

http://www.alexhopmann.com/xmlhttp.htm

Go

MAY

JUN

JUL

17

2005

2006

2007

84 captures

17 Jun 2006 - 4 Mar 2026

2005

2006

2007

About this capture

Email Alex Hopmann

Home

XMLHTTP

Lancair

Art

Travel

Food

Network Magic

About Alex Hopmann

Images

Airplanes

Flying

Family

Food

Travel

Archive

February, 2005

March, 2005

April, 2005



The story of XMLHTTP

Leaving Microsoft has certainly encouraged me to reflect a little bit on some of the projects I've been involved with. One that is probably worth sharing is the history of XMLHTTP.

XMLHTTP has lately become a huge hit. It seems like people noticed it when Google started using it in the [Google Suggest feature](#) and they looked at the source code of that page to figure out how it worked. Google had actually been using it earlier in Gmail, but the JavaScript sources behind Gmail was much more complicated and people didn't notice it right away.

Since then XMLHTTP and the "AJAX" (Asynchronous Javascript And XML) methodology have been mentioned in the Wall Street Journal and [CNET](#) among other news sources.

The reality is that the client architecture of Gmail appears to follow the rough design of the Exchange 2000 implementation of Outlook Web Access for IE5 and later which shipped way back in 2000.

XMLHTTP actually began its life out of the Exchange 2000 team. I had joined Microsoft in November 1996 and moved to Redmond in the spring of 1997 working initially on some Internet Standards stuff as related to the future of Outlook. I was specifically doing some work on meta-data for web sites including an early proposal called "Web Collections". During this time period Thomas Reardon one day dragged me down the hall to introduce me to this guy named Jean Paoli that had just joined the company. Jean was working on this new thing called XML that some people suspected would be very big some day (for some unclear reason at the time).

After I got the gist of XML, I switched my Web Collections proposal to take advantage of it and published a "W3C Submission" entitled "[Web Collections using XML](#)". In retrospect this is actually a fairly embarrassing document, but what can I say- back at the time "push" was the hottest thing on the web and to the best of my knowledge this is the earliest published public document discussing XML anywhere.

This led to me getting involved with some Microsoft folks who were working on the WebDAV standard and I made a big push to move WebDAV to using XML as its model for communicating data. Shortly after I joined the Exchange team which was just ramping up on Exchange 2000 to lead the development of Microsoft's WebDAV servers (which were implemented both on normal IIS over the file system- DAVFS, and over Exchange data- DAVEX).

I don't recall exactly when we started working on Outlook Web Access in Exchange 2000. I think it was about a year after I joined the team, probably sometime in late 1998. In any case we were already a milestone or two into the Exchange 2000 (or "Platinum") project and had been carefully ignoring the issue of OWA mostly because the old version was such a hack. The old version had been written in a big rush by a previous iteration of the team using ASP pages and was both fairly ugly and had huge server scalability and performance problems.

At some point Brian Valentine (who was still the GM of Exchange at the time) asked us to figure out what to do with OWA for Exchange 2000. There were two implementations that got started, one based on serving up straight web pages as efficiently as possible with straight HTML, and another one that started playing with the cool user interface you could build with DHTML. When I first got a demo of the DHTML work that Jim Van Eaton and Bob Gering were doing I was just blown away. However they were basically doing hacky form-posts back to the server and had some of the same scalability and dynamic data problems of the old version.

That weekend I startup up Visual Studio and whipped up the first version of what would become XMLHTTP. The first verison didn't have async support hooked up and was pretty crude, but it was enough to help Jim and Bob talk to some of the WebDAV/XML support we had already built for Exchange 2000 and make some really rapid progress with their DHTML OWA implementation.

Next time- how we actually shipped XMLHTTP.

It was one thing to have this cool component we could use to talk to the server, but we still weren't in any position to use it in the shipping product yet. The basic premise of Outlook Web Access was that you could walk up to any computer that had the browser on it and just get to your email. If we needed to download ActiveX controls, it was going to ruin the basic premise of the product, so the next step was to embark on shipping this thing in Internet Explorer.

Step one was to bring the code up to production quality so we got Shawn Bracewell, one of the devs on the OWA team to take it over. Being a smart guy he promptly threw away all of my code and rewrote it in a more solid fashion, adding async support, error handling and more.

Meanwhile the IE project was just weeks away from beta 2 which was their last beta before the release. This was the good-old-days when critical features were crammed in just days before a release, but this was still cutting it close. I realized that the MSXML library shipped with IE and I had some good contacts over in the XML team who would probably help out- I got in touch with Jean Paoli who was running that team at the time and we pretty quickly struck a deal to ship the thing as part of the MSXML library. Which is the real explanation of where the name XMLHTTP comes from- the thing is mostly about HTTP and doesn't have any specific tie to XML other than that was the easiest excuse for shipping it so I needed to cram XML into the name (plus- XML was the hot technology at the time and it seemed like some good marketing for the component).

While Shawn worked with Chris Lovett on the XML team to integrate with their library, I covered the rest of the "ship it" bases and got sign-off from Joe Peterson who was running IE at the time. We did a quick security review but at the time no one really understood the seriousness or types of vulnerabilities that anything shipping could cause. In terms of quality for the beta the theory was "no one other than OWA is going to try to call it so if it has a bug in the beta, worst case we just can't use it." With a week to go before the beta we checked it in, got a build and tried it out and everything looked great.

The beta shipped and the OWA team was able to start running forward using the beta IE5, while Shawn fixed a number of bugs through the ship day of IE5. It took us another year and a half to ship Exchange 2000, but this was one of those cases where the little bit of planning ahead meant that by the time we shipped the necessary bits were already pretty widely out there.

Actually shipping the thing required help from many more people that I've mentioned so far (and I'm sure I've forgotten others). Karim Battish and Sean Lyndersay were both involved at various times keep the spec updated (which was written pretty much after the implementation, but sometimes that is just how it works out) and working with partner teams. Chris Tyner was critical on the test side- I'm pretty sure without his energy and perseverance we would have shipped the thing with numerous holes.

If I recall Chris made this killer HTML page that gave you a UI for sending arbitrary HTTP requests. I think it was called "DAVUI" and it had templates for all the common HTTP 1.1, WebDAV and special Exchange requests, and unlike the tool we had been using before (that was a traditional Win32 EXE) you could just copy this one .htm file to any machine or else run it remotely and it just worked. Brilliant.

There is a [new article on CNET](#) talking about AJAX / the Dynamic HTML + JavaScript + XMLHTTP development methodology. It questions whether the AJAX approach diminishes the value of things like Flash and Microsoft's Avalon / XAML. Having spent the past 4 years working on Avalon I think this totally misses the value of Avalon and managed code. Building Outlook Web Access was really really very very hard, and even so the results were not comparable to the real thing (Outlook). You can't take it offline, you can't use it with multiple mail accounts in a coordinated way, and the UI is just somewhat more clunky. This doesn't take away from the core value of OWA in being something you can get to from anywhere. Still, the programming model and maintainability of the DHTML approach was really difficult as the app got bigger. The beauty of many of the Google applications is that they are really tight focused apps that don't try to do that much, but really take nice advantages of the web characteristics. It seems like a false assumption to assume that this approach will scale up for applications like Office. Avalon is based on a programming model that is a big step up in terms of componentization and maintainability, plus it leapfrogs the type of graphics, user interface and just plain experience you can deliver even with finely tuned full Windows code.

The lesson to take out of this thing is to appreciate the importance of shipping and having the patience to let something succeed. It feels like the press (and sometimes even the internal attitude) about something like Avalon is to measure it by how many apps have adopted it on the launch day. That is just crazy and it doesn't get the basics of how big a shift these sorts of things can be and how long it takes to move the mind-set, learning, and deployment of a big new platform. I'm convinced that if Microsoft ships Avalon next year and Microsoft and the industry manage to stick to it and take the long-term perspective, its going to be ubiquitous in 5 years. But it really does take a few years before the actual momentum is there.

That was a great thing about working at Microsoft and I hope they still have it- the long term vision and ability to commit to things until version 3 when it really hits the mainstream. I'm trying to build an aspect of that culture at [Pure Networks](#) although it needs to have an entirely different flavor. As a start-up we need to be constantly adapting, shipping very quickly and efficiently. But part of the cool thing about the company and one of the things that really attracted me to it is there is this great opportunity for that really big picture long-term roadmap that is often missing from these types of ventures. [Network Magic](#) is a great teaser of the types of things you can pull off with software that deeply understands your home network but it is really just the beginning.

I've also got a few teasers online of new projects that I'm working on. The image libraries like [this airplane picture set with AJAX based zooming and manipulation](#) and [my calendardata.com web site which helps syndicate calendar events](#) are some to get started.

The fine print- The opinions on this web site are my own and do not necessarily represent my employer.