# Chapter 1

# Essential Ajax

Welcome to the *Ajax Bible*! This is the home of all things Ajax. Ajax is the new technology that's blazing through the Internet, igniting Web sites everywhere. The Ajax revolution has come, and it's changing the Internet. In fact, Ajax is the basis of what's being called Web 2.0, the next version of the World Wide Web.

So what's it all about? The central idea is making Web applications look and feel just like desktop applications. For example, take a look at Figure 1.1, where you see the familiar Google search page. Enter a term to search for, such as "Ajax," and click the Google Search button.

## IN THIS CHAPTER

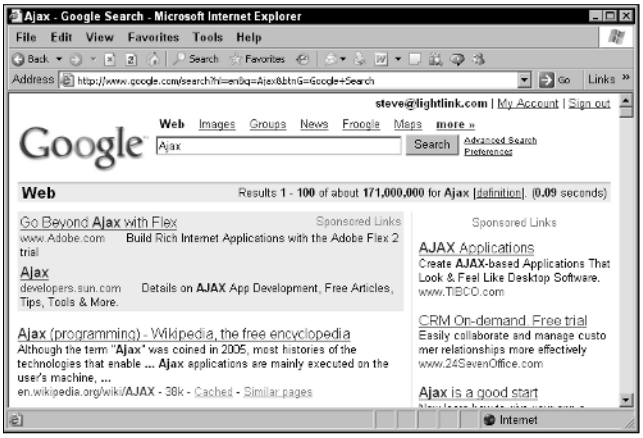
**FIGURE 1.1**

Google searches for matches to the term you enter.


The screen flashes as it's updated with new data, and the matches that Google found to your search term appear, as you see in Figure 1.2.

**FIGURE 1.2**

Google displays the matches it finds.




That works OK, but that's not the Ajax way of doing things. Using Ajax, you can work behind the scenes, connecting to the server to get data without causing a page refresh in the browser. For example, take a look at the Google Suggest page at `www.google.com/webhp?complete=1&hl=en`, which is shown in Figure 1.3.

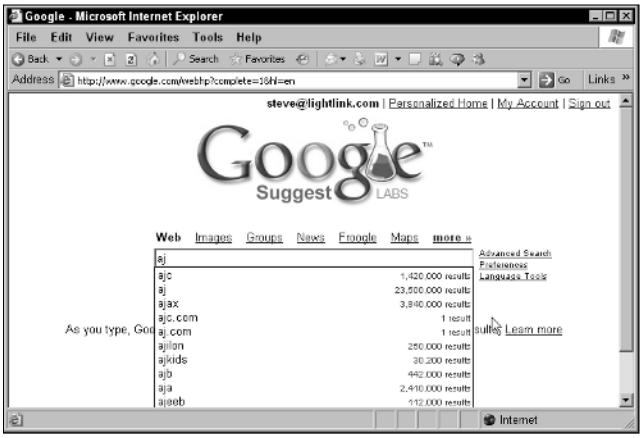
**FIGURE 1.3**

The Google Suggest page


As you type, the page in the browser actually connects to the Google server and looks up matches to the partial search term you entered. For example, type "aj," and you'll see a drop-down list box appear, as in Figure 1.4, with matches found by Google as you're typing.

**FIGURE 1.4**

Google Suggest looks for matches as you type.




Behind the scenes, using Ajax techniques, the Web page connects to Google Suggest and searches for matches to your search term as you're entering it. It then displays a drop-down list box of the matches it's found to your search term, letting you select from those matches — all without a page refresh. That's the crucial point: no page refresh was necessary. In the old days, when you wanted

to send data to the server, you had to click a button, such as the Google Search button. Then you had to wait as the screen flickered and was refreshed. Now, a Web page can send data to the server without creating a page refresh at all, as you see in this example, where your search term was sent to the server automatically and the server sent back data to be displayed in the drop-down list.

No longer do you need to perform a page refresh when you send data to the server, or when you receive data from the server. Instead, Web pages can now act much more like desktop applications, sending data to the server and receiving data back, all behind the scenes.

This conversion of Web applications, making them feel more like desktop applications, is what's meant by Web 2.0. How would you like it if your word processor flashed every time you typed a new character, and the entire document was displayed over again, with the cursor reset to the beginning of the document? Not a very attractive thought. Using Ajax, you can create online word processors that are practically indistinguishable from the desktop version — no flash, no flicker, no resetting the cursor location when you type. Just a smooth word-processing experience, just like the desktop version of the same application.

You can see why Ajax is causing a revolution in Web applications: now it's possible to create online applications that look and feel just like their desktop counterparts.

This chapter gets you started. You'll get an overview of the meaning of the term Ajax, and then a survey of how Ajax is used today. That survey is a very important part of this book because Ajax is turning up in more and more places — sometimes unexpectedly — and if you are familiar with the uses of Ajax, you'll know where you can use it in your own Web applications.

# What Does "Ajax" Mean?

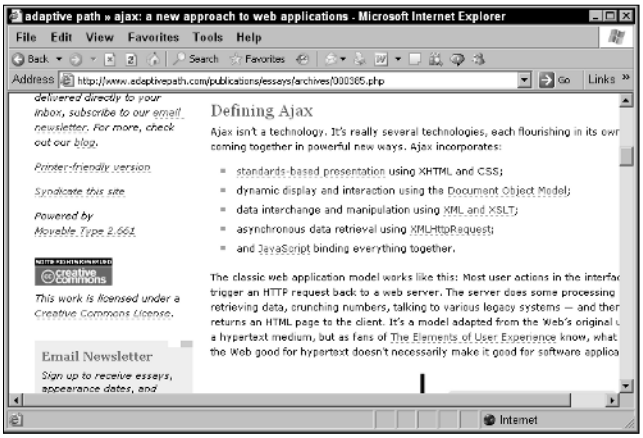
So where did the term "Ajax" come from, exactly? Take a look at Figure 1.5, which shows the very important first article written on Ajax, the article that coined the term and started everything. You can find that article at `www.adaptivepath.com/publications/essays/archives/000385.php`. This article is by Adaptive Path's Jesse James Garrett, who was the first to call this technology Ajax.

Here's how that article starts:

> "If anything about current interaction design can be called 'glamorous,' it's creating Web applications. After all, when was the last time you heard someone rave about the interaction design of a product that wasn't on the Web? (Okay, besides the iPod.) All the cool, innovative new projects are online.
>
> "Despite this, Web interaction designers can't help but feel a little envious of our colleagues who create desktop software. Desktop applications have a richness and responsiveness that has seemed out of reach on the Web. The same simplicity that enabled the Web's rapid proliferation also creates a gap between the experiences we can provide and the experiences users can get from a desktop application.
>
> "That gap is closing."

**FIGURE 1.5**

The original Ajax article




That gap is indeed closing, thanks to Ajax. So, according to the article, what does "Ajax" actually stand for? It stands for *Asynchronous JavaScript and XML*. As you can begin to see from its name, and as you can read in the Jesse James Garrett article, Ajax is really a collection of technologies.

The "asynchronous" part means that the browser isn't going to wait for data to be returned from the server, but can handle that data as it's sent back, when it's sent back. In other words, data transfers take place behind the scenes, without making the browser pause and wait for something to happen. That's a crucial part of Ajax: You can handle data from the server when the server sends you that data. You don't have to put your whole application on hold until that data arrives. If you had to wait for that data, your application would be synchronous; and with slow Internet connections, that could be a problem.

The JavaScript part of the term Ajax is also very important because that's what makes Ajax happen in the browser. Ajax relies on JavaScript in the browser to connect to the server and to handle the data that the server sends back. All the Ajax applications you will develop in this book use JavaScript to connect to the server behind the scenes, uploading and downloading data. And when your data is downloaded, you can use JavaScript in the browser to handle that data, displaying it or crunching it as appropriate.

What about the XML part of the term Ajax? As you probably know, XML has become the lingua franca of the Web, providing a text-based way to send data back and forth across the Internet. The reason XML has become so popular is that it is indeed text-based, which means that you can sling XML around the Internet, because the Internet was designed to handle text-based documents (that is, HTML). For that reason, Ajax applications are often written to handle data sent back from the server using XML. In other words, when you contact the server, it'll send data back to you as an XML document.

In fact, XML is only one of the ways to handle data sent to you from the server. You can also send back plain text as well, and you're going to see both techniques extensively in this book.

Besides JavaScript and XML, Ajax also works with dynamic HTML and Cascading Style Sheets (CSS). Both of these technologies allow you to update the data displayed in a Web page, and, because you don't redraw the entire Web page with Ajax, but just a part of it, you rely on dynamic HTML and CSS quite a bit; both of them allow you to update specific parts of a Web page. You're going to see a lot more on dynamic HTML and CSS in this book because they allow you to refresh just part of a Web page, something that is central to Ajax-enabled applications.

The part of JavaScript that makes Ajax possible is the `XMLHttpRequest` object. This is a special object built into all modern browsers' version of JavaScript. As you're going to see, this is what makes it possible to connect to the server and handle data sent back from the server behind the scenes. It's not just JavaScript that makes Ajax tick, it's the `XMLHttpRequest` object inside JavaScript.

So there you have it; Ajax is a collection of technologies, not just a single technology. You use the `XMLHttpRequest` object built into JavaScript to connect to the server, and then handle the XML — or plain text — the server sends back using JavaScript. And you use dynamic HTML and CSS to display the results in the browser. It's lucky that all the parts of Ajax applications came together as they did — JavaScript, the `XMLHttpRequest` object, dynamic HTML, and CSS — because all together, they make it possible to make your online applications look like desktop applications.

Actually, the technology for Ajax has been around since 1998, and had already been used by a number of applications such as Microsoft's Outlook Web Access. But it didn't really catch on until early 2005 when some high-profile applications such as Google Suggest put it to work, and Jesse James Garrett wrote his article coining the term Ajax, which put everything under one roof.

Since that time, things have exploded as developers have realized that Web software can finally start acting and behaving like desktop software. So what can you do with Ajax? That's what the rest of this chapter is about.

# What Can You Do with Ajax?

There's a great deal you can do with Ajax, and the following pages cover this treasure trove in some detail. Coming up is a good survey of the way Ajax is used today.

## Create Ajax live searches

One of the most popular uses of Ajax is to create ***live searches***, and you've already seen an example with Google Suggest at the beginning of this chapter. With a live search, the user can enter a partial search term, and using Ajax, the Web application connects to the server and finds matches to that partial search term.

There are plenty of live searches besides Google Suggest available online. For example, take a look at Gollum at `http://gollum.easycp.de/en/`, which is a live search of Wikipedia, the online free encyclopedia at `www.wikipedia.org`. Gollum is shown in Figure 1.6.

**FIGURE 1.6**

Gollum performs live searches of Wikipedia.


Enter a partial search term in Gollum, such as "Aja" for Ajax, and you can see the results in Figure 1.6, where Gollum has connected to Wikipedia behind the scenes and found matches to your partial search term. Those matches are displayed, as is usual for a live search, in a drop-down list, and you can select the one that you're looking for. When you do, the matching Wikipedia article is opened in your browser.

## Create an Ajax-enabled calculator

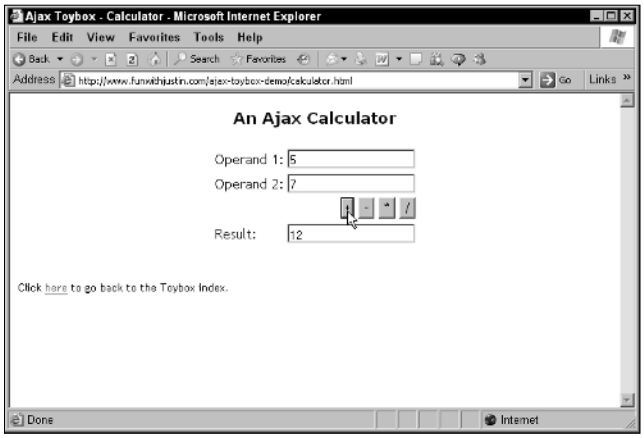
Any situation where you have to send data to the server and handle the data sent back to you behind the scenes is perfect for Ajax. So how about an Ajax-enabled calculator? You can find one at `www.funwithjustin.com/ajax-toybox-demo/calculator.html`, as shown in Figure 1.7.

To use the calculator, just enter two operands to work with, such as 5 and 7 in Figure 1.7, and click the operation you want to perform — addition, subtraction, multiplication, or division. Using Ajax, this Web page sends your operands to a program on the server that adds, subtracts, multiplies, or divides your numbers as appropriate and sends the results back.

The results then appear in the bottom text field, as you can see in Figure 1.7, where 5 and 7 are added. And it's all done without a page refresh — no browser flicker. This application, like other Ajax applications, looks just as if it's a desktop application.

**FIGURE 1.7**

An Ajax-enabled calculator




## Talk with Ajax chat applications

Ajax is great anywhere intensive updating is required, such as chat applications, where any number of users can type and their text appears automatically to everyone currently logged in. Ajax is a good choice here because the text being displayed is always being updated, and having to watch it flicker as the whole page is updated would be very annoying.

Using Ajax, however, you can update text anywhere in a page easily, no page refresh required. Take a look, for example, at `www.phpfreechat.net/demo.en.php`, the PHP Free Chat page. This page connects to a PHP script on the server to support a chat application. When you first navigate to PHP Free Chat, it asks you to enter a username, as you see in Figure 1.8.

**FIGURE 1.8**

Signing in for PHP Free Chat


After you've signed in, you can type your text in the text field that appears at the bottom of Figure 1.9; when you press Enter, that text is sent, using Ajax, to the server, which adds that text to the text that others have typed, and the results appear in the chat box, as you can see in Figure 1.9.

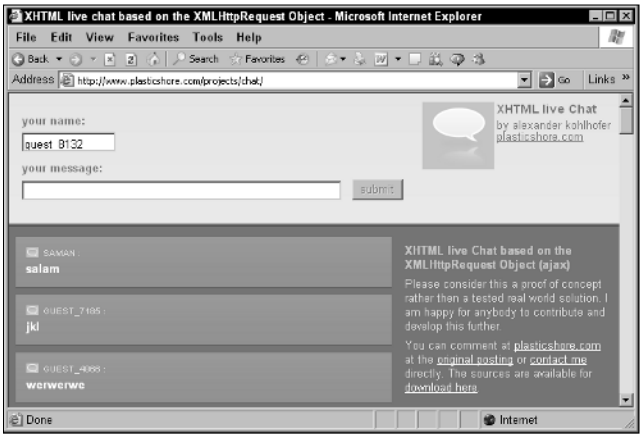
**FIGURE 1.9**

PHP Free Chat lets you enter text that others can see.


There are many Ajax chat applications around. Take a look at `www.plasticshore.com/projects/chat/`, for example, which is shown in Figure 1.10. To use this chat application, all you have to do is enter your name (or accept the default name) and your text, and click the Submit button. When you do, your text appears in the chat box, along with everyone else's.

**FIGURE 1.10**

An Ajax-enabled free chat application



**NOTE** **There are many more Ajax-enabled chat applications. For example, take a look at `http://treehouse.ofb.net/chat/?lang=en` for another good one.**

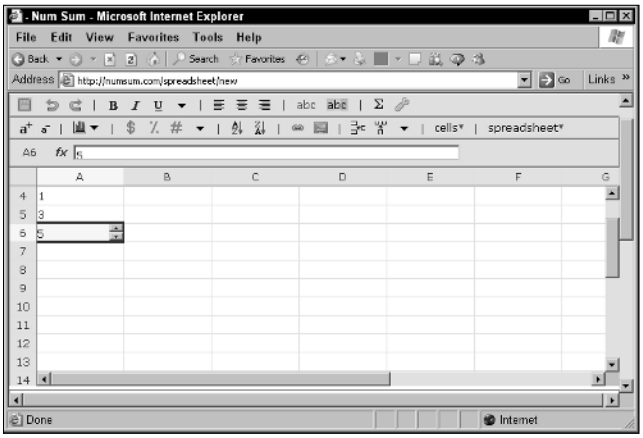
## Crunch numbers with spreadsheets

More and more desktop-type applications are being migrated to Web, thanks to Ajax. One of the latest is for spreadsheets, which you can now find in a number of places online.

For example, take a look at Num Sum, a free online spreadsheet that works just as a desktop version would, at `http://numsum.com/spreadsheet/new`. You can see Num Sum at work in Figure 1.11.

**FIGURE 1.11**

Num Sum is an online Ajax spreadsheet application.



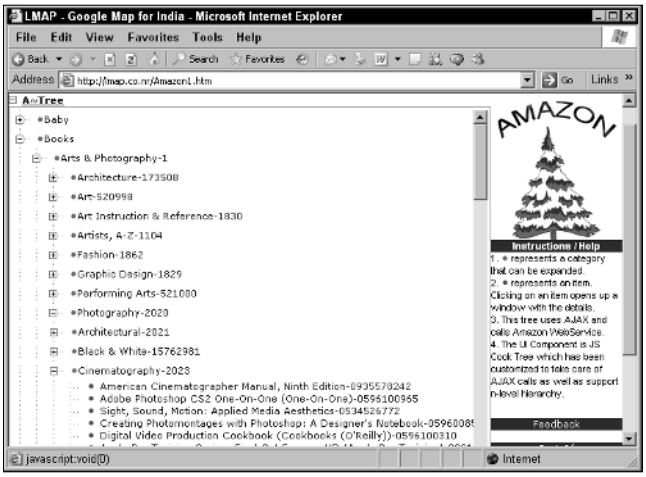

Using Num Sum, you can create real spreadsheets, including the use of formulas, and save your data on the Num Sum server. Using this application is nearly indistinguishable from a desktop version, as you enter data and watch it being updated — all without a browser refresh.

## Browse Amazon

Here's a cute one: an application that lets you browse through the products for sale at Amazon.com. This application lets you display everything in Amazon.com using a clickable tree that is updated using Ajax. Just navigate to `http://lmap.co.nr/Amazon1.htm` and click a node to open that node and see the Amazon products, as shown in Figure 1.12.

**FIGURE 1.12**

Browsing through Amazon.com




## Get the answer with Ajax autocomplete

Ajax autocomplete applications are a lot like live searches. With live searches, you can enter a partial term in a text field. With autocomplete, however, there's no search involved; autocomplete simply offers suggestions to complete the term you're typing (if you do any cell phone text messaging, you're probably familiar with the idea — many cell phones offer suggestions to complete a term as you're typing it).

You can see an autocomplete example at `www.papermountain.org/demos/live/`, which is shown in Figure 1.13.

Just type a partial English word in the text field as shown in Figure 1.13, and the application sends your partial word to the server, which finds matches to that word and sends back autocomplete suggestions. Those suggestions appear, as you see in Figure 1.13, and you can select among them. When you do, the term you select replaces the partial term you've already typed.

You can find another autocomplete example available for download from SWATO, an Ajax toolkit, at `https://swato.dev.java.net/`. You can see this example at work in Figure 1.14.

This example matches the names of countries, as shown in Figure 1.14. All you need to do is to type, say, "A", to be shown a set of possible completions, including Algeria, Austria, Australia, and so on.

**FIGURE 1.13**

Using autocomplete

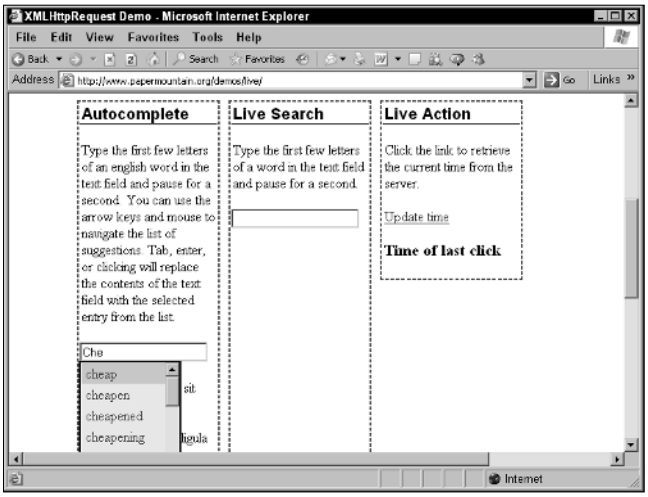
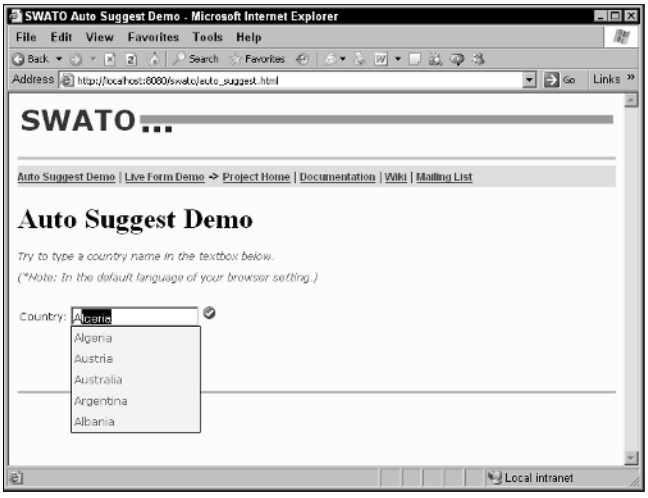



**FIGURE 1.14**

Using SWATO for autocomplete



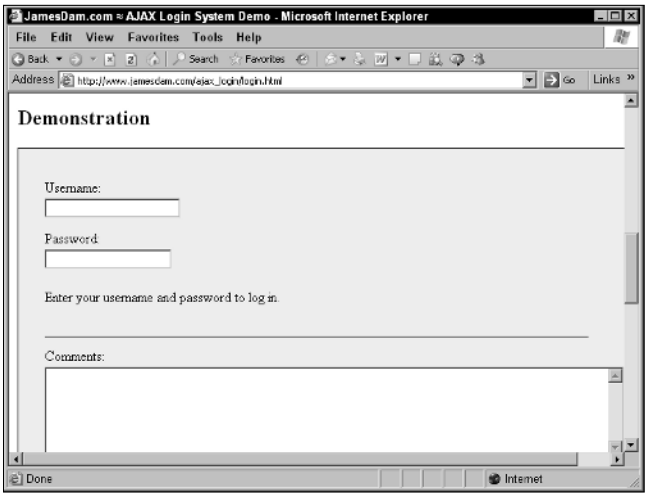
## Log in with Ajax

Here's another good one: rather than asking the users to navigate through a couple of pages to log in to a site, you can use Ajax to make the process easier, checking their typed username and password behind the scenes.

For example, take a look at `www.jamesdam.com/ajax_login/login.html`, which is shown in Figure 1.15. This page lets you log in automatically using Ajax, no page refresh required.

**FIGURE 1.15**

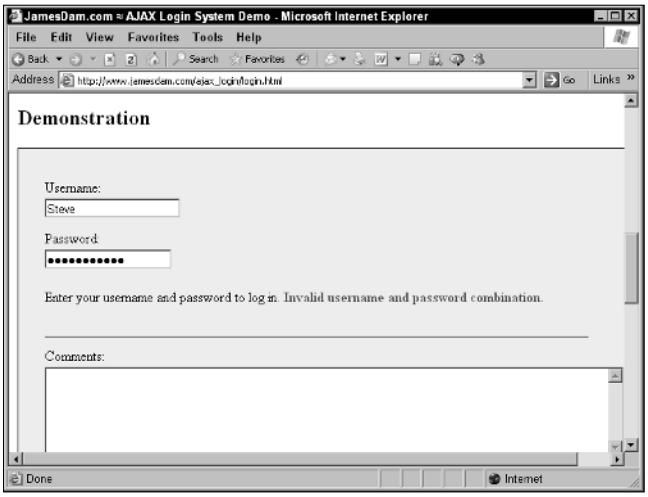
An Ajax-enabled login page




If you enter an incorrect username and password, such as *Steve* and *opensesame* and click the page anywhere, you'll see an error message, as shown in Figure 1.16.

**FIGURE 1.16**

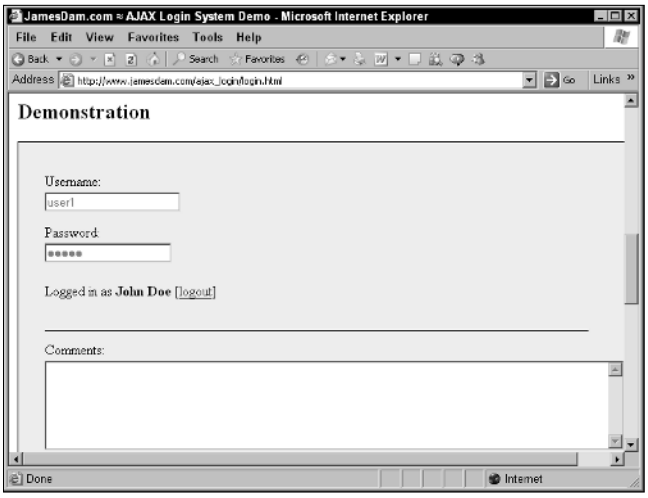
The login is blocked.




On the other hand, if you enter a correct username and password — user1 and pass1 works here — and click the page, you'll see that you're logged in, as shown in Figure 1.17.

**FIGURE 1.17**

A successful Ajax-enabled login

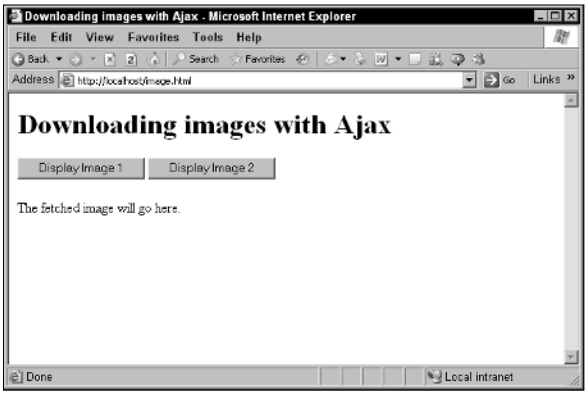


## Download images

You can download only text and XML using Ajax. Or can you? One of the examples you're going to see in this book lets you use Ajax together with dynamic HTML to download images. This example is shown in Figure 1.18.

**FIGURE 1.18**

In this example you can download images.




When you click a button, this example downloads and displays an image, as shown in Figure 1.19. So is Ajax really downloading that image? Isn't Ajax limited to downloading text and XML?

**FIGURE 1.19**

In this example you can download a new image without refreshing the page.

What really happens in this example is that Ajax downloads the *name* of the new image to display. Then, the example uses JavaScript to rewrite an HTML `<img>` element in the Web page, using the name of the file to download. When the browser sees that the `<img>` element has been rewritten, it downloads the image the `<img>` element references, through the magic of dynamic HTML.

The end result is that you click a button and a new image appears, no browser refresh needed. That's a combination of Ajax and dynamic HTML at work, and it indicates that you will indeed be able to download binary data using Ajax in this book.

## Drag and drop with Ajax

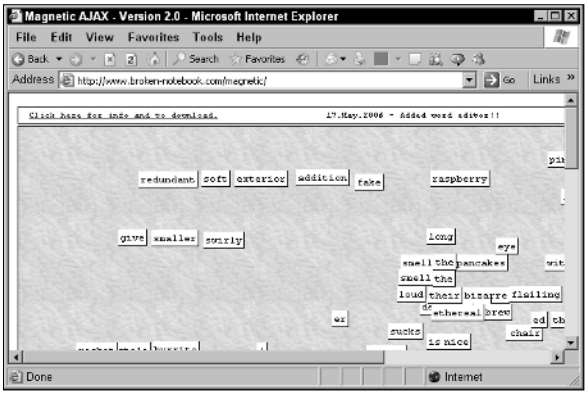
As Web applications become more and more like desktop applications, more and more of what you take for granted in desktop applications is going to start showing up in online applications. For example, drag-and-drop operations can make life a lot easier for the user; and now, when you drag-and-drop items in a Web page, the server can be notified of what you've done behind the scenes. That means the next time you take a look at the page, what you've dragged and dropped appears in the new position you've placed it.

For example, take a look at the refrigerator magnet words at `www.broken-notebook.com/magnetic/`, shown in Figure 1.20.

When you drag a "magnet" to a new location, that new location is sent to the server using Ajax techniques, and that location data is stored. Other people navigating to the page see the magnets in the locations you have set them, and when you come back to the page, the magnets will be where you placed them — unless someone has already moved them.

**FIGURE 1.20**

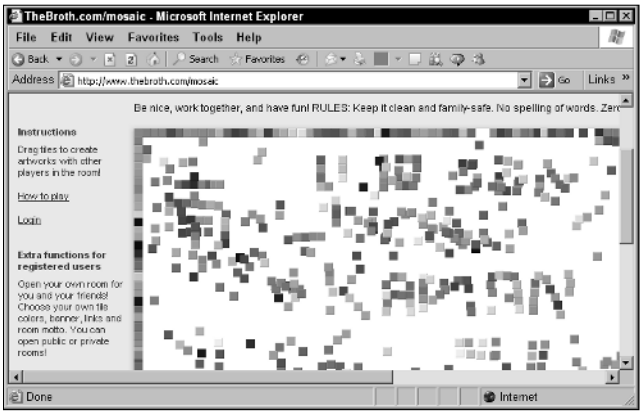
The refrigerator magnet words stay where you placed them unless someone else comes along and moves them.



Here's another drag-and-drop example: the mosaic at `thebroth.com`. The idea here is that you and others can drag tiles to create a shared artwork, using Ajax. When you drag a tile, its new location is sent to the server using Ajax, and the tile's position is updated everywhere, in everyone's browser. You can find the mosaic at `www.thebroth.com/mosaic`, and it is shown in Figure 1.21.

**FIGURE 1.21**

Creating a shared mosaic




Drag and drop doesn't always have to do with individual items. For example, take a look at Ideo Technologies' datagrid in Figure 1.22, which is located at `http://sweetdev-ria.ideotechnologies.com/sweetdev-ria-demo-1.1-RC3/welcome.do`. The datagrid control lets you rearrange columns by dragging them, as shown in the figure.

When you drag a column, the new column arrangement is sent to the server and stored, which means that when you navigate to other pages for the same datagrid (using the number links under the datagrid), that arrangement is preserved.

One of the biggest uses of dragging and dropping with Ajax is to implement shopping carts. Normally, when you want to add an item to a shopping cart online, you have to go through several pages: when you click the Add to Cart button, you then see a new page corresponding to the shopping cart, and then you must navigate backward to continue shopping.

Wouldn't that be much easier if you never had to leave the page you were shopping on? What if you could simply drag an item to the shopping cart, and the server was notified behind the scenes of your purchase? No fuss, no muss.

**FIGURE 1.22**

Dragging a column in Ideo Technologies' datagrid


That's the way it works with the shopping cart at `www.puterpet.com/index.php?sel=crafted&menu=crafted&selection=rocks`, which you can see in Figure 1.23.

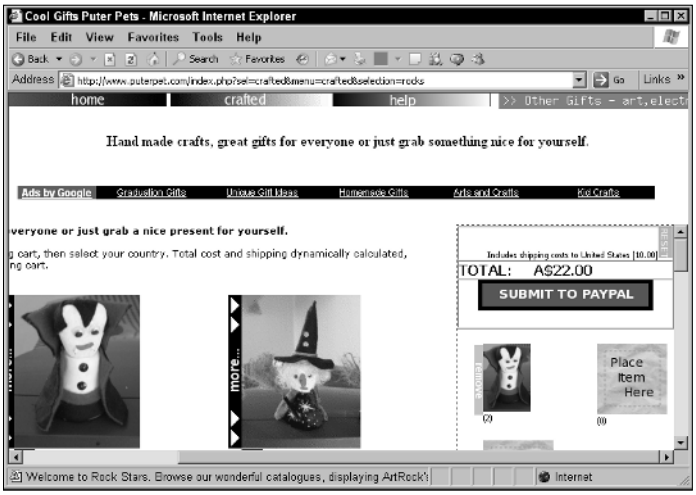
When you drag and drop an item to the Ajax-enabled shopping cart right in the page, the server is notified of your purchase, and the current total appears in the shopping cart, as shown in Figure 1.24.

**FIGURE 1.23**

Dragging an item to an Ajax-enabled shopping cart

**FIGURE 1.24**

Adding an item to an Ajax-enabled shopping cart

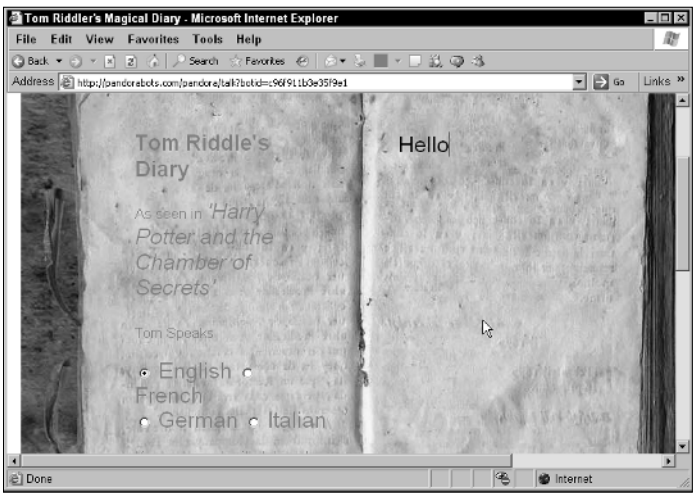



## Play games with Ajax

Here's another fun use of Ajax: a Harry Potter–based "diary" that answers back what you type using Ajax. You can find this one at `http://pandorabots.com/pandora/talk?botid=c96f911b3e35f9e1`, as shown in Figure 1.25.

**FIGURE 1.25**

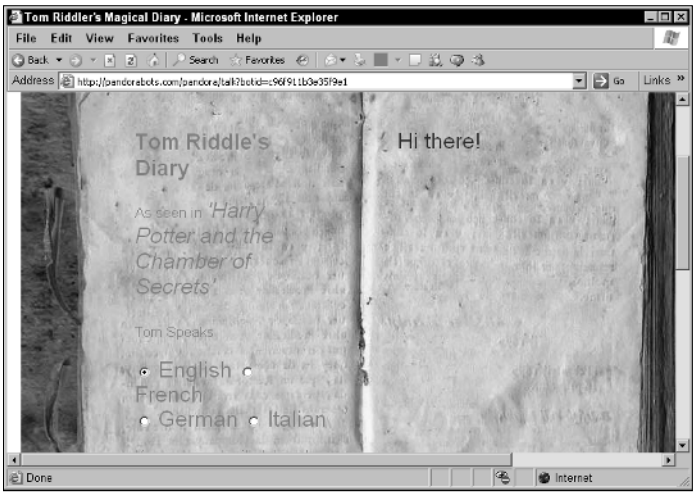
Using the Harry Potter–based diary



Try typing "Hello" in the diary. The word "Hello" appears momentarily, then disappears, followed by the diary's response — fetched using Ajax — which you see in Figure 1.26.

**FIGURE 1.26**

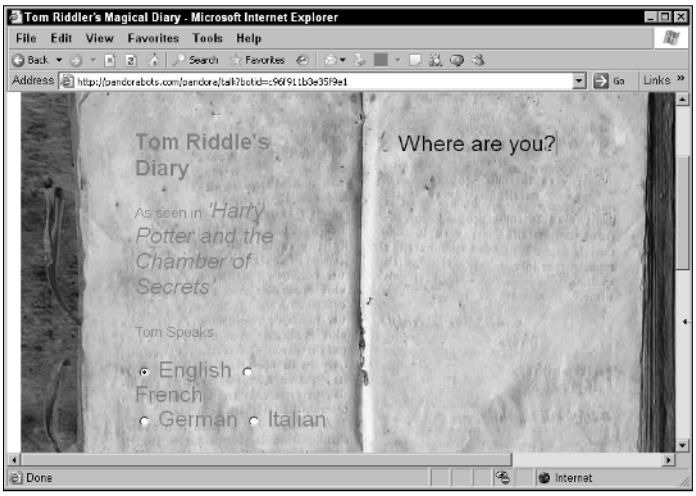
The diary responds.




You can ask quite advanced questions of the diary, as shown in Figure 1.27, in which the diary is being asked where it is.

**FIGURE 1.27**

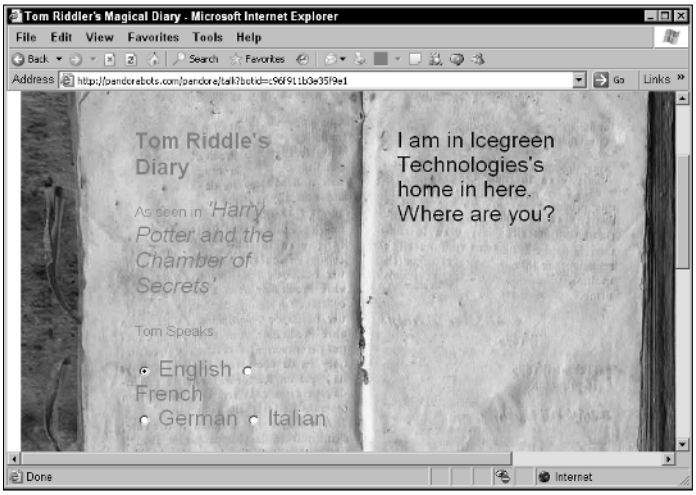
Asking the diary where it is



And you can see the diary's response in Figure 1.28.

**FIGURE 1.28**

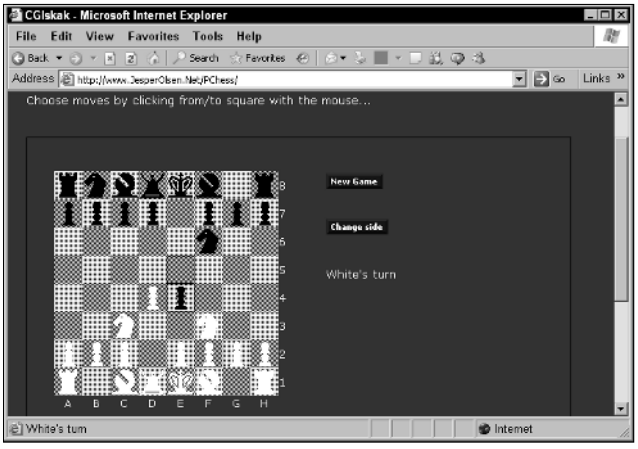
The diary indicates where it is.




How about a game of Ajax-enabled chess? Take a look at `www.JesperOlsen.Net/PChess/`, which is shown in Figure 1.29. To move a piece, you have only to click it, then click its new position, and the piece is moved automatically. The game sends all the needed data to the server using Ajax, behind the scenes, and gets the data it needs back. Then it uses CSS to move its piece accordingly.

**FIGURE 1.29**

Playing Ajax chess



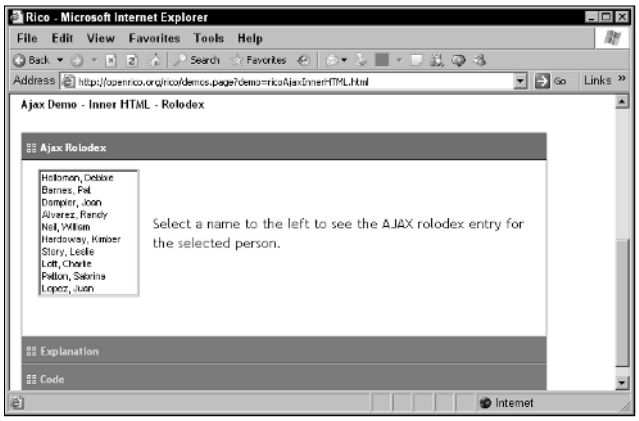
## Modify Web pages on the fly

One of the things that Ajax is big on is updating or modifying a Web page as the user watches. Because Ajax applications avoid complete page refreshes, along with the accompanying flicker and flash, you must update specific parts of a Web page in the browser. There are thousands of Ajax applications out there that operate this way, and you're going to be creating some in this book as well.

Here's an example already online: `http://openrico.org/rico/demos.page?demo=ricoAjaxInnerHTML.html`, which you can see in Figure 1.30.

**FIGURE 1.30**

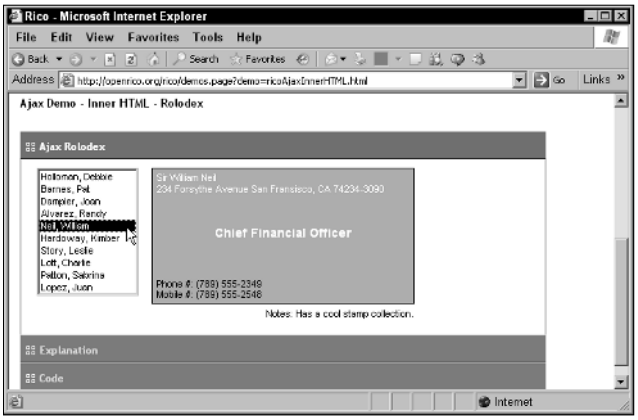
An Ajax Rolodex




This example is an Ajax-enabled Rolodex. You just have to click a person's name and his or her "card" of information appears, as you see in Figure 1.31.

**FIGURE 1.31**

Using an Ajax Rolodex



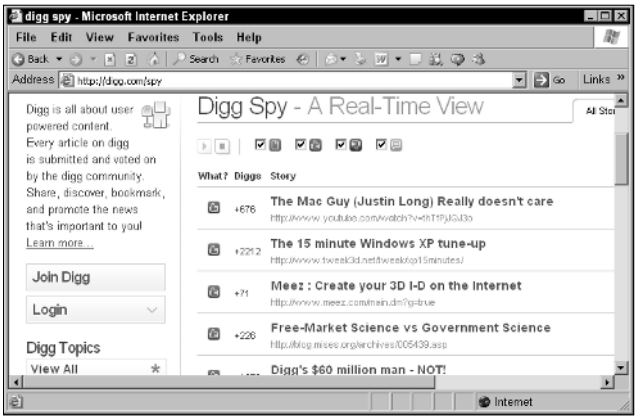

This Rolodex works using CSS to display its data. Each time you click a person's name, the application fetches that person's data from the server, puts together a Rolodex card, and then displays that card using CSS to modify the Web page on the fly.

That's the way it works: You display the results of your actions in a Web page at run time. You don't cause a page refresh to happen.

Another example that shows how to modify a Web page is located at `http://digg.com/spy`, which is shown in Figure 1.32. This site presents news articles that users vote on in real time, and constantly updates itself using a combination of CSS and dynamic HTML. The list of articles you see displayed keeps changing as users vote on them, and each article gets a thumbs-up or thumbs-down icon. It's all done using Ajax to fetch data from the server without causing a page refresh.

**FIGURE 1.32**

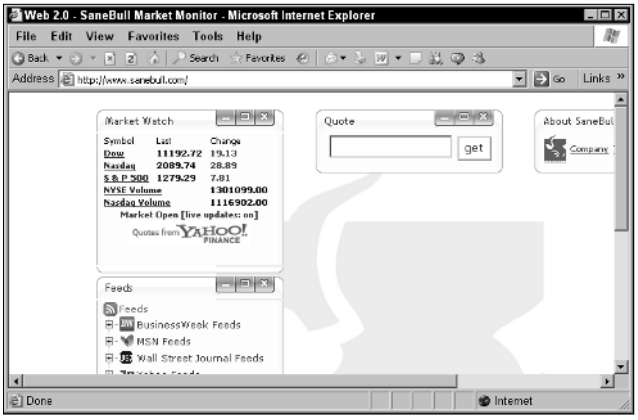
The Digg Spy application updates itself constantly.




The SaneBull Market Monitor is another example that uses Ajax to refresh itself continuously. This page is located at `www.sanebull.com` and is shown in Figure 1.33.

**FIGURE 1.33**

SaneBull constantly updates its stock ticker using Ajax.

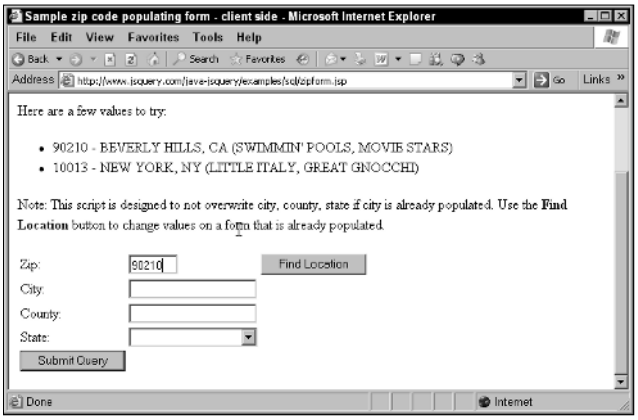


As you watch, the SaneBull monitor updates the Market Watch information at left in the Web page, giving you market quotes by modifying the Web page.

Ajax applications sometimes work by updating HTML controls in the Web page, not just the display in the page itself. You can see an example at `www.jsquery.com/java-jsquery/examples/sql/zipform.jsp`, which is a reverse ZIP code finder. You enter a ZIP code, and the application tells you what city and state the ZIP code is for. You can see the reverse ZIP code finder at work in Figure 1.34.

**FIGURE 1.34**

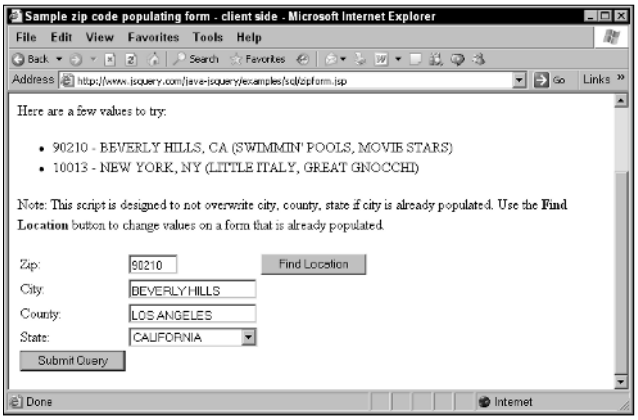
A reverse ZIP code finder




When you click the Find Location button, the city, county, and state matching the ZIP code you've entered appears in the text fields in the Web page, as shown in Figure 1.35, through the magic of Ajax and dynamic HTML.

**FIGURE 1.35**

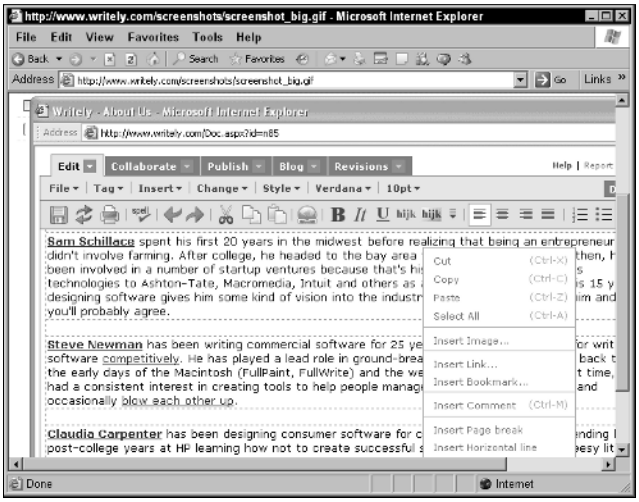
Using the reverse ZIP code finder




One of the all-time champs of Web page modification, and an application that no review of Ajax should omit, is Writely at `http://docs.google.com`. Writely is a complete online word-processing application of the kind made possible by Ajax, and you can see a sample from Writely's Web site in Figure 1.36.

**FIGURE 1.36**

A sample Writely page



Writely lets you create and edit documents just as if you were using it on a desktop, and it relies heavily on Ajax to avoid page refreshes as the user works on a document. Writely is another of those applications making the migration from the desktop to the Web. You've already seen an online spreadsheet application; now you've seen an online word processor — both usually desktop applications, now possible online.

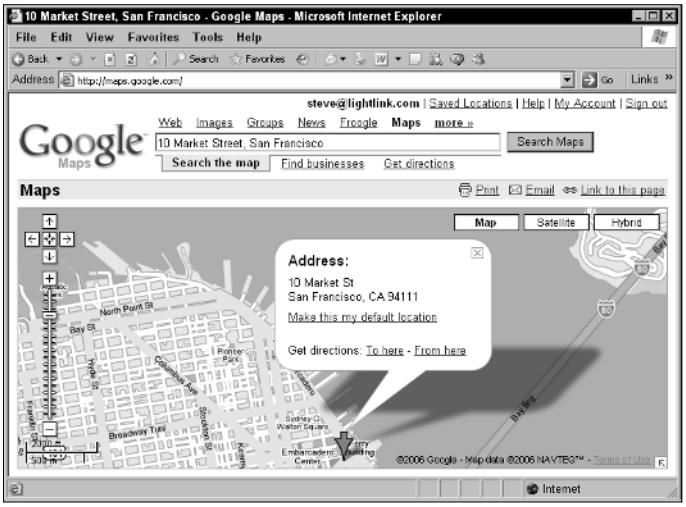
## Add interactivity to maps

One of the first Ajax applications that started it all is Google Maps, at `http://maps.google.com`, which in Figure 1.37 is zooming in on 10 Market Street in San Francisco.

See that popup and the arrow pointing to 10 Market Street? The data for those items is downloaded using Ajax behind the scenes in Google Maps. Cool.

**FIGURE 1.37**

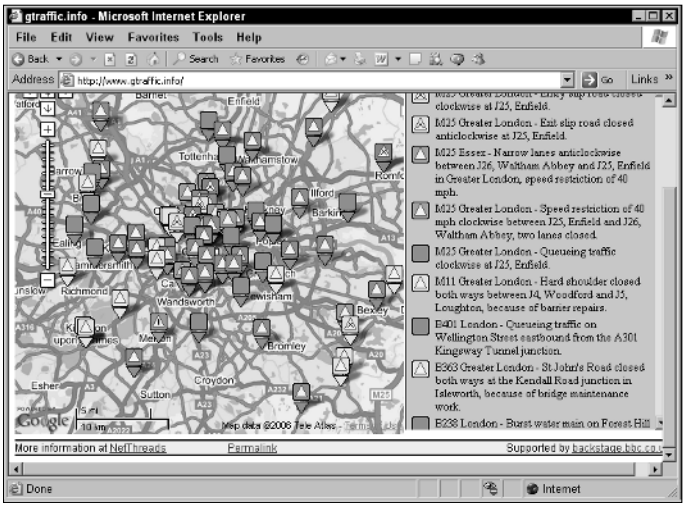
Using Google Maps




Many other map sites have sprung up since the original Google Maps introduction. For example, take a look at `www.gtraffic.info`, which gives you traffic information in the UK. Those arrows on the map you see in Figure 1.38 are Ajax-enabled.

**FIGURE 1.38**

Checking traffic conditions in the UK the Ajax way




## E-mail the Ajax way

Another popular use of Ajax is supporting browser-based e-mail, and you can see an example at `http://demo.nextapp.com/Email/app`, shown in Figure 1.39.

The problem with browser-based e-mail was that you'd see a list of e-mails sent to read, but each time you clicked an e-mail, you'd be taken to an entirely new page. Then you'd have to go back to the e-mail list and scroll down again to check other e-mails.

Handling e-mail the Ajax way is different. You can see the list of e-mails in Figure 1.39; clicking one automatically downloads the e-mail and displays it — without a browser refresh — as you see in Figure 1.40. The body of the e-mail appears at the bottom of the application.

In other words, once again, Ajax has been successful at turning a desktop application into an online one, solving the problems normally faced by online applications.

**FIGURE 1.39**

Browser-supported e-mail

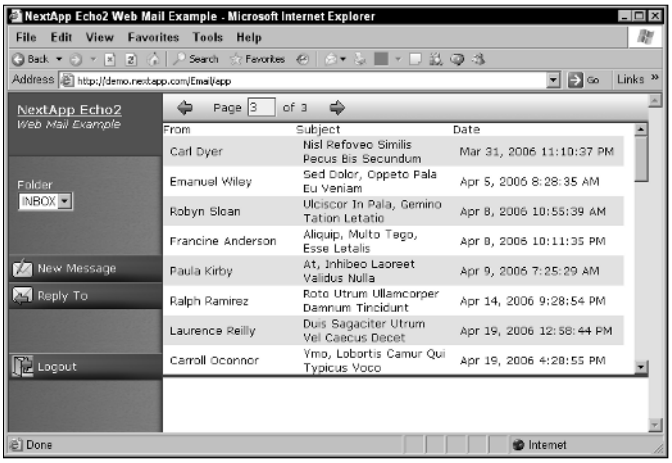
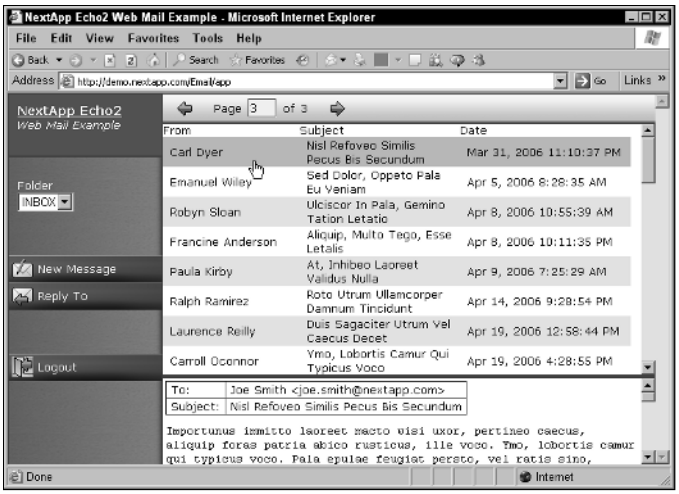



**FIGURE 1.40**

Using e-mail with Ajax



## Ajax-enabled pop-ups

Online map applications aren't the only ones using Ajax-enabled pop-ups; you can find them all over. For example, take a look at the Top 100 Netflix titles at `www.netflix.com/Top100`, shown in the following figure.


*Getting information on the Netflix top 100 movies*

All you've got to do to get more information on a movie is to let the mouse cursor rest on the name of the movie, and the application automatically downloads data from the server using Ajax and displays that data in a popup, as you see in the figure. That's Ajax at work, once again: no page refresh needed, and no need to click the Back button in your browser to get back to the original display.

# Summary

This chapter gave you an introduction to and overview of how Ajax is used today. As you can see, the emphasis is on making online applications feel like desktop applications, which is what Ajax excels at. A great deal of making that work has to do with using JavaScript in the browser, and that's what the next chapter is about: getting up to speed in JavaScript, the very foundation of Ajax.