5 Years Later: OpenAJAX Who?

Lori MacVittie, 2011-29-06

Five years ago the OpenAjax Alliance was founded with the intention of providing interoperability between what was quickly becoming a morass of AJAX-based libraries and APIs. Where is it today, and why has it failed to achieve more prominence?

I stumbled recently over a nearly five year old article I wrote in 2006 for Network Computing on the OpenAjax initiative. Remember, AJAX and Web 2.0 were just coming of age then, and mentions of Web 2.0 or AJAX were much like that of "cloud" today. You couldn't turn around without hearing someone promoting their solution by associating with Web 2.0 or AJAX. After reading the opening paragraph I remembered clearly writing the article and being skeptical, even then, of what impact such an alliance would have on the industry. Being a developer by trade I'm well aware of how impactful "standards" and "specifications" really are in the real world, but the problem – interoperability across a growing field of JavaScript libraries – seemed at the time real and imminent, so there was a need for someone to address it before it completely got out of hand.

> "With the OpenAjax Alliance comes the possibility for a unified language, as well as a set of APIs, on which developers could easily implement dynamic Web applications. A unified toolkit would offer consistency in a market that has myriad Ajax-based technologies in play, providing the enterprise with a broader pool of developers able to offer long term support for applications and a stable base on which to build applications. As is the case with many fledgling technologies, one toolkit will become the standard—whether through a standards body or by de facto adoption—and Dojo is one of the favored entrants in the race to become that standard."
>
> -- AJAX-based Dojo Toolkit , Network Computing, Oct 2006

The goal was simple: interoperability. The way in which the alliance went about achieving that goal, however, may have something to do with its lackluster performance lo these past five years and its descent into obscurity.

5 YEAR ACCOMPLISHMENTS of the OPENAJAX ALLIANCE

The OpenAjax Alliance members have not been idle. They have published several very complete and well-defined specifications including one "industry standard": OpenAjax Metadata.

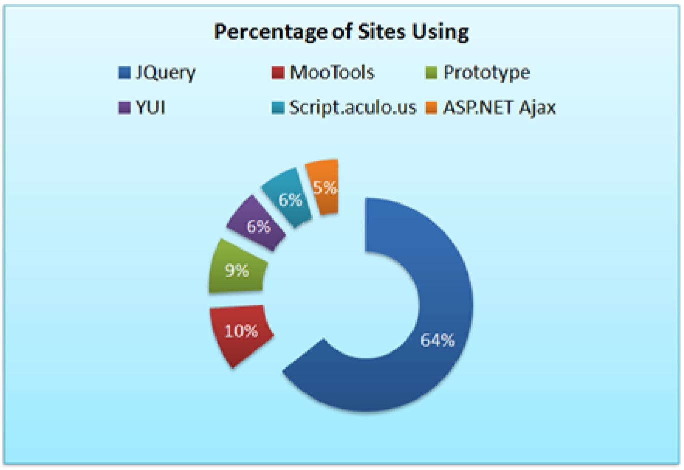
OpenAjax Hub

- The OpenAjax Hub is a set of standard JavaScript functionality defined by the OpenAjax Alliance that addresses key interoperability and security issues that arise when multiple Ajax libraries and/or components are used within the same web page. (OpenAjax Hub 2.0 Specification)

OpenAjax Metadata

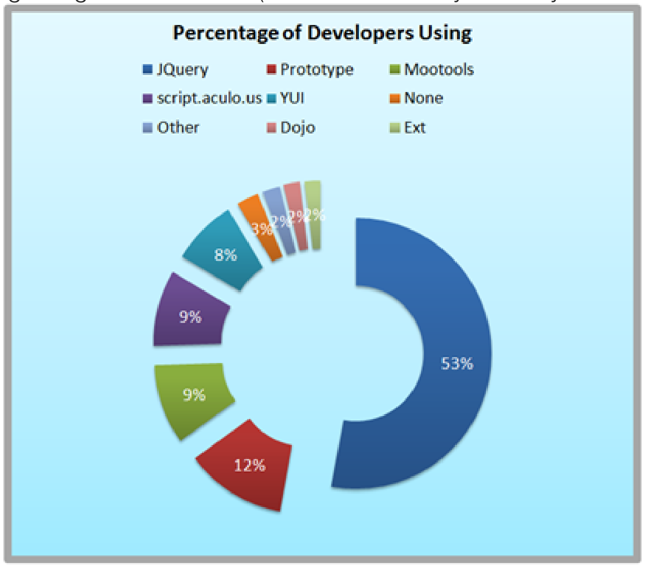
- OpenAjax Metadata represents a set of industry-standard metadata defined by the OpenAjax Alliance that enhances interoperability across Ajax toolkits and Ajax products (OpenAjax Metadata 1.0 Specification)
- OpenAjax Metadata defines Ajax industry standards for an XML format that describes the JavaScript APIs and widgets found within Ajax toolkits. (OpenAjax Alliance Recent News)

It is interesting to see the calling out of XML as the format of choice on the OpenAjax Metadata (OAM) specification given the recent rise to ascendancy of JSON as the preferred format for developers for APIs. Granted, when the alliance was formed XML was all the rage and it was believed it would be the dominant format for quite some time given the popularity of similar technological models such as SOA, but still – the reliance on XML while the plurality of developers race to JSON may provide some insight on why OpenAjax has received very little notice since its inception.



SOURCE: W3Techs, **Historical trends in the usage of JavaScript libraries for websites**
http://w3techs.com/technologies/history_overview/javascript_library/all

Ignoring the XML factor (which undoubtedly is a fairly impactful one) there is still the matter of how the alliance chose to address run-time interoperability with OpenAjax Hub (OAH) – a hub. A publish-subscribe hub, to be more precise, in which OAH mediates for various toolkits on the same page. Don summed it up nicely during a discussion on the topic: it's page-level integration. This is a very different approach to the problem than it first appeared the alliance would take.



SOURCE: **State of Web Development 2010 Results**
http://www.webdirections.org/sotw10/results/

The article on the alliance and its intended purpose five years ago clearly indicate where I thought this was going – and where it should go: an industry standard model and/or set of APIs to which other toolkit developers would design and write such that the interface (the method calls) would be unified across all toolkits while the implementation would remain whatever the toolkit designers desired.

I was clearly under the influence of SOA and its decouple everything premise. Come to think of it, I still am, because interoperability assumes such a model – always has, likely always will. Even in the network, at the IP layer, we have standardized interfaces with vendor implementation being decoupled and completely different at the code base. An Ethernet header is always in a specified format, and it is that standardized interface that makes the Net go over, under, around and through the various routers and switches and components that make up the Internets with alacrity. Routing problems today are caused by human error in configuration or failure – never incompatibility in form or function.

Neither specification has really taken that direction. OAM – as previously noted – standardizes on XML and is primarily used to describe APIs and components - it isn't an API or model itself. The Alliance wiki describes the specification: "The primary target consumers of OpenAjax Metadata 1.0 are software products, particularly Web page developer tools targeting Ajax developers." Very few software products have implemented support for OAM. IBM, a key player in the Alliance, leverages the OpenAjax Hub for secure mashup development and also implements OAM in several of its products, including Rational Application Developer (RAD) and IBM Mashup Center. Eclipse also includes support for OAM, as does Adobe Dreamweaver CS4. The IDE working group has developed an open source set of tools based on OAM, but what appears to be missing is adoption of OAM by producers of favored toolkits such as jQuery, Prototype and MooTools. Doing so would certainly make development of AJAX-based applications within development environments much simpler and more consistent, but it does not appear to gaining widespread support or mindshare despite IBM's efforts.

The focus of the OpenAjax interoperability efforts appears to be on a hub / integration method of interoperability, one that is certainly not in line with reality. While certainly developers may at times combine JavaScript libraries to build the rich, interactive interfaces demanded by consumers of a Web 2.0 application, this is the exception and not the rule and the pub/sub basis of OpenAjax which implements a secondary event-driven framework seems overkill. Conflicts between libraries, performance issues with load-times dragged down by the inclusion of multiple files and simplicity tend to drive developers to a single library when possible (which is most of the time). It appears, simply, that the OpenAJAX Alliance – driven perhaps by active members for whom solutions providing integration and hub-based interoperability is typical (IBM, BEA (now Oracle), Microsoft and other enterprise heavyweights – has chosen a target in another field; one on which developers today are just not playing.

It appears OpenAjax tried to bring an enterprise application integration (EAI) solution to a problem that didn't – and likely won't ever – exist. So it's no surprise to discover that references to and activity from OpenAjax are nearly zero since 2009. Given the statistics showing the rise of JQuery – both as a percentage of site usage and developer usage – to the top of the JavaScript library heap, it appears that at least the prediction that "one toolkit will become the standard—whether through a standards body or by de facto adoption" was accurate.

Of course, since that's always the way it works in technology, it was kind of a sure bet, wasn't it?

WHY INFRASTRUCTURE SERVICE PROVIDERS and VENDORS CARE ABOUT DEVELOPER STANDARDS

You might notice in the list of members of the OpenAJAX alliance several infrastructure vendors. Folks who produce application delivery controllers, switches and routers and security-focused solutions. This is not uncommon nor should it seem odd to the casual observer. All data flows, ultimately, through the network and thus, every component that might need to act in some way upon that data needs to be aware of and knowledgeable regarding the methods used by developers to perform such data exchanges. In the age of hyper-scalability and über security, it behooves infrastructure vendors – and increasingly cloud computing providers that offer infrastructure services – to be very aware of the methods and toolkits being used by developers to build applications. Applying security policies to JSON-encoded data, for example, requires very different techniques and skills than would be the case for XML-formatted data. AJAX-based applications, a.k.a. Web 2.0, requires different scalability patterns to achieve maximum performance and utilization of resources than is the case for traditional form-based, HTML applications. The type of content as well as the usage patterns for applications can dramatically impact the application delivery policies necessary to achieve operational and business objectives for that application.

As developers standardize through selection and implementation of toolkits, vendors and providers can then begin to focus solutions specifically for those choices. Templates and policies geared toward optimizing and accelerating JQuery, for example, is possible and probable. Being able to provide pre-developed and tested security profiles specifically for JQuery, for example, reduces the time to deploy such applications in a production environment by eliminating the test and tweak cycle that occurs when applications are tossed over the wall to operations by developers. For example, the jQuery.ajax() documentation states:

> “By default, Ajax requests are sent using the GET HTTP method. If the POST method is required, the method can be specified by setting a value for the `type` option. This option affects how the contents of the `data` option are sent to the server. POST data will always be transmitted to the server using UTF-8 charset, per the W3C XMLHTTPRequest standard.
>
> The `data` option can contain either a query string of the form `key1=value1&key2=value2`, or a map of the form `{key1: 'value1', key2: 'value2'}`. If the latter form is used, the data is converted into a query string using `jQuery.param()` before it is sent. This processing can be circumvented by setting `processData` to `false`. The processing might be undesirable if you wish to send an XML object to the server; in this case, change the `contentType` option from `application/x-www-form-urlencoded` to a more appropriate MIME type. ”

Web application firewalls that may be configured to detect exploitation of such data – attempts at SQL injection, for example – must be able to parse this data in order to make a determination regarding the legitimacy of the input. Similarly, application delivery controllers and load balancing services configured to perform application layer switching based on data values or submission URI will also need to be able to parse and act upon that data. That requires an understanding of how jQuery formats its data and what to expect, such that it can be parsed, interpreted and processed.

By understanding jQuery – and other developer toolkits and standards used to exchange data – infrastructure service providers and vendors can more readily provide security and delivery policies tailored to those formats natively, which greatly reduces the impact of intermediate processing on performance while ensuring the secure, healthy delivery of applications.

- API Jabberwocky: You Say Tomay-to and I Say Potah-to
- OpenAjax Metadata 1.0 and the Adobe Dreamweaver Widget Browser
- OpenAjax Alliance
- AJAX-based Dojo Toolkit
- The Stealthy Ascendancy of JSON
- JSON Continues its Winning Streak Over XML
- JSON versus XML: Your Choice Matters More Than You Think
- I am in your HTTP headers, attacking your application
- The Web 2.0 API: From collaborating to compromised
- IT as a Service: A Stateless Infrastructure Architecture Model
- JSON Activity Streams and the Other Consumerization of IT

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc. Corporate Headquarters info@f5.com	F5 Networks Asia-Pacific apacinfo@f5.com	F5 Networks Ltd. Europe/Middle-East/Africa emeainfo@f5.com	F5 Networks Japan K.K. f5j-info@f5.com