1

What Is Ajax?

From 2001 to 2005, the World Wide Web went through a tremendous growth spurt in terms of the technologies and methodologies being used to bring this once-static medium to life. Online brochures and catalogs no longer dominated the Internet as web applications began to emerge as a significant portion of online destinations. Web applications differed from their web site ancestors in that they provided an instant service to their users, not just information. Whether for business process management or personal interests, developers were forced to create new interaction paradigms as users came to expect richer functionality.

Spurred on by little-known and lesser-used technologies that had been included in web browsers for some time, the Web took a bold step forward, shattering the traditional usage model that required a full page load every time new data or a new part of the application's logic was accessed. Companies began to experiment with dynamic reloading of portions of web pages, transmitting only a small amount of data to the client, resulting in a faster, and arguably better, user experience.

At the forefront of this movement was Google. After the search giant went public, new experiments conducted by Google engineers began popping up through a special part of the site called Google Labs (`labs.google.com`). Many of the projects at Google Labs, such as Google Suggest and Google Maps, involved only a single web page that was never unloaded but was constantly updated nevertheless. These innovations, which began to bring the affordances of desktop software interfaces into the confines of the browser, were praised around the Web as ushering in a new age in web development. And indeed they did.

Numerous open source and commercial products began development to take advantage of this new web application model. These projects explained their technology using a variety of terms such as JavaScript remoting, web remote procedure calls, and dynamic updating. Soon, however, a new term would emerge.

Ajax Is Born

In February 2005, Jesse James Garrett of Adaptive Path, LLC published an online article entitled, "Ajax: A New Approach to Web Applications" (still available at `www.adaptivepath.com/publications/essays/archives/000385.php`). In this essay, Garrett explained how he believed web applications were closing the gap between the Web and traditional desktop applications. He cited new technologies and several of the Google projects as examples of how traditionally desktop-based user interaction models were now being used on the Web. Then came the two sentences that would ignite a firestorm of interest, excitement, and controversy:

> *Google Suggest and Google Maps are two examples of a new approach to web applications that we at Adaptive Path have been calling Ajax. The name is shorthand for Asynchronous JavaScript + XML, and it represents a fundamental shift in what's possible on the Web.*

From that point forward, a tidal wave of Ajax articles, code samples, and debates began popping up all over the Web. Developers blogged about it, technology magazines wrote about it, and companies began hitching their products to it. But to understand what Ajax is, you first must understand how the evolution of several web technologies led to its development.

The Evolution of the Web

When Tim Berners-Lee crafted the first proposal for the World Wide Web in 1990, the idea was fairly simple: to create a "web" of interconnected information using hypertext and Uniform Resource Identifiers (URIs). The ability to link disparate documents from all around the world held huge potential for scholarly endeavors, where people would be able to access referenced material almost instantly. Indeed, the first version of the HyperText Markup Language (HTML) featured little more than formatting and linking commands, a platform not for building rich interactive software but rather for sharing the kinds of textual and illustrative information that dominated the late age of print. It was from these static web pages that the Web grew.

As the Web evolved, businesses saw potential in the ability to distribute information about products and services to the masses. The next generation of the Web saw an increased ability to format and display information as HTML also evolved to meet the needs and match the expectations of these new media-savvy users. But a small company called Netscape would soon be ready to push the evolution of the Web forward at a much faster pace.

JavaScript

Netscape Navigator was the first successful mainstream web browser, and as such, moved web technologies along quickly. However, Netscape often was ridiculed by standards organizations for implementing new technologies and extensions to existing technologies before the standards were in place (much as Microsoft is being chastised today for ignoring existing standards in its development of Internet Explorer). One such technology was JavaScript.

Originally named LiveScript, JavaScript was created by Brendan Eich of Netscape and included in version 2.0 of the browser (released in 1995). For the first time, developers were able to affect how a web page could interact with the user. Instead of making constant trips to the server and back for simple tasks such as data validation, it became possible to transfer this small bit of processing to the browser. This ability was very important at a time when most Internet users were connected through a 28.8 Kbps modem, turning every request to the server into a waiting game. Minimizing the number of times that the user had to wait for a response was the first major step toward the Ajax approach.

Frames

The original version of HTML intended for every document to be standalone, and it wasn't until HTML 4.0 that frames were officially introduced. The idea that the display of a web page could be split up into several documents was a radical one, and controversy brewed as Netscape chose to implement the feature before the HTML 4.0 standard was completed. Netscape Navigator 2.0 was the first browser to support frames and JavaScript together. This turned out to be a major step in the evolution of Ajax.

When the browser wars of the late 1990s began between Microsoft and Netscape, both JavaScript and frames became formalized. As more features were added to both technologies, creative developers began experimenting using the two together. Because a frame represented a completely separate request to the server, the ability to control a frame and its contents with JavaScript opened the door to some exciting possibilities.

The Hidden Frame Technique

As developers began to understand how to manipulate frames, a new technique emerged to facilitate client-server communication. The hidden frame technique involved setting up a frameset where one frame was set to a width or height of 0 pixels, its sole purpose being to initiate communication with the server. The hidden frame would contain an HTML form with specific form fields that could be dynamically filled out by JavaScript and submitted back to the server. When the frame returned, it would call another JavaScript function to notify the calling page that data had been returned. The hidden frame technique represented the first asynchronous request/response model for web applications.

While this was the first Ajax communication model, another technological advance was just around the corner.

Dynamic HTML and the DOM

In 1996, the Web was still mainly a static world. Although JavaScript and the hidden frame technique livened up the user interaction, there was still no way to change the display of a page without reloading it, aside from changing the values contained within form fields. Then came Internet Explorer 4.0.

At this point, Internet Explorer had caught up with the technology of market leader Netscape Navigator and even one-upped it in one important respect through the introduction of Dynamic HTML (DHTML). Although still in the development phase, DHTML represented a significant step forward from the days of static web pages, enabling developers to alter any part of a loaded page by using JavaScript. Along with the emergence of Cascading Style Sheets (CSS), DHTML reinvigorated web development, despite deep disparities between the paths Microsoft and Netscape followed during the early years of each discipline. Excitement in the developer community was justified, however, because combining DHTML with the hidden frame technique meant that any part of a page could be refreshed with server information at any time. This was a genuine paradigm shift for the Web.

DHTML never made it to a standards body, although Microsoft's influence would be felt strongly with the introduction of the Document Object Model (DOM) as the centerpiece of the standards effort. Unlike DHTML, which sought only to modify sections of a web page, the DOM had a more ambitious purpose: to provide a structure for an entire web page. The manipulation of that structure would then allow DHTML-like modifications to the page. This was the next step towards Ajax.

Iframes

Although the hidden frame technique became incredibly popular, it had a downside — one had to plan ahead of time and write a frameset anticipating the usage of hidden frames. When the `<iframe/>` element was introduced as an official part HTML 4.0 in 1997, it represented another significant step in the evolution of the Web.

Instead of defining framesets, developers could place iframes anywhere on a page. This enabled developers to forego framesets altogether and simply place invisible iframes (through the use of CSS) on a page to enable client-server communication. And when the DOM was finally implemented in Internet Explorer 5 and Netscape 6, it introduced the ability to dynamically create iframes on the fly, meaning that a JavaScript function could be used to create an iframe, make a request, and get the response — all without including any additional HTML in a page. This led to the next generation of the hidden frame technique: the hidden iframe technique.

XMLHttp

The browser developers at Microsoft must have realized the popularity of the hidden frame technique and the newer hidden iframe technique, because they decided to provide developers with a better tool for client-server interaction. That tool came in the form of an ActiveX object called XMLHttp, introduced in 2001.

One of the Microsoft extensions to JavaScript allowed the creation of ActiveX controls, Microsoft's proprietary programming objects. When Microsoft began supporting XML through a library called MSXML, the XMLHttp object was included. Although it carried the XML name, this object was more than just another way of manipulating XML data. Indeed, it was more like an ad hoc HTTP request that could be controlled from JavaScript. Developers had access to HTTP status codes and headers, as well as any data returned from the server. That data might be structured XML, pre-formatted swaths of HTML, serialized JavaScript objects, or data in any other format desired by the developer. Instead of using hidden frames or iframes, it was now possible to access the server programmatically using pure JavaScript, independent of the page load/reload cycle. The XMLHttp object became a tremendous hit for Internet Explorer developers.

With popularity mounting, developers at the open source Mozilla project began their own port of XMLHttp. Instead of allowing access to ActiveX, the Mozilla developers replicated the object's principal methods and properties in a native browser object, `XMLHttpRequest`. With both of the major browsers supporting some form of XMLHttp, the development of Ajax-type interfaces really took off and forced the fringe browsers, Opera and Safari, to support some form of XMLHttp as well (both chose to do so natively with an `XMLHttpRequest` object, mimicking Mozilla). Ironically enough, the popularity of this XMLHttp clone reached back to Microsoft, which introduced the native `XMLHttpRequest` object in Internet Explorer 7.

The Real Ajax

Despite the frequently asked questions attached to the end of Garrett's essay, some confusion still exists as to what Ajax really is. Put simply, Ajax is nothing more than an approach to web interaction. This approach involves transmitting only a small amount of information to and from the server in order to give the user the most responsive experience possible.

Instead of the traditional web application model where the browser itself is responsible for initiating requests to, and processing requests from, the web server, the Ajax model provides an intermediate layer — what Garrett calls an *Ajax engine* — to handle this communication. An Ajax engine is really just a JavaScript object or function that is called whenever information needs to be requested from the server. Instead of the traditional model of providing a link to another resource (such as another web page), each link makes a call to the Ajax engine, which schedules and executes the request. The request is done asynchronously, meaning that code execution doesn't wait for a response before continuing.

The server — which traditionally would serve up HTML, images, CSS, or JavaScript — is configured to return data that the Ajax engine can use. This data can be plain text, XML, or any other data format that you may need. The only requirement is that the Ajax engine can understand and interpret the data

When the Ajax engine receives the server response, it goes into action, often parsing the data and making several changes to the user interface based on the information it was provided. Because this process involves transferring less information than the traditional web application model, user interface updates are faster, and the user is able to do his or her work more quickly. Figure 1-1 is an adaptation of the figure in Garrett's article, displaying the difference between the traditional and Ajax web application models.

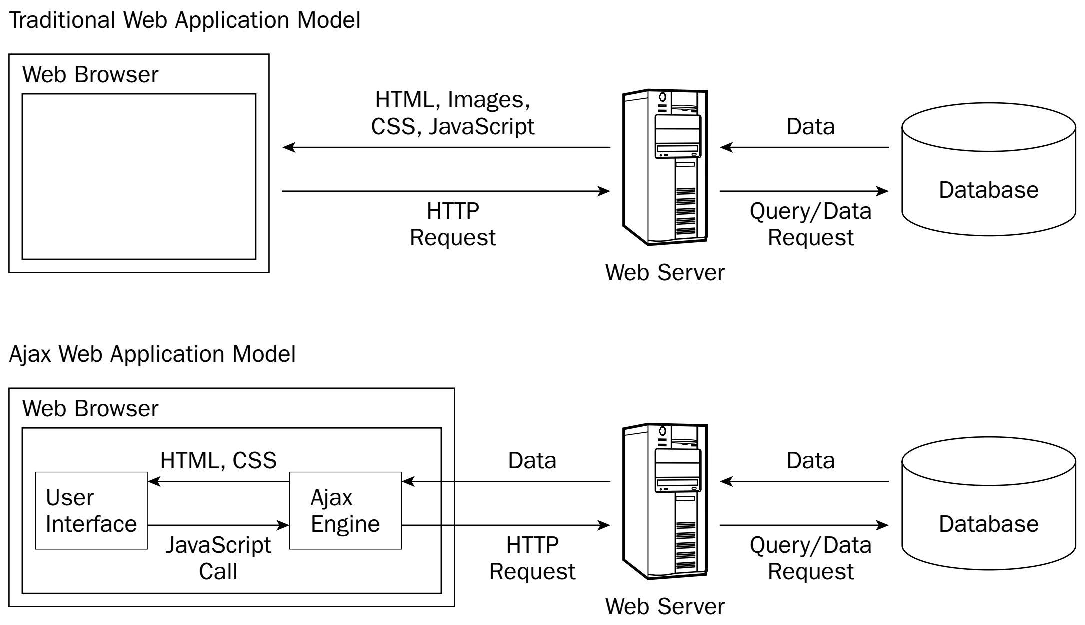


Figure 1-1

Ajax Principles

As a new web application model, Ajax is still in its infancy. However, several web developers have taken this new development as a challenge. The challenge is to define what makes a good Ajax web application versus what makes a bad or mediocre one. Michael Mahemoff (`www.mahemoff.com`), a software developer and usability expert, identified several key principles of good Ajax applications that are worth repeating:

- **Minimal traffic:** Ajax applications should send and receive as little information as possible to and from the server. In short, Ajax can minimize the amount of traffic between the client and the server. Making sure that your Ajax application doesn't send and receive unnecessary information adds to its robustness.
- **No surprises:** Ajax applications typically introduce different user interaction models than traditional web applications. As opposed to the web standard of click-and-wait, some Ajax applications use other user interface paradigms such as drag-and-drop or double-clicking. No matter what user interaction model you choose, be consistent so that the user knows what to do next.
- **Established conventions:** Don't waste time inventing new user interaction models that your users will be unfamiliar with. Borrow heavily from traditional web applications and desktop applications, so there is a minimal learning curve.
- **No distractions:** Avoid unnecessary and distracting page elements such as looping animations and blinking page sections. Such gimmicks distract the user from what he or she is trying to accomplish.
- **Accessibility:** Consider who your primary and secondary users will be and how they most likely will access your Ajax application. Don't program yourself into a corner so that an unexpected new audience will be completely locked out. Will your users be using older browsers or special software? Make sure you know ahead of time and plan for it.
- **Avoid entire page downloads:** All server communication after the initial page download should be managed by the Ajax engine. Don't ruin the user experience by downloading small amounts of data in one place but reloading the entire page in others.
- **User first:** Design the Ajax application with the users in mind before anything else. Try to make the common use cases easy to accomplish and don't be caught up with how you're going to fit in advertising or cool effects.

The common thread in all these principles is usability. Ajax is, primarily, about enhancing the web experience for your users; the technology behind it is merely a means to that end. By adhering to the preceding principles, you can be reasonably assured that your Ajax application will be useful and usable.

Technologies behind Ajax

Garrett's article mentions several technologies that he sees as parts of an Ajax solution. These are:

- **HTML/XHTML:** Primary content representation languages
- **CSS:** Provides stylistic formatting to XHTML

- **DOM:** Dynamic updating of a loaded page
- **XML:** Data exchange format
- **XSLT:** Transforms XML into XHTML (styled by CSS)
- **XMLHttp:** Primary communication broker
- **JavaScript:** Scripting language used to program an Ajax engine

In reality, all these technologies are available to be used in Ajax solutions, but only three are required: HTML/XHTML, DOM, and JavaScript. XHTML is obviously necessary for the displaying of information, while the DOM is necessary to change portions of an XHTML page without reloading it. The last part, JavaScript, is necessary to initiate the client-server communication and manipulate the DOM to update the web page. The other technologies in the list are helpful in fine-tuning an Ajax solution, but they aren't necessary.

There is one major component that Garrett neglected to mention in his article: the necessity of server-side processing. All of the previously listed technologies relate directly to the client-side Ajax engine, but there is no Ajax without a stable, responsive server waiting to send content to the engine. For this purpose, you can use the application server of your choice. Whether you choose to write your server-side components as PHP pages, Java servlets, or .NET components, you need only ensure that the correct data format is being sent back to the Ajax engine.

The examples in this book make use of as many server-side technologies as possible to give you enough information to set up Ajax communication systems on a variety of servers. Most of the examples covered in the book are available in PHP, JSP, and ASP.NET versions at `www.wrox.com`.

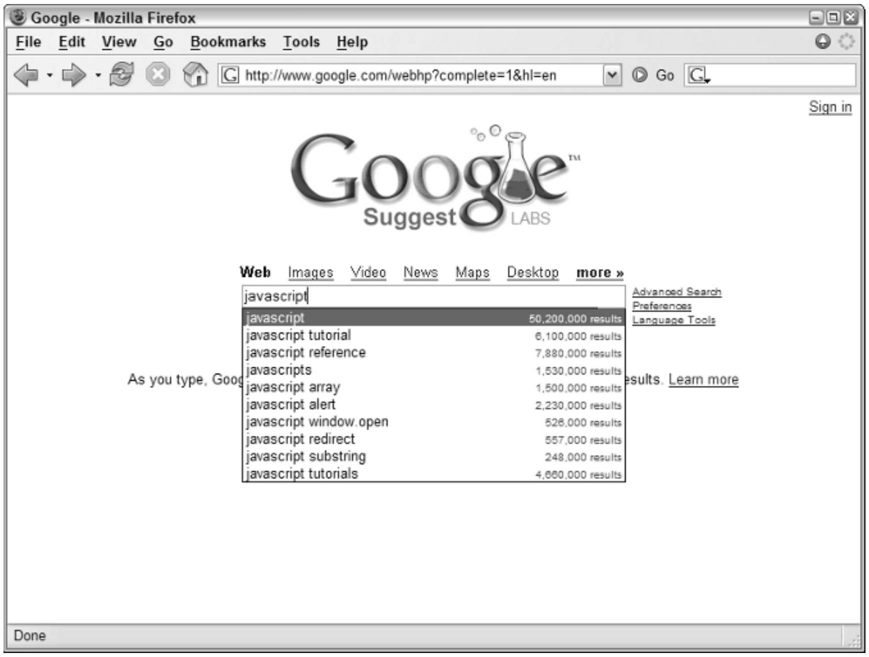
Who Is Using Ajax?

A number of commercial web sites use Ajax techniques to improve their user experience. These sites are really more like web applications than traditional brochureware web sites that just display information because you visit it to accomplish a specific goal. The following are some of the more well-known and well-executed web applications that use Ajax.

Google Suggest

One of the first examples that developers cite when talking about Ajax is Google Suggest (`www.google.com/webhp?complete=1`). The interface is simply a clone of the main Google interface, which prominently features a text box to enter search terms. Everything appears to be the same until you start typing in the textbox. As you type, Google Suggest requests suggestions from the server, showing you a drop-down list of search terms that you may be interested in. Each suggestion is displayed with a number of results available for the given term to help you decide (see Figure 1-2).

This simple client-server interaction is very powerful and effective without being obtrusive to the user. The interface is responsive beyond what you may have learned to expect from a web application; it updates no matter how quickly you type and, as with autocomplete features in desktop software, you can use the up and down arrows to highlight and select each item in the suggestions list. Although still in beta, expect to see this approach make its way into the main Google page eventually.



javascript	50,200,000 results
javascript tutorial	6,100,000 results
javascript reference	7,880,000 results
javascripts	1,530,000 results
javascript array	1,500,000 results
javascript alert	2,230,000 results
javascript window.open	526,000 results
javascript redirect	557,000 results
javascript substring	248,000 results
javascript tutorials	4,660,000 results

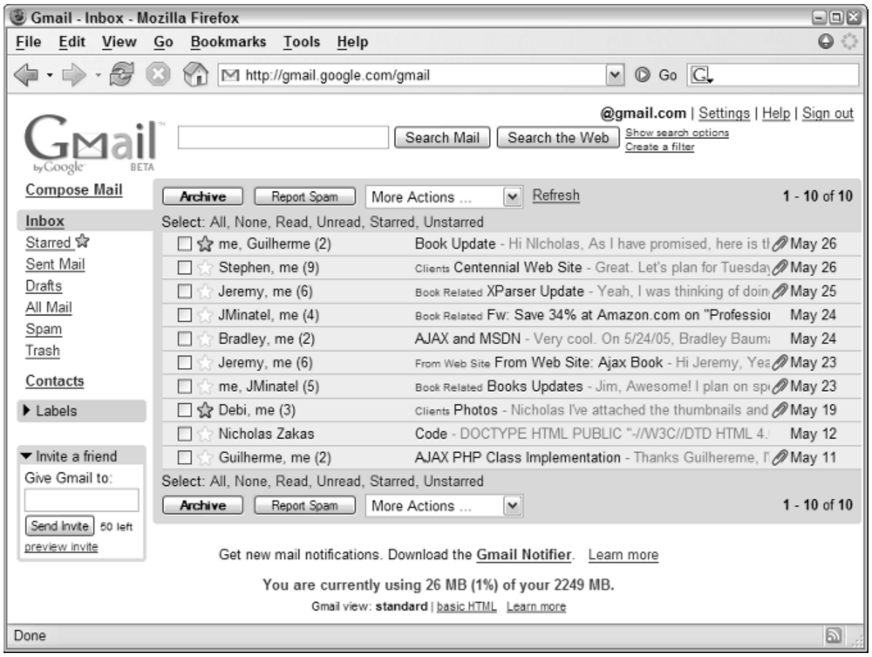
Figure 1-2

Gmail

Gmail, Google's free e-mail service, has been raved about as a marvel of client-server interaction in the age of Ajax. When you first log in to Gmail, a user interface engine is loaded into one of the few iframes the application uses. All further requests back to the server occur through this user interface engine through an XMLHttp object. The data being transferred back and forth is JavaScript code, which makes for fast execution once downloaded by the browser. These requests serve as instructions to the user interface engine as to what should be updated on the screen.

Additionally, the Gmail application uses several frames and iframes to manage and cache big user interface changes. The extremely complicated use of frames enables Gmail to function properly with the Back and Forward buttons, which is one of the advantages of using frames or iframes instead of or in conjunction with XMLHttp (discussed later in the book).

The biggest win for Gmail is its usability. The user interface, as shown in Figure 1-3, is simple and uncluttered. Interaction with the user and communication with the server is all seamless. Once again, Google used Ajax to improve on an already simple concept to provide an exceptional user experience.



me, Guilherme (2)	Book Update - Hi NIcholas, As I have promised, here is tl	May 26
Stephen, me (9)	Clients Centennial Web Site - Great. Let's plan for Tuesday	May 26
Jeremy, me (6)	Book Related XParser Update - Yeah, I was thinking of doin	May 25
JMinatel, me (4)	Book Related Fw: Save 34% at Amazon.com on "Professio	May 24
Bradley, me (2)	AJAX and MSDN - Very cool. On 5/24/05, Bradley Baum	May 24
Jeremy, me (6)	From Web Site From Web Site: Ajax Book - Hi Jeremy, Yes	May 23
me, JMinatel (5)	Book Related Books Updates - Jim, Awesome! I plan on sp	May 23
Debi, me (3)	Clients Photos - Nicholas I've attached the thumbnails and	May 19
Nicholas Zakas	Code - DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.	May 12
Guilherme, me (2)	AJAX PHP Class Implementation - Thanks Guilhereme, I	May 11

Get new mail notifications. Download the Gmail Notifier. Learn more

You are currently using 26 MB (1%) of your 2249 MB.

Gmail view: standard | basic HTML Learn more

Done

Figure 1-3

Google Maps

Another part of Google's dominant Ajax web applications is Google Maps (`maps.google.com`). Designed to compete with well-established mapping sites, Google Maps uses Ajax to avoid reloading its main page at all (see Figure 1-4).

Unlike other mapping web applications, Google Maps enables you to drag the map to move it in various directions. The dragging code is nothing new to JavaScript developers, but the tiling of the map and seemingly endless scrolling effect are another story. The map is broken up into a series of images that are tiled together to make the appearance of a contiguous image. The number of images used to display the map is finite, as creating new images every time the user moves the map would quickly lead to memory problems. Instead, the same images are used over and over to display different segments of the map.

The client-server communication is done through a hidden iframe. Whenever you do a search or ask for new directions, this information is submitted and returned within that iframe. The data returned is in XML format and is passed to a JavaScript function (the Ajax engine) to handle. This XML is then used in a variety of different ways: some is used to call the correct map images, and some is transformed using XSLT into HTML and displayed in the main window. The bottom line is that this complex Ajax application is, as of late 2006, the number two destination for mapping on the Web.

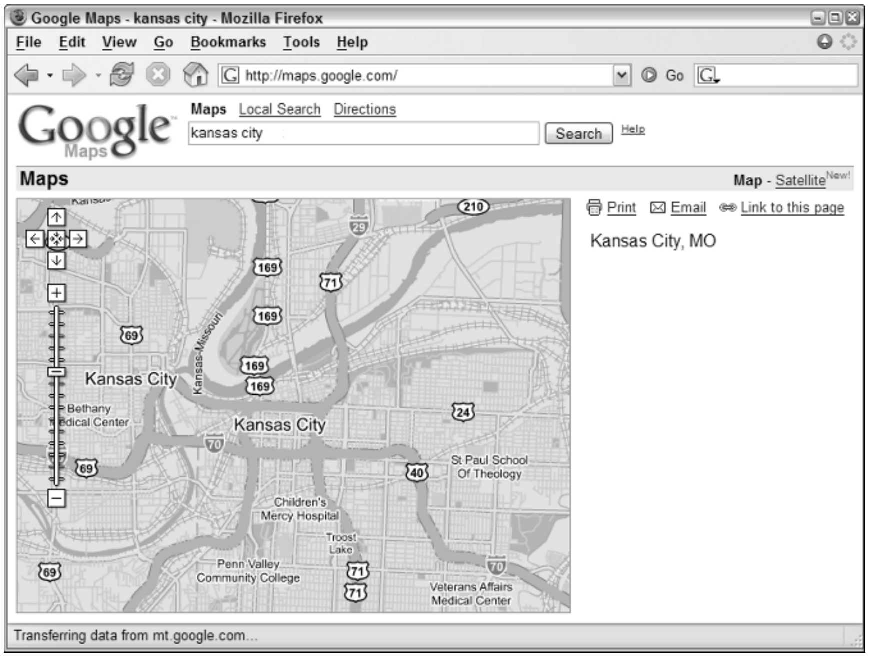

Google Maps - kansas city - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://maps.google.com/
Go
Google Maps
Maps Local Search Directions
kansas city
Search
Help
Maps
Map - Satellite New!
Print Email Link to this page
Kansas City, MO
Kansas City
Kansas City
Bethany Medical Center
St Paul School Of Theology
Children's Mercy Hospital
Troost Lake
Penn Valley Community College
Veterans Affairs Medical Center
Transferring data from mt.google.com...

Figure 1-4

A9

Amazon.com is world famous for being an online marketplace for just about anything, but when it released a search engine, it did so with little fanfare and attention. The introduction of A9 (`www.a9.com`) showed off enhanced searching, enabling you to search different types of information simultaneously. For web and image searches it uses MSN to fetch results. It performs searches of books on Amazon.com and movies on IMDb (Internet Movie Database). Searches for Yellow Pages, Wikipedia, and Answers.com debuted in mid-2005.

What makes A9 unique is how its user interface works. When you perform a search, the different types of results are displayed in different areas of the page (see Figure 1-5).

On the search results page, you have the option of selecting other searches to perform using the same criteria. When you select a check box corresponding to a type of search, the search is performed behind the scenes using a combination of hidden iframes and XMLHttp. The user interface shifts to allow room for the extra search results, which are loaded as soon as they are received from the server. The result is a more responsive search results page that doesn't need to be reloaded when you want to search on different types of information.

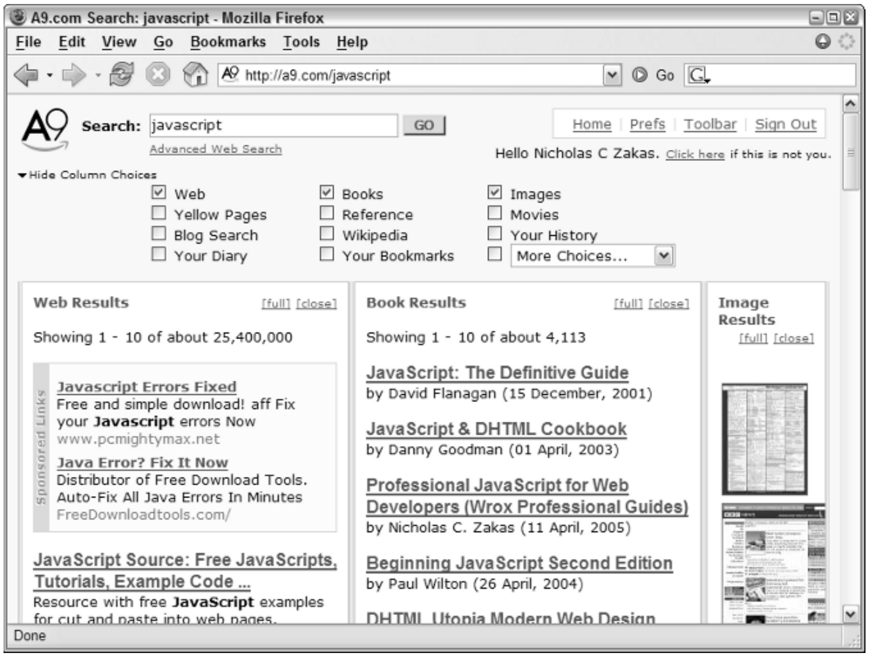
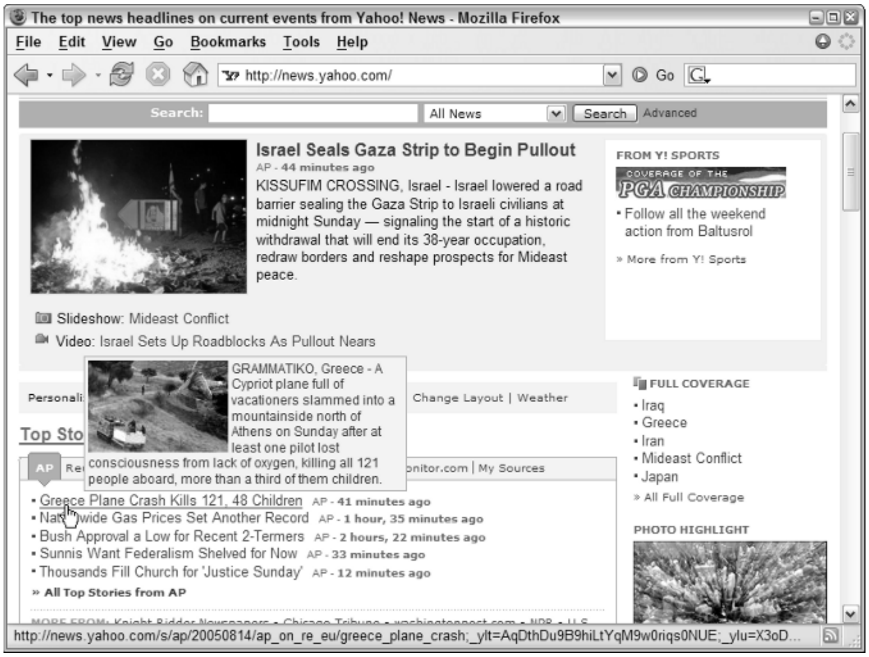

A9.com Search: javascript - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://a9.com/javascript
Search: javascript GO
Advanced Web Search
Home | Prefs | Toolbar | Sign Out
Hello Nicholas C Zakas. Click here if this is not you.
Hide Column Choices
Web
Books
Images
Yellow Pages
Reference
Movies
Blog Search
Wikipedia
Your History
Your Diary
Your Bookmarks
More Choices...
Web Results [full] [close]
Showing 1 - 10 of about 25,400,000
Javascript Errors Fixed
Free and simple download! aff Fix your Javascript errors Now
www.pcmightymax.net
Java Error? Fix It Now
Distributor of Free Download Tools. Auto-Fix All Java Errors In Minutes
FreeDownloadtools.com/
JavaScript Source: Free JavaScripts, Tutorials, Example Code ...
Resource with free JavaScript examples for cut and paste into web pages.
Book Results [full] [close]
Showing 1 - 10 of about 4,113
JavaScript: The Definitive Guide
by David Flanagan (15 December, 2001)
JavaScript & DHTML Cookbook
by Danny Goodman (01 April, 2003)
Professional JavaScript for Web Developers (Wrox Professional Guides)
by Nicholas C. Zakas (11 April, 2005)
Beginning JavaScript Second Edition
by Paul Wilton (26 April, 2004)
DHTML Utopia Modern Web Design
Image Results [full] [close]
Done

Figure 1-5

Yahoo! News

Also introduced in 2005 was a new design for the Yahoo! News site (`news.yahoo.com`). The new design features an interesting enhancement: when you move your mouse over a particular headline, a small box pops up with a summary and, optionally, a photo associated with that story (see Figure 1-6).

The photo information and summary are retrieved from the server using XMLHttp and inserted into the page dynamically. This is a perfect example of how Ajax can be used to enhance a web page. Rather than making Ajax the primary usage mode, the Yahoo! News site is completely usable without Ajax; the Ajax functionality is used only to add a more responsive user experience in browsers that support it. Underneath is a semantically correct HTML page that is laid out logically even without CSS formatting.



Israel Seals Gaza Strip to Begin Pullout
AP - 44 minutes ago
KISSUFIM CROSSING, Israel - Israel lowered a road barrier sealing the Gaza Strip to Israeli civilians at midnight Sunday — signaling the start of a historic withdrawal that will end its 38-year occupation, redraw borders and reshape prospects for Mideast peace.

Slideshow: Mideast Conflict
Video: Israel Sets Up Roadblocks As Pullout Nears

• Follow all the weekend action from Baltusrol
» More from Y! Sports

GRAMMATIKO, Greece - A Cypriot plane full of vacationers slammed into a mountainside north of Athens on Sunday after at least one pilot lost consciousness from lack of oxygen, killing all 121 people aboard, more than a third of them children.

• Greece Plane Crash Kills 121, 48 Children AP - 41 minutes ago
• Nationwide Gas Prices Set Another Record AP - 1 hour, 35 minutes ago
• Bush Approval a Low for Recent 2-Termers AP - 2 hours, 22 minutes ago
• Sunnis Want Federalism Shelved for Now AP - 33 minutes ago
• Thousands Fill Church for 'Justice Sunday' AP - 12 minutes ago
» All Top Stories from AP

• Iraq
• Greece
• Iran
• Mideast Conflict
• Japan
» All Full Coverage

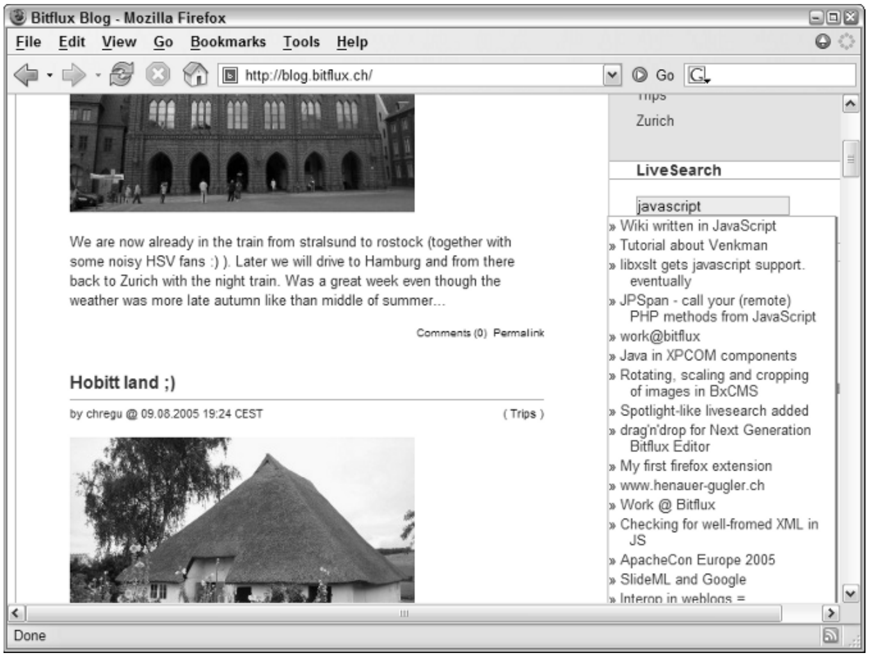
http://news.yahoo.com/s/ap/20050814/ap_on_re_eu/greece_plane_crash;_ylt=AqDthDu9B9hiLtYqM9w0riqs0NUE;_ylu=X3oD...

Figure 1-6

Bitflux Blog

Another great example of using Ajax only as an enhancement is Bitflux Blog (`blog.bitflux.ch`), which features a technology called LiveSearch. LiveSearch works in conjunction with the search box on the site. As you type into the box, a list of possible search results is displayed immediately below (see Figure 1-7).

The search results are retrieved using XMLHttp as an HTML string that is then inserted into the page. You can search the site the old-fashioned way as well: by filling in the text box and pressing Enter. The LiveSearch Ajax functionality is just an enhancement to the overall site and isn't required to search.



We are now already in the train from stralsund to rostock (together with some noisy HSV fans :)). Later we will drive to Hamburg and from there back to Zurich with the night train. Was a great week even though the weather was more late autumn like than middle of summer...

Hobitt land ;)

by chregu @ 09.08.2005 19:24 CEST (Trips)

» Wiki written in JavaScript
» Tutorial about Venkman
» libxslt gets javascript support. eventually
» JPSpan - call your (remote) PHP methods from JavaScript
» work@bitflux
» Java in XPCOM components
» Rotating, scaling and cropping of images in BxCMS
» Spotlight-like livesearch added
» drag'n'drop for Next Generation Bitflux Editor
» My first firefox extension
» www.henauer-gugler.ch
» Work @ Bitflux
» Checking for well-fromed XML in JS
» ApacheCon Europe 2005
» SlideML and Google
» Interop in weblogs =

Figure 1-7

Confusion and Controversy

Despite the popularity of the term *Ajax*, it has been met with its fair share of dissenters and controversy. Some believe that Ajax is an aberration of what the Web was moving toward before Ajax entered the picture. The proponents of semantic HTML design, accessibility, and the separation of content and presentation were gaining ground and acceptance among web developers, and some believe that the popularity of Ajax has pushed that movement into the background. The belief of these detractors is that Ajax promotes creating presentation within JavaScript, thus turning it into a messy mix similar to the early days of server-side scripting. Many believe that accessibility will suffer if more developers turn to Ajax solutions.

Others have spent a significant amount of time dissecting Garrett's article and disproving several assumptions that he makes. For instance, the article mentions using XML and XMLHttp repeatedly as being the core of the Ajax model, but many of the examples he lists don't use them. Gmail and Google Maps use these technologies sparingly; Google Suggest uses only XMLHttp and uses JavaScript arrays instead of XML for data exchange. Critics also point out that the technical explanation of Ajax in the article is completely misleading, citing several technologies that are not only unnecessary (such as XML and XMLHttp) but unlikely to be used in many cases (such as XSLT).

Another big argument surrounding Ajax and Garrett's Adaptive Path article is that it's merely a new name for a technique that has already been used for some time. Although this type of data retrieval could be enacted in Netscape Navigator 2.0, it really became more prominent in 2001–2002, especially with the publication of an article on Apple's Developer Connection site entitled, "Remote Scripting With IFRAME" (available at `http://developer.apple.com/internet/webcontent/iframe.html`). This article is widely believed to be the first mainstream article published on Ajax-like methodologies. The term *remote scripting* never caught on with quite the staying power as Ajax.

Still others scoff at the term *Ajax* and Garrett's article, believing that its creation was little more than a marketing gimmick for Garrett's company, Adaptive Path, LLC. Some believe that creating a name for a technique that already existed is disingenuous and a clear sign of ill intent. Regardless of this and other controversies surrounding Ajax, the approach now has a name that developers are quickly becoming familiar with, and with that comes a need for a deeper understanding and explanation so that it may be used in the best possible ways.

Ajax and Web 2.0

Shortly after the term *Ajax* was coined, another term began popping up. Web 2.0 was originally the name of a conference held by O'Reilly Media and CMP Media in late 2005. After that, the term *Web 2.0* took on a life of its own and began popping up all over the Internet in descriptions of how the Web had changed. To try to rein in the term before it got out of control, Tim O'Reilly (founder and CEO of O'Reilly) wrote an article entitled, "What is Web 2.0" (available online at `www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html`), describing the concepts that he believes Web 2.0 represents. These concepts include:

❑ The Web as services, not software

❑ The group mentality of the Web — users encouraged to participate (as with tagging, blogging, networking, and so on)

❑ Separation of data and presentation – data can be represented in any number of ways and combined with any other data sources (called *mashups*)

❑ Richer, more responsive user experience

Ajax is tied to the last point, creating a richer experience for the user. To be clear, Ajax is not synonymous with Web 2.0, and Web 2.0 doesn't speak just of Ajax; Web 2.0 is about a shift in the very character of the Web. While Ajax is an important part of creating the next generation user experience that Web 2.0 signifies, it is just a one piece of a much larger puzzle.

Summary

This chapter introduced you to the basic premise of Ajax. Short for Asynchronous JavaScript + XML, the term *Ajax* was coined by Jesse James Garrett in an article posted on the Adaptive Path, LLC web site. The article introduced Ajax as a new user interaction model for web applications in which full page loads are no longer necessary.

This chapter also explored the evolution of the Web in relation to the development of technologies that enable Ajax to be a reality today. Ajax owes its existence to the introduction of both JavaScript and frames into web browsers, which made asynchronous data retrieval using JavaScript theoretically possible in Netscape Navigator 2.0. Throughout the evolution of new web technologies, Ajax methodologies such as the hidden frame technique developed. The introduction of iframes and XMLHttp really pushed Ajax development forward.

Although Ajax can be used to accomplish many things, it is best used to enhance the user experience rather than providing cool effects. This chapter discussed several Ajax principles, all circling back to the requirements of the user being paramount to anything else in web application development.

Several of the most popular Ajax applications were also discussed, including Google Suggest, Gmail, Google Maps, Yahoo! News, and the Bitflux Blog.

Finally, the chapter covered the controversy surrounding Ajax, Garrett's article, and Ajax's place on the Web. Some feel that the popularization of Ajax will lead to an overall lack of accessibility, whereas others question Garrett's motive for writing the now-famous article. As with all approaches, Ajax is at its best when used in a logical enhancement to a well-designed web application.