

Simplifying Ajax-Style Web Development

Keith Smith
Microsoft



A new framework makes it easier to create rich Web experiences using Ajax techniques.

Conventional browser-based Web applications require the user to submit a request to the server, wait for the server to process the request and generate a response, and then wait for the browser to update the interface with the results. This request-wait-response-wait pattern is extremely disruptive and lowers productivity.

High network latency and interface complexity and slow server responsiveness can further impair the user experience, resulting in decreased customer satisfaction, shorter and less frequent Web site visits, and, ultimately, reduced revenue to e-businesses.

Asynchronous JavaScript and XML is a standards-based programming technique designed to make Web-based applications more responsive, interactive, and customizable—in short, to recreate the seamless user experience of most other desktop applications.

In his February 2005 essay, “Ajax: A New Approach to Web Applications” (www.adaptivepath.com/publications/essays/archives/000385.php), Adaptive Path’s Jesse James Garrett listed five key characteristics of applications built using Ajax:

- a user interface constructed with open standards such as the Dynamic Hypertext Markup Language and cascading stylesheets;
- a dynamic, interactive user experience enabled by the document object model;
- data exchange and transformation using the Extensible Markup Language (XML) and Extensible Stylesheet Language Transformations;
- asynchronous client/server communication via XMLHttpRequest; and
- JavaScript as the lingua franca joining all the components together.

Ajax offers many advantages over conventional approaches to Web application development, but it also has several shortcomings that have slowed widespread adoption.

ADVANTAGES

The primary advantages of Ajax-style Web applications are less waiting and more control for the user. Ajax accomplishes this by

- eliminating full-page post-backs in

favor of smaller, incremental in-place updates;

- leveraging the client machine’s processing power and temporal proximity by making the Web browser responsible for more aspects of the application execution; and
- exploiting modern Web browsers’ rich graphics capabilities—transparency, shading, animation, Z-ordering, compositing, and so on—to add more glitz and interactivity to the presentation of information.

Consider a fairly common scenario involving an online travel reservation system. A user types in her place of departure, travel destination, and preferred dates and times, then clicks a button to locate available flights. She then waits for the server to process her query and generate a response.

When the browser eventually receives the response, the user notices she typed the wrong airport code for her destination. Consequently, the dozens of flight itineraries returned are useless. She therefore corrects the airport code (or clicks her browser’s back button) and resubmits her request.

Again she must wait for the server to process the new query and return the results to her browser before she can view them. Any sorting or filtering of the returned flight itineraries requires additional post-backs and more waiting.

By simply typing the wrong airport code, the user has wasted her time as well as network and server resources—bandwidth, database accesses, and so on.

The system’s designers could employ several Ajax techniques to improve the user’s experience and increase productivity. For example, as the user types the airport code, the browser could make a request in the background for the airports matching the letters she has typed so far and display an autocomplete list of matching codes including the airport name and city. The user would then immediately notice whether she had typed the correct code before submitting the request to the server.

She could also send a small XML fragment describing her request—locations, dates, times, and airport codes—instead of a complete page post-back, which generates a chunk of XML describing the matching flight itineraries as a response. The browser-to-server data exchange would be less verbose and more efficient than a round-trip between browser and server, which requires re-rendering the entire Web page.

In addition, since her browser now has a full description of the matching flight itinerary information in XML, the user now can manipulate the presentation of that data without incurring the overhead of full-page post-backs to the server.

Further, she can now quickly sort, page through, and search the flight itinerary results again, foregoing unnecessary round-trips between browser and server. Once she locates her preferred flight itinerary, the user can drag-and-drop it into the “My Itinerary” cart and move on to finding a hotel and rental car for her trip.

SHORTCOMINGS

Ajax’s underlying technologies, such as JavaScript and XML, have been around for several years. For instance, Microsoft released Ajax’s workhorse, XMLHttpRequest, with Internet Explorer 5.0 in March 1999. Further, Ajax-style Web applications existed long before Garrett coined the phrase—one notable example being Microsoft Outlook Web Access, which has powered the Web interfaces of many enterprise messaging systems since its release with Microsoft Exchange 2000.

So why has it taken so long for Ajax-style development to catch on, and why aren’t all Web application designers and developers rushing to incorporate Ajax techniques into their sites? The main reason is that such development is extremely difficult due to

- limitations of the basic JavaScript language;
- browser incompatibilities; and
- a lack of tool support for design-

ing, developing, and debugging this new breed of Web application.

It was not until popular online services such as Google Maps, Google Gmail, and Yahoo Flickr debuted that Ajax gained ground in the development community. These applications have helped drive mainstream expectations of a richer Web experience, and now companies including Microsoft are creating code libraries and tools that any developer can use to quickly build sites that incorporate Ajax techniques.

ATLAS

Microsoft’s ASP.NET code name “Atlas” (<http://atlas.asp.net>) is a new cross-platform, cross-browser framework for building rich, interactive, personalized Web experiences using Ajax techniques. The key objectives of Atlas are to

- create a high-productivity platform for browser-centric Web applications;
- enable ubiquitous reach with friction-free deployment and administration;
- make Ajax-style Web applications

easy to author, debug, and maintain; and

- provide an enhanced user and developer experience on the Microsoft platform.

Figure 1 shows the framework architecture.

Browser compatibility

Web developers often waste valuable time working around browser compatibility issues rather than focusing on the logic unique to their applications. Atlas addresses this problem by providing a suite of components, controls, and behaviors for common scenarios. In addition, Atlas provides standards-based JavaScript extensions that developers can use to create cross-browser applications using familiar object-oriented programming concepts such as types, namespaces, classes, and interfaces.

What this means is fewer new concepts to learn, fewer lines of code to write, and a lower barrier to entry for Java, .NET, and other server-side developers making the transition to browser-centric, client-side development.

The Atlas client script library is implemented using standard Java-

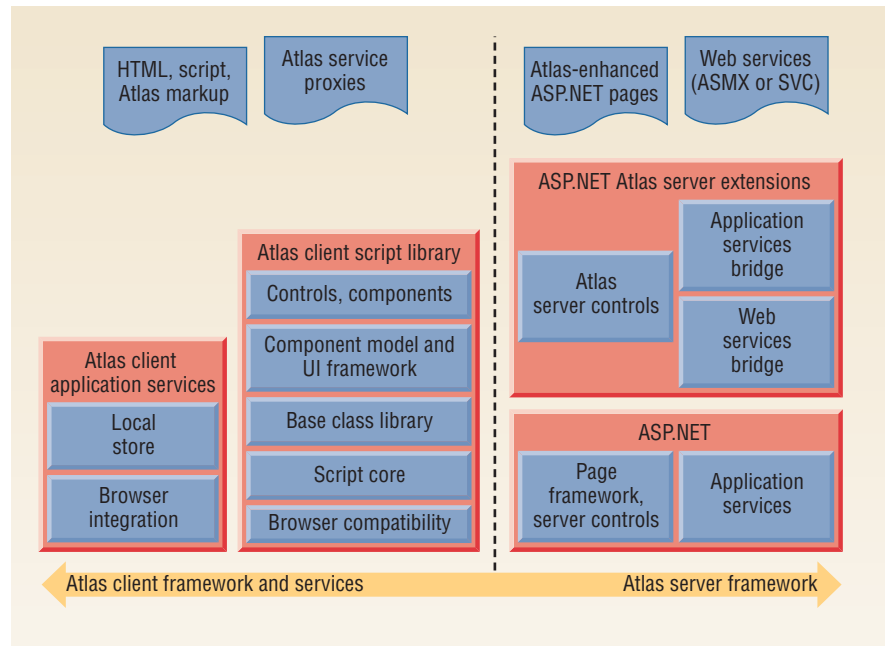


Figure 1. Microsoft Atlas architecture. Atlas offers a cross-platform, cross-browser framework for building rich, interactive, personalized Web experiences using Ajax techniques.

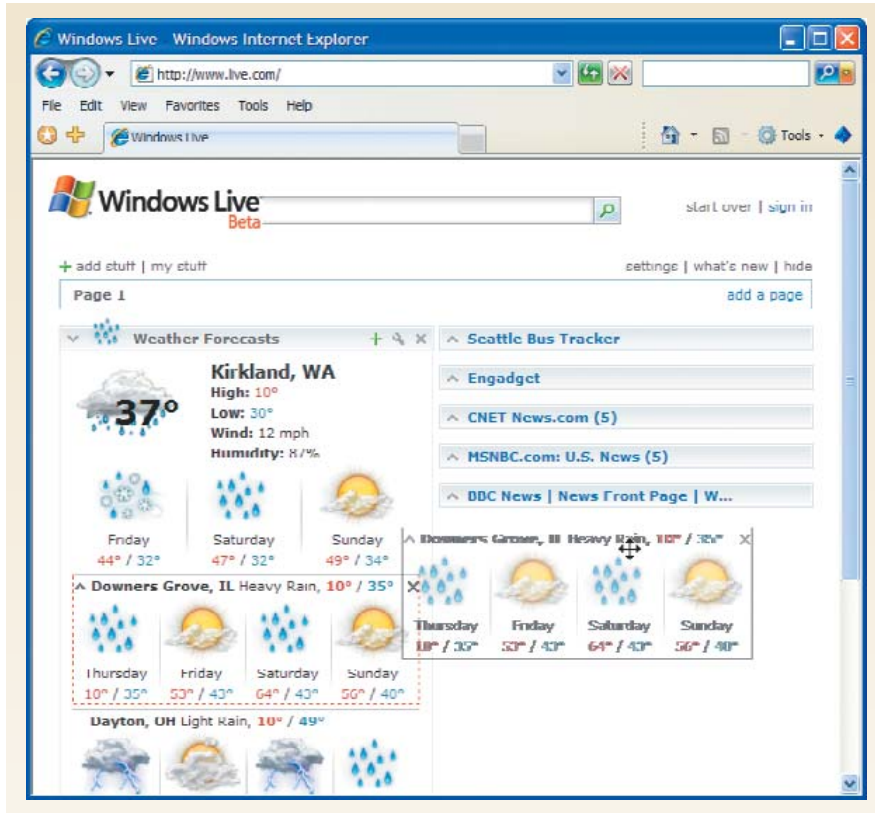


Figure 2. Microsoft Gadgets. Atlas can be used to create user-customizable Web applications, like this weather Gadget hosted on Windows Live, that present information in a readily accessible way.

Script (ECMAScript) and is responsible for most productivity enhancements. Together, Atlas.js and Atlas-Runtime.js implement the script core, base class library, component model, UI framework, and controls and component support common to every Atlas application independent of the Web browser making the requests.

AtlasCompat.js provides compatibility for Firefox and other Mozilla-based browsers, while AtlasCompat2.js provides compatibility for Apple Safari. The client script library core natively supports Internet Explorer.

Atlas also supports a clean, object-oriented programming syntax with rich objects for HTML intrinsic controls (Sys.UI.TextBox) and for making background server requests from the client (Sys.Net.WebRequest) and processing the results of those requests (Sys.Net.WebResponse).

Ubiquity

Using Atlas, Web designers and

developers can quickly create Web applications that are accessible using any modern browser running on any operating system without requiring installation of any client-side components. Moreover, the client script library can be deployed to any Web application framework including PHP, JavaServer Pages, ColdFusion, and ASP.NET.

Atlas can be used to enhance any Web application with Ajax functionality independent of the server or client platform. For example, a PHP-based Web site running on Linux can be enhanced with Atlas to provide an equally rich and completely seamless user experience to users browsing the site with Safari on a Mac, Firefox on a Linux box, or Internet Explorer on a Windows PC.

Usability

To make Ajax-style Web applications easy to author, debug, and maintain, Atlas features an extensive declarative

programming model that cleanly separates content, style, structure, and program logic. Web designers and developers collaborating on an application can swap design and programmatic elements from their respective tools while maintaining complete fidelity.

Future versions of the Microsoft Expression and Microsoft Visual Studio product lines will provide robust tool support for Atlas including features for modeling, debugging, testing, and deployment during the entire software development life cycle.

Enhanced Web experience

Several other script libraries and frameworks in various stages of development seek to address Ajax's shortcomings—most notably, Ajax.NET, AjaxTK, Backbase, Dojo, script.aculo.us, and the Yahoo UI Library. However, these solutions typically focus on creating a more productive client-side scripting platform or enabling a richer user experience across all Web browsers. Few have easy authoring and integrated tool support as a primary goal.

In addition to providing a portable client-side library, Atlas seamlessly integrates with the Microsoft platform. For example, developers targeting ASP.NET 2.0 have full access to features including the Membership application service, the Profile/Personalization system, and ASMX-based Web services within the Atlas client script library and programming model.

Both developers and users can benefit from Ajax-style Web applications created using Atlas in conjunction with products such as Internet Explorer, Internet Information Server, Windows Live, Windows Communication Foundation, Office, and Windows Vista.

For example, a company can exploit Atlas support for Microsoft Gadgets (<http://microsoftgadgets.com>) to make existing Web properties reusable and easy to deploy. After creating Gadgets through a simple step-by-step process, the company can let its partners host

its shared components and services on their sites, while its customers can load the Gadgets on their Windows Live portals and drop them onto their Windows Vista Sidebar. This gives the company broader reach and additional revenue opportunities through increased traffic and brand awareness, while customers get convenient access to useful functionality in the places they visit most frequently.

Figure 2 shows a weather Gadget hosted on Windows Live that implements a rich interface with drag-and-drop positioning, animation, and dynamic updating. This Gadget is customizable and presents information in a way that is readily accessible alongside other content the user finds interesting or relevant.

Another example would be a major consumer electronics retailer that has a section on its main page dedicated to tracking inventories of Xbox 360 consoles across all locations. The retailer could use Atlas to package this inventory tracker as a Gadget, allowing its customers to determine local Xbox 360 availability from their personal homepages.

Atlas is still under development, but Microsoft releases fully functional, high-quality *community technology preview* builds every six to eight weeks with updated samples and documentation. You can download the latest CTP from the Atlas Web site, which also contains links to blogs, tutorials, webcasts, training, and a community forum. Users can participate in the development process by providing feedback directly to members of the Atlas product team. ■

Keith Smith is a senior product manager at Microsoft. Contact him at keiths@microsoft.com.

Editor: Simon S.Y. Shim, Department of Computer Engineering, San Jose State University; sishim@email.sjsu.edu

How to Reach Computer

Writers

We welcome submissions. For detailed information, visit www.computer.org/computer/author.htm.

News Ideas

Contact Lee Garber at lgarber@computer.org with ideas for news features or news briefs.

Products and Books

Send product announcements to developertools@computer.org. Contact computer-ma@computer.org with book announcements.

Letters to the Editor

Please provide an e-mail address with your letter. Send letters to computer@computer.org.

On the Web

Explore www.computer.org/computer/ for free articles and general information about *Computer* magazine.

Magazine Change of Address

Send change-of-address requests for magazine subscriptions to address.change@ieee.org. Make sure to specify *Computer*.

Missing or Damaged Copies

If you are missing an issue or received a damaged copy, contact membership@computer.org.

Reprint Permission

To obtain permission to reprint an article, contact William Hagen, IEEE Copyrights and Trademarks Manager, at whagen@ieee.org. To buy a reprint, send a query to computer@computer.org.

