had a solid understanding of JavaScript from previous projects. Developers seem to agree that it is not enough to have a solid understanding of the XMLHttpRequest Object but rather proficiency in JavaScript, which at times can be clumsy in its attempts at OOP features. XML was generally regarded as easy to work with. The consensus on the hardest aspect of AJAX was debugging. Inability to kill, put to sleep, or track threads made it extremely difficult for developers to track errors among their asynchronous calls.

Most participants used PHP (53.6%) alongside and/or outside of their AJAX use while 12.5% used Flash, 35.7% used ASP, 8.9% used XUL, 0% used XAML, and 23.2% used Java. 32 respondents chose to answer this question.

When asked to compare AJAX with other web technologies, most developers wouldn’t or couldn’t make the comparison. Those who thought AJAX was more difficult complained that

1. learning AJAX meant leaning at least 2 different languages which made for a steeper learning curve,
2. there were robust IDEs on the market
3. it was easy to create websites that didn’t align with all the standards of all the technologies involved in AJAX
4. browser compatibility issues were rampant

<table>
<thead>
<tr>
<th>QResourceRanking_A: Please rank these resources in order of how useful they were to you when learning AJAX.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- Least useful  6- Most useful</td>
</tr>
<tr>
<td>N/A - I didn’t use this resource</td>
</tr>
<tr>
<td><img src="image" alt="Table 4" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>N/A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>I consulted/ was mentored by someone who knew more about AJAX than I did</td>
<td>25</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>35</td>
</tr>
<tr>
<td>I consulted online blogs and journals</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>35</td>
</tr>
<tr>
<td>I followed online tutorials</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>12</td>
<td>33</td>
</tr>
<tr>
<td>I attended AJAX/web developer conferences</td>
<td>29</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>34</td>
</tr>
<tr>
<td>I read books/manuals on AJAX</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>34</td>
</tr>
<tr>
<td>I learned AJAX in school</td>
<td>29</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>Total</td>
<td>116</td>
<td>9</td>
<td>9</td>
<td>12</td>
<td>20</td>
<td>14</td>
<td>24</td>
<td>204</td>
</tr>
<tr>
<td>% by Row</td>
<td>56.8%</td>
<td>4.4%</td>
<td>4.4%</td>
<td>5.9%</td>
<td>9.8%</td>
<td>6.9%</td>
<td>11.8%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>
One developer felt that AJAX was less difficult than others but this individual was already familiar with DOM, and XML.

**AJAX Development Environments**

Interestingly enough, 19 out of the 32 developers, who indicated their IDE preferences, more than half, do not use AJAX IDEs at all. They chose to manipulate the XMLHttpRequest object directly. Furthermore, 2 individuals are in the process of developing their own development environment. However, among those that used IDEs, Developers identified object oriented abstraction of XMLHttpRequest options as a strength of AJAX IDEs. The following advantages were identified according to particular developer kits:

1. **SAJAX** – great for use in relation to PHP
2. **PROTOTYPE** - helpful with managing JavaScript code; gives the Developer plenty of options
3. **OPENRICO** – good for animation; excellent extension of the PROTOYPE library
4. **AJAX Extended** – highly portable (works with older browsers)
5. **BACKBASE** – allows focus to be fixed on higher level problems versus lower level ones (I feel that ALL AJAX IDE are effective at this).
6. **YAHOO UI Library** - easy to use, simple and well-documented.

Most developers indicated that the biggest weaknesses surrounding these Development kits are poor documentation, and extraneous features. The steep learning curve and verbose documentation of small snippets of code were both frustrating to respondents.

---

**QEssentialFeatures_A:** Please rank the following AJAX IDE features according to how necessary they are in making a good development tool for AJAX.

9 = most necessary
1 = least necessary

<table>
<thead>
<tr>
<th>Feature</th>
<th>Count</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>A solid javascript debugger</td>
<td>Count</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>15</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>7.4%</td>
<td>0.0%</td>
<td>3.7%</td>
<td>3.7%</td>
<td>7.4%</td>
<td>3.7%</td>
<td>7.4%</td>
<td>11.1%</td>
<td>15.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>A built in webserver component</td>
<td>Count</td>
<td>11</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>40.7%</td>
<td>14.8%</td>
<td>14.8%</td>
<td>7.4%</td>
<td>7.4%</td>
<td>0.0%</td>
<td>3.7%</td>
<td>3.7%</td>
<td>7.4%</td>
<td>100.0%</td>
</tr>
<tr>
<td>A built in database component</td>
<td>Count</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>44.4%</td>
<td>11.1%</td>
<td>11.1%</td>
<td>7.4%</td>
<td>11.1%</td>
<td>3.7%</td>
<td>3.7%</td>
<td>0.0%</td>
<td>7.4%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Built in Support for browser compatibility</td>
<td>Count</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>12</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>11.1%</td>
<td>0.0%</td>
<td>3.7%</td>
<td>3.7%</td>
<td>0.0%</td>
<td>7.4%</td>
<td>11.1%</td>
<td>7.4%</td>
<td>44.4%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Built in support for Browser Back and Forward buttons</td>
<td>Count</td>
<td>8</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>6</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>29.8%</td>
<td>0.0%</td>
<td>3.7%</td>
<td>3.7%</td>
<td>3.7%</td>
<td>3.7%</td>
<td>25.9%</td>
<td>7.4%</td>
<td>22.2%</td>
<td>100.0%</td>
</tr>
<tr>
<td>A Rich Drag and Drop GUI Interface</td>
<td>Count</td>
<td>9</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>33.3%</td>
<td>11.1%</td>
<td>7.4%</td>
<td>14.8%</td>
<td>7.4%</td>
<td>0.0%</td>
<td>3.7%</td>
<td>3.7%</td>
<td>18.5%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Built in features that handle security issues</td>
<td>Count</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>14.8%</td>
<td>3.7%</td>
<td>11.1%</td>
<td>3.7%</td>
<td>25.9%</td>
<td>7.4%</td>
<td>11.1%</td>
<td>0.0%</td>
<td>22.2%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Good Documentation</td>
<td>Count</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>15</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>3.7%</td>
<td>0.0%</td>
<td>3.7%</td>
<td>3.7%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>11.1%</td>
<td>22.2%</td>
<td>55.6%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Links/Associations with online or offline Developer Forums</td>
<td>Count</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>18.5%</td>
<td>3.7%</td>
<td>3.7%</td>
<td>14.8%</td>
<td>3.7%</td>
<td>3.7%</td>
<td>14.8%</td>
<td>7.4%</td>
<td>29.6%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Total</td>
<td>Count</td>
<td>55</td>
<td>12</td>
<td>17</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>25</td>
<td>17</td>
<td>71</td>
<td>243</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>22.6%</td>
<td>4.9%</td>
<td>7.0%</td>
<td>6.6%</td>
<td>8.2%</td>
<td>4.1%</td>
<td>16.3%</td>
<td>7.0%</td>
<td>29.2%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>
Next, developers were asked to rank the following IDE features: a JavaScript debugger, a built-in web server component, a built-in database component, support for browser compatibility, support for BACK/FORWARD buttons, A Rich Drag Drop GUI Interface, Security Handling, Good Documentation and associations with online/offline developer forums. It is clear that developers appreciate a solid JavaScript debugger, support for multi-browser deployment, good documentation and links to developer forums far more than built in web servers, database components, Back and Forward buttons (surprisingly) and fancy Drag and Drop UIs. (See Table 5) Most respondents were neutral or strongly in favor of built in features that handled security. When asked what features they would add to IDEs developers seemed most keen on stronger debugging tools, cross-browser compatibility inherent security handling, easy data submission functions and data validation reminders. Developers wanted to remove overused transition effects and extraneous features. Finally developers would favor a method of customizing their IDE. This flexibility would allow them to only download the parts of the IDE that they need to fulfill their job.

Happily, respondents rank minimal page refreshing, Browser Back and Forward buttons, immediate feedback, security, cross browser compatibility and effective management of server states as all MOST important features in applications that they develop. These developers place little importance on ensuring that links open in new windows however. (See Table 6). However, as much as developers want to include this functionality, only about half of these features are rated as easy to implement. Back and Forward buttons and cross browser compatibility were the rated the most difficult to implement in AJAX. (See Table 7).

QFeatureRanking_A: As a programmer, how important is it for the following features to appear in applications you develop? Please rank each of these features in order of importance to you by choosing a number for every row.
7 - MOST IMPORTANT
1 - LEAST IMPORTANT

<table>
<thead>
<tr>
<th>Feature</th>
<th>Count</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little/No Refreshing of the entire page when a user clicks on a link or button</td>
<td>Count</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>3.4%</td>
<td>13.6%</td>
<td>0.0%</td>
<td>10.3%</td>
<td>17.2%</td>
<td>20.7%</td>
<td>34.5%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Fully functioning Browser Back/Forward Buttons</td>
<td>Count</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>11</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>14.3%</td>
<td>7.1%</td>
<td>0.0%</td>
<td>7.1%</td>
<td>10.7%</td>
<td>21.4%</td>
<td>39.3%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Ability to open all links in new windows</td>
<td>Count</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>25.9%</td>
<td>7.4%</td>
<td>3.7%</td>
<td>11.1%</td>
<td>18.5%</td>
<td>14.0%</td>
<td>19.5%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Immediate feedback (some notification that tells the user what’s going on right after they press a button or a link)</td>
<td>Count</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>0.0%</td>
<td>6.9%</td>
<td>13.8%</td>
<td>3.4%</td>
<td>17.2%</td>
<td>27.6%</td>
<td>31.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>A Secure Application</td>
<td>Count</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>15</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>3.4%</td>
<td>0.0%</td>
<td>3.4%</td>
<td>3.4%</td>
<td>17.2%</td>
<td>20.7%</td>
<td>51.7%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Ability to work in all major browsers</td>
<td>Count</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>21</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>3.4%</td>
<td>24.1%</td>
<td>72.4%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Effective management of server and client states</td>
<td>Count</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>7</td>
<td>13</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>10.3%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>17.2%</td>
<td>24.1%</td>
<td>44.8%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Total</td>
<td>Count</td>
<td>16</td>
<td>10</td>
<td>6</td>
<td>11</td>
<td>29</td>
<td>44</td>
<td>84</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>% by Row</td>
<td>8.0%</td>
<td>5.0%</td>
<td>3.0%</td>
<td>5.5%</td>
<td>14.6%</td>
<td>22.0%</td>
<td>42.0%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

Table 6

Most developers recognized no novel security issues raised by AJAX. Many took the same precautions with their AJAX sites that they took in previous projects for example, client and server side data validation, and encryption. Most employed server side techniques to ensure application security.
Slightly more than half of the respondents felt that AJAX would become a more pervasive tool mostly because of the backing of major corporations like Microsoft and Google and benefits to user interaction. Other developers raised concerns about AJAX being unable to solve all current issues on the web and about AJAX being a fading trend. Surprisingly, more people though Flash would be more of a competitor to AJAX than XUL and XAML. Adobe Flex was cited another prominent competitor of AJAX. Developers had a very difficult time distinguishing DHTML from AJAX, granted the distinctions are quite limited.

Slightly more than half of the respondents felt that AJAX would become a more pervasive tool mostly because of the backing of major corporations like Microsoft and Google and benefits to user interaction. Other developers raised concerns about AJAX being unable to solve all current issues on the web and about AJAX being a fading trend. Surprisingly, more people though Flash would be more of a competitor to AJAX than XUL and XAML. Adobe Flex was cited another prominent competitor of AJAX. Developers had a very difficult time distinguishing DHTML from AJAX, granted the distinctions are quite limited.

Overall 43.3% of users are very satisfied with their experience as an AJAX Developer and 86.7% felt that of respondents that they will continue to use AJAX. No respondent indicated with certainty that they would forsake AJAX, while 4 respondents were not sure.

**User Experience**

**Demographics**

There were 99 legitimate responses to the user survey. Legitimate responses were non duplicate submissions which were either fully or partially completed responses to the survey. About 50 duplicate submissions were manually deleted from the response set resulting in the 99 responses reported in this paper.

Approximately 70% of the respondents were male while the majority of respondents, male and female were in the 20 – 22 age group (30%). The oldest respondents were in the 30 -34 age range (4.2%) while15 respondents were under 20 (15.6%). The survey indicated that the majority of users access their Gmail account using a laptop (60%) while the vast majority of the respondents run Windows XP Professional or XP Home on their machines. Despite this fact, about 20% more users check their GMAIL in Firefox than in Internet Explorer. 70.8% of users enjoy CABLE/DSL as their connection speed while screen resolutions were split evenly between 1024 x 768 and 1280 x 1024 (32.3 %). Most participants were a part of either engineering or

**Table 7**

<table>
<thead>
<tr>
<th>QDifficultyRanking_A: Please rank these features according to how hard they are to implement with AJAX both in terms of complexity and time to implement.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>16</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>29</td>
</tr>
<tr>
<td>% by Row</td>
<td>55.2%</td>
<td>10.3%</td>
<td>6.9%</td>
<td>6.9%</td>
<td>6.9%</td>
<td>6.9%</td>
<td>6.9%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Count</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>29</td>
</tr>
<tr>
<td>% by Row</td>
<td>13.8%</td>
<td>10.3%</td>
<td>6.9%</td>
<td>17.2%</td>
<td>10.3%</td>
<td>10.3%</td>
<td>31.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Count</td>
<td>10</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>29</td>
</tr>
<tr>
<td>% by Row</td>
<td>35.7%</td>
<td>7.1%</td>
<td>10.7%</td>
<td>10.7%</td>
<td>14.3%</td>
<td>14.3%</td>
<td>7.1%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Count</td>
<td>14</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>29</td>
</tr>
<tr>
<td>% by Row</td>
<td>48.3%</td>
<td>10.3%</td>
<td>3.4%</td>
<td>13.8%</td>
<td>13.8%</td>
<td>3.4%</td>
<td>6.9%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Count</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>29</td>
</tr>
<tr>
<td>% by Row</td>
<td>10.3%</td>
<td>10.3%</td>
<td>17.2%</td>
<td>27.6%</td>
<td>6.9%</td>
<td>13.8%</td>
<td>13.8%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Count</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>29</td>
</tr>
<tr>
<td>% by Row</td>
<td>13.8%</td>
<td>13.8%</td>
<td>10.3%</td>
<td>20.7%</td>
<td>6.9%</td>
<td>6.9%</td>
<td>27.6%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Count</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>29</td>
</tr>
<tr>
<td>% by Row</td>
<td>17.2%</td>
<td>13.8%</td>
<td>6.9%</td>
<td>27.6%</td>
<td>17.2%</td>
<td>6.9%</td>
<td>10.3%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Count</td>
<td>56</td>
<td>22</td>
<td>18</td>
<td>34</td>
<td>22</td>
<td>16</td>
<td>30</td>
<td>202</td>
</tr>
<tr>
<td>% by Row</td>
<td>27.7%</td>
<td>10.9%</td>
<td>8.9%</td>
<td>17.8%</td>
<td>10.9%</td>
<td>8.9%</td>
<td>14.9%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>
sciences field of study or profession (86.9%) and most users describe themselves as advanced users of the internet (45.8%).

Page Refreshing Preferences

The survey asked respondents to indicate whether they preferred websites (in general, not just within GMAIL) that refreshed the entire page, or parts of the page. A decisive majority, 58.3% of the 96 users who answered this question indicated that they preferred websites that refreshed parts of the page only (See Figure 1 Refreshing the Entire Page – Number of users in each response option).

These users cited improved response time, aesthetic benefits, lack of interruption, lack of wasted resources, similarity to desktop applications, lack of information loss, reduced CPU time, easy access to persistent information and reduced time for image loading as reasons for their preference for refreshing only parts of the page. The most popular reasons for this choice were faster loading times and conservation of resources. Next, users appreciated that refreshing only parts of the page made applications look nicer and feel more intuitive (like desktop applications). See Appendix A for this definition. 30% of participants indicated that they had no preference, while the minority believed that the whole page should be refreshed because of easier management of back and forward buttons, desire to see fresh data, fear of missing information, desire to open links in a new tab, and messy linking.

Users were given a question in table format, asking them to rate how often they used BACK and FORWARD browser buttons and how important these features were to them. Many users did not comprehend this tabular format and submitted responses in every row (see Appendix A to view the exact presentation of question QBrowserNavigation_A). Due to this, no conclusive deduction can be made on how often users use the Back and Forward Buttons but it is clear from the results that most value the functionality very highly. (See Table 8)
Table 8

QBrowserNavigation_A: How often do you use the FORWARD and BACK buttons in your browser and how important are these features to you? [Please choose an option from only ONE row]

1 - I can easily live without this feature
3 - I am neither for nor against these features
5 - I can't live without these features

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Often</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>28</td>
<td>47</td>
<td>85</td>
</tr>
<tr>
<td>% by Row</td>
<td>3.5%</td>
<td>2.4%</td>
<td>5.9%</td>
<td>32.9%</td>
<td>55.3%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Occasionally</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>10</td>
<td>29</td>
</tr>
<tr>
<td>% by Row</td>
<td>10.3%</td>
<td>6.8%</td>
<td>24.1%</td>
<td>24.1%</td>
<td>34.5%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Rarely</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>% by Row</td>
<td>21.4%</td>
<td>0.0%</td>
<td>35.7%</td>
<td>14.3%</td>
<td>28.6%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Not at all</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>% by Row</td>
<td>9.1%</td>
<td>9.1%</td>
<td>18.2%</td>
<td>18.2%</td>
<td>45.5%</td>
<td>100.0%</td>
</tr>
<tr>
<td>My browser does not have BACK and FORWARD buttons</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>% by Row</td>
<td>27.3%</td>
<td>18.2%</td>
<td>18.2%</td>
<td>0.0%</td>
<td>36.4%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Total</td>
<td>13</td>
<td>7</td>
<td>21</td>
<td>39</td>
<td>79</td>
<td>150</td>
</tr>
<tr>
<td>% by Row</td>
<td>8.7%</td>
<td>4.7%</td>
<td>14.0%</td>
<td>26.0%</td>
<td>48.7%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

Table 9

The following questions asking users to give similar opinions on opening messages in new windows suffered from a similar problem due to the tabular format. However, it is clear here as well that most users found this feature extremely important to them. (See Table 9)

Gmail Use

Most survey participants used MSN Hotmail in addition to GMAIL (73.7%). Others used, a mail system provided by a university or business, Yahoo Mail, Outlook, Eudora, AOL, Netscape and others, and ( in order of number of responses per email client). Most respondents
were happily very familiar with GMAIL: 56.8% of them have been using the client for over a year while 88.3% have used GMAIL for over 6 months.

Respondents were asked what times of the day they checked their GMAIL and for how long. The question was in tabular format but users were allowed to respond in multiple rows (see Appendix A question labeled QTimeOfUse) so, unlike earlier questions, there was less confusion. One can conclude that most participants check their email for a few minutes only at almost all times of the day. 44.2% of participants were preoccupied with something else when checking their GMAIL.

**Gmail Experience**

A decisive majority of survey takers caught on very quickly to GMAIL, taking an hour or less to get used to its features (72.4%). Of the 87 persons who chose to answer this question, only 3 took more than a week to get acclimated to GMAIL.

Users noted that Google talk, the star button, the delete email feature, GMAIL’s label rather than folders, the conversation view, keyboard shortcuts, and lack of sorting mail as the hardest features to get used to in GMAIL. The overwhelming majority noted that the absence of folders in favor of labels and the awkward delete message functionality we far more frustrating than the others mentioned above. Of the above, it is probable that Google TALK, the star button, labels, conversations and the delete function are made possible at least in part by AJAX. GMAIL Talk is Google’s instant messenger client which is built into GMAIL. In GMAIL, users are allowed to label messages and not store them in actual folders. This means that messages remain in the inbox with a simple tag when users choose to apply it. The ‘star’ button in GMAIL let’s you mark a message with a gold star much in the same way you can tag messages with labels. The star can hold any significance the user wishes. Users are allowed to select multiple messages from the inbox and click on a delete button in order to remove messages.

On the other hand, the easiest features for users to understand were conversations. User praised the condense/expand functions, and the nested replies. One users acknowledged the fact that within the conversation view, users are not warned about the subject of the continuing thread (called a conversation in GMAIL), leading on some occasions to users sending replies with the wrong subject line. Search, the spellchecker, auto suggestion of email addresses when composing a message, sending emails with attachments, starred messages and archiving were all other pleasing to users. The number of respondents rating the latter two as easy to learn was less than the number finding it difficult to grasp. All of the above features have their roots in AJAX.

In general the list of features that participants liked was longer that that of those they found difficult to master. The latter list showed more conversion on a few key problem areas like labels and the delete function.

The majority of survey respondents felt that GMAIL’s UI Design and conversation view distinguished GMAIL from other applications and were among its best features. They described the interface as easy to use, user friendly and responsive. Auto saving of drafts, auto suggesting email addresses, auto refreshing the inbox were also the best distinguishing features according to respondents. On the contrary, many users called the interface boring, cramped and cluttered. The use of AJAX coupled with the use of labels in the inbox severely frustrated users by crowding out useful information on the page. Bas design and structuring of information can never be solved by the benefits of AJAX. Google CHAT, which must use AJAX extensively, was listed equally as a best and worst feature by users as a whole but users seemed to be discontent with GMAIL Chat as a concept and not the functionality itself.
Users were asked to rank features found both in general websites and in GMAIL specifically according to personal importance. For minimal page refreshing, 23.2% of respondents were a little above neutral, weighting it a ‘5’. (Mean is 4.21). About half the respondents indicated that dynamic suggestions in the To, CC and BCC lines are most important to them (mean is 5.83). Auto updating the GMAIL contact list within the GMAIL Chat widget is a very low priority for respondents (mean 3.61). Browser Back and Forward button functionality, opening messages in new windows, immediate feedback and desktop like features were all rated most important to participants (means are 4.87, 4.49, 5.17 and 4.63 respectively). In general, one can say that users as a whole weighed, dynamic email suggestions, browser back and forward buttons, ability to open links in new windows, immediate feedback and desktop-like features as most important to them, well above little or no page refreshing and the auto updating of GMAIL Chat.

QFeatureRanking_A: Please rank each of these features in order of importance to you?
PLease choose a number for every row.
7 - MOST IMPORTANT
1 - LEAST IMPORTANT

<table>
<thead>
<tr>
<th>Little/No Refreshing of the entire page when you click on a link or button</th>
<th>Count</th>
<th>% by Row</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13</td>
<td>15.9%</td>
<td>8</td>
<td>7</td>
<td>11</td>
<td>19</td>
<td>12</td>
<td>12</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>Quickly listing possible email addresses in a dropdown box while you are typing in the “To”, “Cc” or “Bcc” fields.</td>
<td>3</td>
<td>3.6%</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>18</td>
<td>42</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>Auto-updating your Gmail Chat contact list</td>
<td>18</td>
<td>22.0%</td>
<td>9</td>
<td>15</td>
<td>13</td>
<td>10</td>
<td>6</td>
<td>11</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>Browser’s Back/Forward Buttons</td>
<td>5</td>
<td>6.1%</td>
<td>5</td>
<td>9</td>
<td>17</td>
<td>10</td>
<td>13</td>
<td>23</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>Ability to open messages or links in new windows</td>
<td>11</td>
<td>13.4%</td>
<td>6</td>
<td>7</td>
<td>13</td>
<td>15</td>
<td>13</td>
<td>17</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>Immediate feedback (some notification that tells you what’s going on right after you press a button or a link)</td>
<td>3</td>
<td>3.7%</td>
<td>6</td>
<td>5</td>
<td>10</td>
<td>16</td>
<td>20</td>
<td>22</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>Desktop application-like features in web applications</td>
<td>7</td>
<td>8.5%</td>
<td>8</td>
<td>5</td>
<td>17</td>
<td>13</td>
<td>13</td>
<td>18</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>60</td>
<td>10.5%</td>
<td>45</td>
<td>50</td>
<td>89</td>
<td>90</td>
<td>95</td>
<td>145</td>
<td>574</td>
<td></td>
</tr>
</tbody>
</table>

Table 10 – Users’ Ranking of website and GMAIL features that are affected by AJAX

According to participants checking email feels faster in GMAIL (57.3%). 36.6% feel that that load times are about the same while a handful think other email clients load faster. Users don’t usually have to wait longer than 5 seconds for a GMAIL page to load. Out of 81 valid responses to this question, only 10 could remember having to wait over one minute for a GMAIL page to load.

Two tabular questions asking whether users noticed erratic BACK and FORWARD button functionality and whether they noticed a constant refreshing of the entire page both within GMAIL suffered from the afore mentioned problems (the table format confused the users). One can conclude however that though most of these respondents clearly indicate that they find BACK and FORWARD functionality important to them in addition to minimal page refreshing, their feelings change to neutral on both counts when applied to GMAIL. This may indicate that GMAIL’s rich features satisfy the users so much that broken browser buttons become less
important to them. Less than a handful of participants noted the broken BACK and FORWARD buttons as worst features of GMAIL which suggests that this may be a flaw they are willing to forgive in light of a clean and responsive interface.

Figure 2 – Users’ evaluation of GMAIL’s Desktop-Like Features

Quite surprisingly, 47.3% - almost half - of survey takers recognized no desktop-like features in GMAIL. This who recognized at least a few desktop-like features listed the following:

- Auto-saving
- Dynamic contact suggestions
- The GMAIL messenger feature
- Keyboard shortcuts

- Quick loading times
- One user felt that without the browser, GMAIL would essentially work exactly like a desktop application
- Lack of page refreshing
- Instant feedback
- Good Interface design
- Storage space
- The ability of the interface to keep up with user activity

The pie chart in Figure 2 – Users’ evaluation of GMAIL’s Desktop-Like Features however, may not be so astonishing in light of the following. Users felt that the following changes would make GMAIL behave more like a desktop application (in order of frequency):

- Drag and Dropping of messages and elements
- Interactive calendar functionality and/or integration with mini applications inside and outside of the browser.
- The introduction of folders
- Spellchecking
- Accessing GMAIL from other programs
- Less constant refreshing
- More feedback about browser activity i.e. more than a “Loading” Label

From the above, it is clear that Google’s GMAIL lacks a lot of desktop application features that users identify with. In general, it seems like the traits users identify with desktop
applications are features closely related to storage of a component (in this case messages whereas on a desktop it would be files) in a well defined space. They want to manipulate that content i.e. drag and drop them from folder to folder (well defined spaces) and they want the freedom to validate (spell-check) and save that content either remotely (in GMAIL) or on their machines. Integration with other applications both on the web and on the computer is an important hint to these users that an application displays desktop-like features. Finally, users recognize that desktop apps usually give more feedback about the results of a user action than web applications: in addition to a “loading”, “working” or similar label, the mouse icon changes, CPU usage goes up, interface elements change more quickly and users are given more detailed status information. The fact that one user mentioned less constant refreshing may suggest that auto refreshing may even be distracting and frustrating to some users. Even if desktop applications make use of auto-refreshing, it is never as obvious and cumbersome to the user as it is on the web.

In GMAIL, the use of AJAX disables opening certain links in new windows. The majority of respondents found this neither pleasing nor displeasing (40.3%) and a significant 18.2 % didn’t even know the feature was absent. However, slightly more respondents found this behavior displeasing than pleasing (a 2.6% margin).

In certain sites supported by AJAX, for example live.com, CPU Usage will dramatically increase once the user begins to interact with the web page. According to these participants, GMAIL does not suffer from that problem. 63.3% of respondents have never experienced this behavior while only 7 respondents (8.9%) remembered CPU stress during their GMAIL use.

The final three sections of the user survey asked what features they would add to, remove from or change in GMAIL. The responses to these questions did not supplement or contradict those mentioned above, but rather reiterated sentiments afore mentioned.

## Conclusions and Recommendations

This survey suggests that the overwhelming majority of AJAX developers are men who tend to be more skilled in AJAX than their female counterparts. Though computer science as a whole is suffering from low percentages of women, AJAX is below the computer science average (3/56 respondents to this survey were women or about 5%). AJAX has not yet attained the maturity of other areas of Computer Science but current developers need to address the issue of diversity of thought and gender within the AJAX developer community if AJAX is to become prominent. According to various university studies, 25 % of developers on a whole are women, but 2% of developers on open source projects are women. 14 AJAX is a mélange of various open source technologies and my fear is that these low percentages may become even lower without initiatives by current developers to attract and retain women such as focus groups, high school programs, strong mentorship etc.

AJAX Developers covers all age ranges but are especially prolific in their 20s. This may suggest that many developers are young, open to new ideas and ready to populate the work force should a high demand for AJAX developers arise. The life of AJAX is undoubtedly online. Expensive AJAX Developer conferences, AJAX technical books and manuals etc did not contribute to developer awareness of AJAX or help in learning AJAX as much as online tutorials and blogs. In fact, the accepted one stop spring board site for becoming familiar with AJAX is a

http://www.computerworld.com/developmenttopics/development/story/0,10801,103777,00.html
blog called ajaxian.com. This suggests that the best way to influence the AJAX community is online therefore, initiatives that target more women can be launched from such blogs in addition to maintaining and increasing the level of tutorial quality that exists now.

AJAX developers found it extremely hard to compare AJAX with other technologies they have used simply because they build in these technologies into AJAX. The respondents who found AJAX to be easiest were those who had proficiency with JavaScript and/or a scripting language they decided to use with JavaScript itself. This indicates that AJAX, even though, it’s strict definition is the use of asynchronous JavaScript with XML, is being used to refer to any asynchronous communication with the server coupled with the use of another technology to retrieve and present data. This approach has its advantages and disadvantages. Firstly, developers are becoming more skilled in more technologies through their efforts to become skilled in AJAX. However, this could mean that learning is prolonged because more technologies need to be understood to make AJAX work. In addition, it may cause extreme confusion for beginners in light of browser reactions to the various technologies being mixed. Seasoned programmers may find themselves spread too thin as their skills need to grow on all fronts with the growing complexity of AJAX applications. This mélange of web technology use that tends to be called AJAX needs to be consolidated into a standard. OPEN AJAX has already begun this process in hopes that AJAX will soon become a technology that you can learn once, use in one development environment and rely on to execute flawlessly in multiple browsers. These changes would represent immense improvements to the lives of current and future developers.

One big part of developer experience is the quality of the development environment that is used but the results of this survey suggest that while many libraries and tools kits boast different strengths, they are, in general, poorly documented and out of sync with the needs of the developers. For this reason, most developers simply write code around the XMLHttpRequest Object directly without any additional tools or, in some cases even write their own IDEs! Current AJAX IDEs are too bulky and lack far too much documentation. It seems as though, in some cases, the documentation is excessive when considering the amount of code needed while in others, it is insufficient information to understand the package. A casual survey administered through ajaxian.com has confirmed the results of this survey. Developers were asked to choose which environments they use, and 40% noted that they do not use any helper at all. (763 developers participated in this survey). AJAX IDEs are hard to use and complicate the lives of a significant amount of AJAX Programmers.

It is important to consider that a lot of usability issues on the web result from the fact that the programmer experience is bad. If the programmer has to deal with unusable tools, he/she may not take the effort to make a usable experience for the user. Toolkits like Dojo seem to be attacking the usability problem from the angle of simplifying life for the programmer a formidable task. Harder than hiding problems from the user, is hiding problems for the programmer such as cross browser idiosyncrasies.

Therefore, to improve developer experience, they need a way to customize downloads of these packages so that they can make use of features that are useful to them only. Should the package have an extensive feature set, documentation needs to be commensurate. Bad IDEs are only throwing cogs in the wheels of AJAX learning especially when it comes to beginners. It may even be creative to present an AJAX IDE that executes in the browser itself. Such an IDE can help programmers debug faster and grow in their understanding of desktop application features within web applications.

15 Information about this survey can be found at http://ajaxian.com/by/topic/survey/
Developers who took this survey did not want necessarily a fancy interface with impressive graphics. They rated JavaScript debuggers and cross browser compatibility (sophisticated fail-over techniques that adapted the page structure according to how old the browser is.) as the most important features that they look for. In fact, it was interesting to note that though developers found minimal page refreshing, immediate feedback, browser compatibility, and back and forward button functionality as being very important to them, noting the latter two were the most difficult to implement, very few wanted built in support for back and forward functionality in their IDE. This could be because most developers simply don’t use IDEs or it could reflect a certain nonchalance on the part of developers regarding this issue. Built in rectification of this problem should be built into IDEs as it is clear from the results of this survey that most users rely on this functionality. If developers treat this as a secondary concern, users may become less and less enamored with AJAX sites. The same statistics apply to opening messages in new windows. Users rely on the feature, while developers hardly care. That said, when asked if they were frustrated by the broken BACK and FORWARD buttons in GMAIL, users were mostly neutral. This suggests that if developers are forced to ignore or postpone these issues, the feature set of exemplary caliber may be able to appease users who would have otherwise been annoyed. The motto for developers however should be to fix the back and forward buttons. If AJAX is to shape the future of the web and contribute to the GUIsation of web applications, these buttons must work. In almost every desktop application, a user’s action is undoable or reversible to some extent. Users have come to expect this functionality to be present and without it, might feel lost within an application and distrustful of sites where they cannot predict when these buttons will/will not work. Therefore to lead web applications towards the kind of stability that desktop software offers, it is paramount that these browser buttons be fully functional in all AJAX Apps. Perhaps, even future browsers can cache the states of the structure of web pages and manage back and forward functionality natively. That said, developers should be given the option of temporarily disabling the buttons because it is probable that in certain situations, that kind of control would be needed to protect the user from losing information. Of course, all of the above should be done with adequate and prominent feedback to the user.

The survey onajaxian.com also showed that most programmers (40.2%) use AJAX in conjunction with PHP (a conclusion reported by this survey as well) and 38.2% use AJAX in a development setting (rather than in production or for prototyping purposes). Most developers felt that AJAX spelt rest for the server but since most participants in this survey pushed their security handling to the server side in their AJAX applications, this assumption might not be true. The server may be under even more stress due to security handling.

Since no respondent indicated that they would completely abandon AJAX, it is safe to say that developers are on the whole very optimistic about its future potential. 43.3% of respondents were very satisfied with their experience and feel that with the support of major corporations, AJAX has a bright future.

On the flip side of the coin, users are recognizing that it is better to update parts of a page only. Perhaps, it is now safe to say they are now beginning to expect it in their favorite sites and move away from the entire page reload standard that was common even up to 3 years ago. This indicates a promising future for AJAX as it easily provides this functionality.

One very important observation that was made was that the vast majority of users felt that GMAIL possessed no desktop-like features at all. This indicates that users in general do NOT
make the connection between some of the advantages of AJAX and certain desktop traits. Users recognize the following as prominent desktop traits:

- Some kind of concept of data (either messages or files) and the storage, personalization, saving and retrieving of that data.
- Linked with the above, the user’s focus is on the manipulation of the object (an email message or a file) not necessarily application related components such as browsers.
- To a lesser extent, quick response times are associated with desktop software.
- Searching and sorting of the above objects is synonymous with a desktop application to users who took this survey.

From the above, it is clear that Google’s GMAIL lacks a lot of desktop application features that users identify with. In general, I suggest that web applications have a long way to go to convince the user that they can behave like desktop applications for the following reasons:  

- **Devices** – On the web, we still can never predict who will access our website with what machine, browser, operating system, connection speed, screen resolution etc. The variables are enormous and unlike desktop applications, web applications have to adapt themselves to the host machine. This is still a major problem that standards like OPEN AJAX are trying to address.

- **User Focus** – It may take a while for users to change from thinking of web surfing as moving between pages of information to viewing and manipulating data elements. Perhaps if objects, like mail messages, were stored online in the way that applications are stored in a file system on your computer, users would make the connection. In Gmail, you are never interacting with objects directly (example, drag and dropping messages). In general, the objects are hidden and you interact with representations of the information within the objects.

- **User Tasks** – Users usually use desktop applications for longer periods of time than web applications because there is simply more flexibility and more control. If web applications offered more control of data, users would spend more time using their web applications and perhaps view them as being as stable as desktop applications.

- **User’s Conceptual Space** – On desktops, users operate in a defined space. On the web, there is no defined area of operation, just pages of information. This is why many respondents insisted that the introduction of folders would give them more of a desktop feel.

- **Presentation Elements** – A desktop app has many transient, dynamically appearing and disappearing components, much of which AJAX can do for web applications. With AJAX, there are no longer two components to the website: the browser and the page. There are now the browser the page, and dynamic objects within the page, which users can control. AJAX is well on the way to aiding in this category.

- **Context** – As mentioned above, desktop applications are good at providing context while web application has a ways to go in terms of giving the user a better understanding of their workspace (for example with the use of folders).

- **Response Time** – No matter how prolific the use of AJAX is within web pages, web applications still depend on the internet connection and by extension, network activity. On the internet, the response time can *seem* instantaneous since the whole page may not

---

16 Galitz, pg 30-32, makes a comparison between desktop and web applications using the categories in this section.
be refreshed but times can still vary quite a bit from computer to computer and internet connection to internet connection

- **Visual Style** – In this category, we applications are becoming more and more like desktop applications because users are now given more control over the visual style of their pages.
- **Security** – websites will always introduce more security issues than desktop applications.

Some of the above issues can never be resolved but it is clear that web applications can do some more work on the transforming the behavior of web applications to match that of desktop software.

At the beginning of this study, I expected to find that every advantage that AJAX brought to the user spelt complexity for the programmer as he/she attempted to design an application that reverses the paradigm of traditional web programs However, I have found that on both fronts, there are significant gains and losses.

**Final Comments**

While this survey has made some important conclusions, a good extension of this study would be to examine the experience of AJAX on female developers and to probe deeper into the user satisfaction of AJAX websites. Internet based surveys would be the clear method of gathering data since the life of the AJAX developer and user is online. Surveys would be shorter (a maximum of 5 questions and could be launched occasionally on ajaxian.com (for developers), the spring board site for most beginners. It would also be helpful to distribute both user and developer surveys to a large pool of respondents and compare those results to the snapshots gathered in this paper.

There is no doubt that AJAX is going to become even more prominent, having proved its metal. Other technologies like DHTML are seen as a stepping stone to AJAX while its competitors are in many ways either too specialized, too bulky and are not open source. That said, the initial hype is over. Improved experiences for all stakeholders must start with the developer as he/she balances the advantages of AJAX with what users perceive as useful to them. This study also proves that the success of a website is depends on how good the design is rather than how much animation is present (whether provided by AJAX or not). Take the star functionality in GMAIL as an example. It is probable that GMAIL developers update the server asynchronously when messages are “starred”. However, the way in which the functionality is presented to the user promotes confusion. All developers should live in a balance between exhilarating features and good design.

The future of web programming is the intermingling of web technologies, offsetting the disadvantages of some technologies by harnessing the strengths of another. With accepted standards and a more open ear to the user community, the AJAX developer and user experience can vastly increase in the coming years.
Appendix A – Survey Questions

User Survey on GMAIL and AJAX

Basic Information

Gender. What is your sex?
☐ Female
☐ Male

Age. How old are you?
☐ Under 20 years
☐ 20 - 22
☐ 23 - 25
☐ 25 - 29
☐ 30 - 34
☐ 35 - 39
☐ 40 - 44
☐ 45 - 49
☐ 50 - 54
☐ 55 - 59
☐ 60 - 64
☐ 65 or older

Machine. What type of machine do you normally use to check your Gmail?
☐ Desktop
☐ Laptop
☐ Tablet PC
☐ PDA
☐ Other (please specify) ____________________________

Platform. What operating system runs on that machine?
☐ DOS
☐ OS2
☐ OS8

17 All surveys were hosted through Perseus Survey Solutions at http://express.perseus.com
<table>
<thead>
<tr>
<th>OS9</th>
<th>OSX</th>
<th>Unix</th>
<th>PC Running UNIX</th>
<th>Windows 95</th>
<th>Windows 98</th>
<th>Windows ME</th>
<th>Windows NT</th>
<th>Windows 2000</th>
<th>Windows XP Home</th>
<th>Windows XP Professional</th>
<th>Terminal/vt100</th>
<th>I don't know</th>
<th>Other (please specify)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser. What browser do you normally use to check your Gmail?</td>
<td></td>
<td></td>
<td></td>
<td>Microsoft Internet Explorer</td>
<td>Netscape Navigator</td>
<td>Mozilla Firefox</td>
<td>Apple Safari</td>
<td>Other (please specify)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ConnectionSpeed. What is your internet connection speed?</td>
<td></td>
<td></td>
<td></td>
<td>T2 or better</td>
<td>Cable/DSL</td>
<td>ISDN</td>
<td>56k dial-up</td>
<td>28.8k dial-up</td>
<td>Other (please specify)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ScreenResolution. What is the screen resolution on that machine?</td>
<td></td>
<td></td>
<td></td>
<td>1024x768</td>
<td>1280x1024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Field. What field of study or profession are you a part of? (Check all that apply)

- Engineering
- Sciences
- Business
- Humanities
- Medicine
- Law
- Hospitality
- Other (please specify)

Proficiency. How would you describe your level of proficiency with using the internet?

- Beginner
- Intermediate
- Advanced
- Professional

General Questions About Your Web Usage

Desktop Applications are software programs that are downloaded from the web or installed directly from a CD. They run on your desktop or laptop and do not need the internet to be able to function properly, for instance, Microsoft Word, Adobe Photoshop etc.

Web Applications on the other hand, are programs that run in a browser. For example, MSN hotmail, cnn.com etc.

Desktop Application. In your opinion, what is (are) the major difference(s) between desktop applications and web applications?

RefreshPage. Some websites will refresh an entire page when you click on a link. Others opt to only change parts of a page when you click on a link. Which do you prefer and why?
Websites should refresh the whole page because:

Websites should only refresh parts of a page because:

I have no preference.

**Browser Navigation.** How often do you use the FORWARD and BACK buttons in your browser and how important are these features to you? **[Please choose an option from only ONE row]**

1 - I can easily live without this feature
3 - I am neither for nor against these features
5 - I can't live without these features

<table>
<thead>
<tr>
<th>Frequency</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Often</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Occasionally</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rarely</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not at all</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>My browser does not have BACK and FORWARD buttons</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Windows.** How often do you need to open a link in a new window and how important is this feature to you? **[Please choose a number from only one row]**

1 - I can easily live without this feature
3 - I am neither for nor against these features
5 - I can't live without these features

<table>
<thead>
<tr>
<th>Frequency</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Often</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Occasionally</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rarely</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not At All</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Gmail Use**

**Other Mail Systems.** Other Than Gmail, what mail systems have you used?

- [ ] MSN Hotmail
- [ ] Yahoo Mail
- [ ] Outlook
- [ ] Eudora
A Mail System provided by a university or business  
Other (please specify)  

**LengthOfUse.** For approximately how long have you been using Gmail?  
☐ A month or less  
☐ 1-2 months  
☐ 3-5 months  
☐ 6-8 months  
☐ 9-11 months  
☐ About a year  
☐ Over a year  

**TimeofUse.** In general, at what time(s) of the day do you usually check your GMAIL account and for how long? [please respond in one or more rows]  

<table>
<thead>
<tr>
<th>Time of Day</th>
<th>Not at this time</th>
<th>A few seconds only</th>
<th>A few minutes</th>
<th>15 mins to 30 mins</th>
<th>30 mins to 1 hour</th>
<th>Over an hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early Morning</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Mid Morning</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Early Afternoon</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Mid Afternoon</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Early Evening</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Late at Night</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>At all times of the day</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

**Busy.** How busy are you when you check your email at these times?  
☐ I am always busy when checking my GMAIL  
☐ I am frequently busy doing something else while checking my GMAIL  
☐ I am busy when checking my GMAIL only about half the time  
☐ I frequently check my GMAIL when I'm **NOT** busy  
☐ I am never busy when I check my GMAIL  

**GMAIL Experience**  
For the following questions, please note that a GMAIL "feature" is defined as a special task it enables you to do. It does not have to be a formal tool or service within GMAIL. It could also be a particular behavior in GMAIL that you've found new, interesting or boring, or easy or difficult to learn.
**Familiarity.** How long did it take you to get used to Gmail's features?
- An hour or less
- A couple of hours or less
- A few days but under a week
- About a week
- Several weeks
- More than several weeks

**Familiarity1.** Which Gmail feature was hardest for you to use?

**Familiarity2.** Which Gmail feature was easiest for you to use?

**Distinction.** In your opinion, what distinguishes GMAIL from other email systems that you have used, for example Yahoo Mail?

**BestFeature.** In your opinion, what is Gmail's best feature?

**WorstFeature.** In your opinion, what is Gmail's worst feature?

**FeatureRanking.** Please rank each of these features in order of importance to you?
Please choose a number for every row.
- 7 - MOST IMPORTANT
- 1 - LEAST IMPORTANT

<table>
<thead>
<tr>
<th>Feature</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little/No Refreshing of the entire page when you click on a link or button</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quickly listing possible email addresses in a dropdown box while you are typing in the &quot;To&quot;, &quot;Cc&quot; or &quot;Bcc&quot; fields.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auto-updating your Gmail Chat contact list</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browser's Back/Forward Buttons</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ability to open messages or links in new windows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Immediate feedback (some notification that tells you what's going on right after you press a button or a link)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Desktop application-like features in web applications</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ResponseTime.** Do you feel that you can check your email in GMAIL faster than with other email systems (in terms of time taken to load pages and messages)?
Checking email feels faster with Gmail
Page load times are about the same
Page load times seem faster in other mail systems

**PageLoading.** On the machine you usually use to access your GMAIL, what was the longest period of time you’ve had to wait for a GMAIL page to load?

- Less than 1 second
- 1 to 5 seconds
- 5 to 15 seconds
- 15 to 30 seconds
- 30 to 60 seconds
- 1 to 5 minutes
- Over 5 minutes

**BrowserButtons.** Have you noticed that the BACK and FORWARD Buttons behave differently in GMAIL? Has this been a pleasing or frustrating experience for you?

NB: Behaving differently means that clicking BACK or FORWARD does not take you to the previous page you were at and/or suddenly become unavailable.

[Please choose an option from only **ONE** row]

<table>
<thead>
<tr>
<th></th>
<th>Highly Frustrating</th>
<th>Mildly Frustrating</th>
<th>Neither Frustrating nor Pleasing</th>
<th>Mildly Pleasing</th>
<th>Highly Pleasing</th>
</tr>
</thead>
<tbody>
<tr>
<td>These buttons Always behave differently</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>These buttons Frequently behave differently</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>These buttons Ocassionally behave differently</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>These buttons Rarely behave differently</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>These buttons Never behave differently</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>My browser does not have Back and Forward Buttons</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PageRefresh.** Do you notice GMAIL refreshing the entire page when you click on links and do you find this pleasing or frustrating?

[Please choose an option from only **ONE** row]

<table>
<thead>
<tr>
<th></th>
<th>Highly Frustrating</th>
<th>Mildly Frustrating</th>
<th>Neither Frustrating nor Pleasing</th>
<th>Mildly Pleasing</th>
<th>Highly Pleasing</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMAIL always refreshes the entire page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMAIL refreshes the entire page frequently but not always</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Highly Frustrating, Mildly Frustrating, Neither Frustrating nor Pleasing, Mildly Pleasing, Highly Pleasing

GMAIL refreshes the entire page occasionally
GMAIL rarely reloads the whole page
GMAIL never refreshes the whole page

Desktop Applications are software programs that are downloaded from the web or installed directly from a CD. They run on your desktop or laptop and do not need the internet to be able to function properly, for instance, Microsoft Word, Adobe Photoshop etc.

Web Applications on the other hand, are programs that run in a browser. For example, MSN hotmail, cnn.com etc.

**DesktopTraits.** In your opinion, does GMAIL behave more like a desktop application than other email systems and if so, what desktop application-like features does it display? Are these behaviors useful?

- [ ] Yes, GMAIL behaves very much like a desktop application. My explanation:

- [ ] GMAIL exhibits a few desktop-like features. My explanation:

- [ ] I don't think GMAIL displays any desktop-like features

**DesktopFeatures2.** If you could add a feature(s) to GMAIL that could make it behave more like a desktop application, what would it(they) be?

**NewWindows.** When you right click on a message in your GMAIL inbox, you usually do not have the option to open that message in a new window. How do you feel about the absence of this feature?

- [ ] Very happy - If I really needed to, I could click on the message then click on an icon to open in a new window
- [ ] Happy
- [ ] Neither Happy Nor Unhappy
- [ ] Unhappy
- [ ] Very Unhappy - I have to leave my inbox to open a message in a new window
- [ ] I have never noticed that this feature was absent

**CPUusage.** Has navigating to GMAIL ever caused the CPU usage on your machine to increase suddenly and significantly?

35
Final Comments

**Additions.** What features would you add to GMAIL and why? Please think carefully about this question.

**Removals.** What features would you remove from GMAIL and why? Please think carefully about this question.

**Changes.** Perhaps you wouldn't like to fully remove some features, just tweak them. What Gmail features would you change and why? Please think carefully about this question.
**Developer Survey on AJAX**

**Basic Information**

**Gender.** What is your gender?
- [ ] Female
- [ ] Male

**Age.** How old are you?
- [ ] Under 20 years
- [ ] 20 - 22
- [ ] 23 - 25
- [ ] 25 - 29
- [ ] 30 - 34
- [ ] 35 - 39
- [ ] 40 - 44
- [ ] 45 - 49
- [ ] 50 - 54
- [ ] 55 - 59
- [ ] 60 - 64
- [ ] 65 or older

**Proficiency.** How would you describe your level of proficiency with programming in AJAX?
- [ ] Beginner (have not implemented a full scale AJAX system but have programmed small scale AJAX pages)
- [ ] Intermediate (implemented 1-3 systems)
- [ ] Advanced (implemented more than 3 systems with AJAX and can be considered a resource on the technology)
- [ ] Professional (this is your job. You work with AJAX on a daily basis)

**General Questions**

**DesktopApplication.** As you may know, AJAX is said to give web applications GUIs that behave a lot like desktop applications. In your opinion, what is (are) the major difference(s) between desktop applications and web applications?

**DesktopGUI.** What AJAX features, if any, makes web applications behave more like desktop GUIs?
Little/No Page Refreshing
Immediate Feedback
The fact that it makes user interactions faster
None
Other (s) (please specify)

AJAX Programming Experience

Why AJAX. Why have you chosen to develop in AJAX?

AJAX Start. How did you hear about AJAX?

☐ From a Friend
☐ From a blog or journal online
☐ From a professor and/or within a school/ university setting
☐ At your workplace
☐ In a technical book
☐ Other (please specify)

Length Of Use. For approximately how long have you been programming in AJAX?

☐ A month or less
☐ 1-2 months
☐ 3-5 months
☐ 6-8 months
☐ 9-11 months
☐ About a year
☐ Over a year
☐ A few years

Resources. What steps did you take to become skilled in AJAX programming?

☐ I consulted/ was mentored by someone who knew more about AJAX than I did
☐ I consulted online blogs and journals
☐ I followed online tutorials
☐ I attended AJAX/web developer conferences
☐ I read books/manuals on AJAX
I learned AJAX in school
Other (please specify) _______________________________________________________________________

**Resource Ranking.** Please rank these resources in order of how useful they were to you when learning AJAX.
1- Least useful
6- Most useful
N/A - I didn't use this resource

<table>
<thead>
<tr>
<th>Resource</th>
<th>N/A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>I consulted/ was mentored by someone who knew more about AJAX than I did</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I consulted online blogs and journals</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I followed online tutorials</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I attended AJAX/web developer conferences</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I read books/manuals on AJAX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I learned AJAX in school</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Learning Curve.** How would you describe the AJAX learning curve?
AJAX is very easy to learn and use
AJAX is somewhat easy to learn and use
AJAX is somewhat hard to learn and use
AJAX is very hard to learn and use
I can't really tell.

**Easy Features.** In your experience, what was the easiest aspect of AJAX to learn?

**Hard Features.** In your experience, what was the hardest aspect of AJAX to learn?

**Familiarity.** How long did it take you to become comfortable in AJAX programming?
An hour or less
A couple of hours or less
A few days but under a week
About a week
Several weeks but under a month
About one month
Several months
A year or more
I am still not comfortable programming in AJAX.
I can't remember

**ProjectDifficulty.** In your opinion, on average, about how difficult are your AJAX projects in terms of complexity and time of implementation?

- Very Hard
- Hard
- Moderate
- Easy
- Very Easy
- Decline to answer

**OtherWebLangs.** What other web technologies have you used to build webpages?

- HTML
- PHP
- JSP
- Perl
- Flash/Coldfusion
- ASP
- VBScript
- XUL
- XAML
- Java
- JavaScript
- Ruby
- Python
- Others (please specify)

**AJAXComparison.** Please compare and contrast your experience with AJAX with your experience with the above scripting languages, in terms of ease of use, complexity, ability to learn the language quickly, scalability, portability, etc.

NB: AJAX is essentially JavaScript so please compare your experience with AJAX to languages you have used other than JavaScript.
AJAX Development Environments

DevKits. Which Developer Kits have you used to program in AJAX?
- Backbase
- Dojo Toolkit
- SAJAX
- TIBCO General Interface.
- AJAX.NET
- Prototype
- Other (please specify)
- I don't use an AJAX development environment

Strengths. What strengths have each of these development kits demonstrated?

Weaknesses. What weaknesses have you noticed in each of these development kits?

EssentialFeatures. Please rank the following AJAX IDE features according to how necessary they are in making a good development tool for AJAX.
9 - most necessary
1 - least necessary

<table>
<thead>
<tr>
<th>Feature</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>A solid JavaScript debugger</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A built in webserver component</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A built in database component</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Built in Support for browser compatibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Built in support for Browser Back and Forward buttons</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A Rich Drag and Drop GUI interface</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Built in features that handle security issues</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Good Documentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Links/Associations with online or offline Developer Forums</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OtherEssentials. Are there any other features that you consider essential in an AJAX IDE?

AJAX Features

Feature Ranking. As a programmer, how important is it for the following features to appear in applications you develop? Please rank each of these features in order of importance to you by choosing a number for every row.

7 - MOST IMPORTANT
1 - LEAST IMPORTANT

<table>
<thead>
<tr>
<th>Feature</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little/No Refreshing of the entire page when a user clicks on a link or button</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fully functioning Browser Back/Forward Buttons</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ability to open all links in new windows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Immediate feedback (some notification that tells the user what's going on right after they press a button or a link)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A Secure Application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ability to work in all major browsers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective management of server and client states</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Difficulty Ranking. Please rank these features according to how hard they are to implement with AJAX both in terms of complexity and time to implement.

7 - Hard to implement
1 - Easy to implement

<table>
<thead>
<tr>
<th>Feature</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little/No Refreshing of the entire page when a user clicks on a link or button</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fully functioning Browser Back/Forward Buttons</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ability to open all links in new windows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Immediate feedback (some notification that tells the user what's going on right after they press a button or a link)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Secure Application

Ability to work in all major browsers

Effective management of server and client states

Security. What techniques, if any, are you using to combat security issues that AJAX raises?

The Future of AJAX

Future Growth. In your opinion, will AJAX become more or less prominent as a web technology?
- [ ] AJAX will become a more pervasive tool because
- [ ] AJAX will not grow as fast as many believe because
- [ ] AJAX will actually become less and less prominent because
- [ ] I have no opinion on the matter

AJAX Competitors. In the future, what technologies do you feel will be AJAX’s biggest competitors?
- [ ] XUL - XML User Interface Language
- [ ] XAML - eXtensible Application Markup Language, part of Avalon
- [ ] DHTML - Dynamic Hyper Text Markup Language
- [ ] Java Applets
- [ ] Flash
- [ ] Other (please specify)

Better Alternatives. Do you feel that there are better alternatives to AJAX? If so, please list them here and explain briefly why you believe them to be more useful than AJAX.

Final Comments

Satisfaction. Overall, are you satisfied with your experience as an AJAX developer?
- [ ] Very Satisfied
- [ ] Somewhat Satisfied
- [ ] Neutral
Continuing Use. In the future, will you continue using AJAX?

☐ Yes
☐ No
☐ I don't know

Additions. What features, if any, need to be added to AJAX development environments and why? Please think carefully about this question.

Removals. What features, if any, would you remove from AJAX development kits and why? Please think carefully about this question.

Changes. Perhaps you wouldn't like to fully remove some features, just tweak them. What features in AJAX developer tools would you change or adjust and why? Please think carefully about this question.
APPENDIX B – AJAX Frameworks and Libraries

JavaScript based

- The Dojo Toolkit: The kitchen sink
- Prototype: base Ajax library and more
- Script.aculo.us: rich effects built on Prototype
- Rico: rich Ajax components and effects built on Prototype
- Behaviour: Keep your HTML clean
- MochiKit: MochiKit makes JavaScript suck less
- Moo.fx: tiny effect library
- AJFORM
- Qooxdoo
- Zimbra AjaxTK: vendor toolkit
- Backbase: Commercial vendor toolkit
- TIBCO General Interface (aka “GI‿ ”): Commercial vendor toolkit
- JackBe: Commercial vendor toolkit
- Bindows: Commercial vendor toolkit
- Isomorphic: Commercial vendor toolkit
- ThyAPI
- Tibet: Commercial vendor toolkit
- AjaxFace: Commercial

Java based

- DWR: Proxy-based Ajax
- Echo2
- Ajax Tags
- WebWork 2
- Struts Layout
- Tapestry Tacos
- Wicket
- ICEfaces: Commercial
- ThinkCap JX: Commercial

PHP

- Sajax
- XOAD
- xajax
- AjaxAC
- JPSpan
- CPAINT
- Pear::Html::Ajax

.NET
• Ajax.NET
• Microsoft Atlas
• AjaxAspects
• WebORB for .NET

Cross Language / Other

• Ruby: Ruby on Rails
• Python: TurboGears
• Python: Crack Ajax
• Perl: CGI::Ajax
• Lisp: CL-Ajax
• JSON-RPC
Sources

- http://www.w3schools.com/ajax/default.asp
  SaferFasterBetterAJAX
- Dynamic User Interface
- Flexible AJAX, made with object oriented PHP 5
- http://ajaxpatterns.org/
- http://coldfusion.sys-con.com/read/138966.htm

HCI Links

- http://www.cc.gatech.edu/gvu/streaming/archives.html
- http://www.otal.umd.edu/guse/university.html - Degree programs and research centers
- encyclopedia of terms
- http://www.interaction-design.org/quote/

GMAIL

- http://mail.google.com/mail/help/blogs_more.html

Technical Books