

AJAX — A New Face for Client/Server Computing

Dan Kusnetzky, EVP Marketing Strategy
An Open-Xchange Position Paper



This is another in a series of Open-Xchange position papers on various topics. Each of the papers will focus on a trend Open-Xchange has observed, what impact this trend is expected to have on the industry and how Open-Xchange, Inc. will help organizations respond to that trend.

This position paper will address the question, "Does Asynchronous JavaScript and XML or AJAX represent a breakthrough in computer science or another evolutionary step in the development of distributed computing architectures?"

Everything old is new again

Distributed computing architectures have been evolving for decades. Competitive pressures force organizations to fully utilize the lower cost computing cycles offered by inexpensive desktop computers and industry standard servers in the data center. Today's Internet and Web-based applications are the culmination of this process.

The ultimate goal of distributed computing is to find the most efficient balance between servers, desktop computers, high speed networks and new development models. AJAX represents an important new development toward realizing the promise of distributed computing, but its impact will be incremental or evolutionary, not revolutionary.

What is AJAX?

Wikipedia states that "AJAX, shorthand for Asynchronous Javascript and XML, is a Web development technique for creating interactive web applications. The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user makes a change. This is meant to increase the web page's interactivity, speed, and usability."

Another way to say this is that AJAX strives to reproduce the usability benefits of a rich client without forcing developers to discard a standard web browser and develop dedicated client software for each supported client operating environment, i.e., Microsoft, UNIX, Linux, Mac OS etc. AJAX is based upon developing code using a neutral scripting language that the browser downloads each time a web site is visited or an application is run.

The result is a dynamic, graphical application that uses a standard web browser as a client rather than requiring an organization's IT department to install special code on each and every personal computer separately.

How is AJAX being used today?

AJAX is the foundation for a great deal of experimentation today. Many web sites and web-based applications are being enhanced in order to take advantage of its dynamic features. A few well-known examples are Google Mail and Google Calendar. In each case, a standard web browser appears to offer many of the features that previously were only available in a client/server application through a rich client.

A user of Google Mail or Google Calendar is presented with a very simple screen and menus that change depending upon the function or module that is currently being executed. The user's experience is better because screen elements can be updated individually rather than requiring a complete refresh of the whole page. This is possible because much of the code driving the screen was sent down the network to the client system when the web site or web-based application was first started. This code anticipates what the user was going to do and requests the needed information in the background while the user reads other content of the screen.

This also means that information can be made to pop up when the user places the mouse cursor over a highlighted area of the screen. This information could be gathered from several different sources and presented as needed. This can make a web site appear to be as dynamic as an older client/server application with a dedicated rich client.

Since standards for AJAX and AJAX usage are still emerging, web site and application developers feel free to experiment. In some cases, this experimentation has resulted in clean, easy-to-use and easy-to-understand applications. In other cases, the end result has been a very complex, feature laden application that is both hard to use and slow starting up.

AJAX – the good, the bad and the ugly

As with previous versions of distributed computing, the challenge is to use each component of the environment fully without also imposing bottlenecks. AJAX offers a new way to move the processing that supports the graphical user interface and some of the application's functions down to the client system.

The Good

We've seen very carefully implemented AJAX-based software that keeps most of the heavy computing and data management on the server. The results seen by the user are that the application is highly responsive and can be made to anticipate what data the user is going to request and fetch it from the server ahead of time.

The Bad

Other software products that are based upon AJAX haven't been as well thought through. A large amount of code must be downloaded to the personal computer each and every time the application is started. This means the user must wait for the download to finish before being able to use the software.

Since the application code is developed in a neutral scripting language, it can not be optimized for every client environment. This means that it might perform well in one client environment and very poorly or not at all in another client environment. This also means that simple web browsers, such as those found in SmartPhones, PDAs and other handheld devices may not be able to access the application at all.

The Ugly

We've seen AJAX-based applications that attempt to present a very large number of functions to the user on each page. While this may look good in a short demonstration, this overloaded interface is likely to be confusing in day-to-day use. We believe that applications should be architected so that each module only presents the commands, data and information relevant to that module.

What about standards?

Open-Xchange believes that software should be based upon open, international standards. This means using standard APIs, standard protocols and, whenever possible, standard data formats. Taking this approach offers organizations the greatest flexibility while still offering a high level of interoperability with products offered by many different suppliers. Organizations who select standards-based approaches are less likely to find themselves locked in to a single supplier's products or tools.

At this point, standards for AJAX are still evolving. A simple search for "AJAX Frameworks" results in a list of over 24 million pages at the time this is being written. Frameworks are available for C++, Coldfusion, DotNet, Java, Lisp, Lotus Notes, Multi-Language Ajax Frameworks, Perl, PHP, Python, Ruby, and even Smalltalk. Each implementation offers its own set of features. Suppliers such as Adobe, HP, IBM, Microsoft, Oracle and Sun each offer their own framework. Selecting the framework offered by a specific vendor may appear to be the correct choice today but leave the organization locked in tomorrow.

There are many open source projects as well. In each case, the project focuses on helping developers make use of AJAX-based client applications communicate with server-based computing. Although not an exhaustive list, some examples are DWR, JSON-RPC-Java, AjaxTags, AjaxAnywhere, Dojo and Echo. Developers of open source solutions are likely to select an open source framework. Dojo, for example, is extremely flexible and allows developers of Java-based solutions to support AJAX-enabled clients.

It's clear that the standards for AJAX are still being developed. Different vendors are jockeying for position in this arena. Each is trying to gain some advantage by having its own framework selected as a standard. In some cases, vendors have submitted a

framework for consideration and then withdrawn it later. Zimbra, for example, recently withdrew its "Kabuki" framework from the Apache Incubator. We suspect that once the framework was reviewed by other developers it became clear that the architecture used either was not generalized enough for broad use, needed some architectural enhancement or didn't meet the needs of the open source community as a whole.

Open-Xchange believes that it's wise for organizations to be very careful when selecting software that appears to have a single strength, such as it's based upon AJAX technology. AJAX, after all, is only a tool to make graphical applications more responsive. What's really crucial to performance of distributed applications is the overall reliability and scalability of a well-architected back end! The identity management, and transaction/ database processing architectures are more important in the long term.

Another way to look at this is that AJAX allows developers to create a dynamic, interactive window into back-end processing. A pretty user interface isn't much good to users if it doesn't connect with a strong back end, one that is built upon open standards, an open API, open data formats, and is reliable and scalable.

We suggest that server software be based upon open standards and use standard interfaces. This approach assures that functions can be easily integrated into the overall solution and a collaborative solution can be integrated into the organizations IT infrastructure.

Open-Xchange believes that AJAX is an important emerging technology and will make use of it in future versions of Open-Xchange Server. It also believes that AJAX, all by itself, can not be viewed as a panacea. An improperly implemented AJAX front end may actually cause the overall collaboration application to perform badly.

If you have any questions please contact:

OPEN-XCHANGE Inc.
303 South Broadway
Tarrytown, NY 01591, USA
info@OPEN-XCHANGE.com
www.OPEN-XCHANGE.com