Chapter 1

Ajax 101

In This Chapter

- Introducing how Ajax works
- Seeing Ajax at work in live searches, chat, shopping carts, and more

"We aren't getting enough orders on our Web site," storms the CEO. "People just don't like clicking all those buttons and waiting for a new page all the time. It's too distracting."

"How about a simpler solution?" you ask. "What if people could stay on the same page and just drag the items they want to buy to a shopping cart? No page refreshes, no fuss, no muss."

"You mean people wouldn't have to navigate from page to page to add items to a shopping cart and then check out? Customers could do everything on a single Web page?"

"Yep," you say. "And that page would automatically let our software on the server know what items the customer had purchased — all without having to reload the Web page."

"I love it!" the CEO says. "What's it called?"

"Ajax," you say.

Welcome to the world of Ajax, the technology that lets Web software act like desktop software. One of the biggest problems with traditional Web applications is that they have that "Web" feel — you have to keep clicking buttons to move from page to page, and watch the screen flicker as your browser loads a new Web page.

Ajax is here to take care of that issue, because it enables you grab data from the server without reloading new pages into the browser.

How Does Ajax Work?

With Ajax, Web applications finally start feeling like desktop applications to your users. That's because Ajax enables your Web applications to work behind the scenes, getting data as they need it, and displaying that data as you want. And as more and more people get fast Internet connections, working behind the scenes to access data is going to become all the rage. Soon, it'll be impossible to distinguish dedicated desktop software from software that's actually on the Internet, far from the user's machine. To help you understand how Ajax works, the following sections look at Ajax from a user's and a programmer's perspective.

A user's perspective

To show you how Ajax makes Web applications more like desktop applications, I'll use a simple Web search as an example. When you open a typical search engine, you see a text box where you type a search term. So say you type Ajax XML because you're trying to figure out what XML has to do with Ajax. Then, you click a Search the Web button to start the search. After that, the browser flickers, and a new page is loaded with your search results.

That's okay as far as it goes — but now take a look at an Ajax-enabled version of Yahoo! search. To see for yourself, go to `http://openrico.org/rico/yahooSearch.page`. When you enter your search term(s) and click Search Yahoo!, the page doesn't refresh; instead, the search results just appear in the box, as shown in Figure 1-1.

Figure 1-1: An Ajax-enabled Yahoo! search.

That's the Ajax difference. In the first case, you got a new page with search results, but to see more than ten results, a user has to keep loading pages. In the second case, everything happens on the same page. No page reloads, no fuss, no muss.

You can find plenty of Ajax on the `http://openrico.org` Web site. If you're inclined to, browse around and discover all the good stuff there.

A developer's perspective

In the article "Ajax: A New Approach to Web Applications" (`www.adaptivepath.com/publications/essays/archives/000385.php`), Jesse James Garrett, who was the first to call this technology Ajax, made important insights about how it could change the Web. He noted that although innovative new projects are typically online, Web programmers still feel that the rich capabilities of desktop software were out of their reach. But Ajax is closing the gap.

So how does Ajax do its stuff? The name *Ajax* is short for Asynchronous JavaScript and XML, and it's made up of several components:

- Browser-based presentation using HTML and Cascading Style Sheets (CSS)
- Data stored in XML format and fetched from the server
- Behind-the-scenes data fetches using `XMLHttpRequest` objects in the browser
- JavaScript to make everything happen

JavaScript is the scripting language that nearly all browsers support, which will let you fetch data behind the scenes, and XML is the popular language that lets you store data in an easy format. Here's an overview of how Ajax works:

1. In the browser, you write code in JavaScript that can fetch data from the server as needed.
2. When more data is needed from the server, the JavaScript uses a special item supported by browsers, the `XMLHttpRequest` object, to send a request to the server behind the scenes — without causing a page refresh.

   The JavaScript in the browser doesn't have to stop everything to wait for that data to come back from the server. It can wait for the data in the background and spring into action when the data does appear (that's called *asynchronous* data retrieval).

3. The data that comes back from the server can be XML (more on XML in Chapters 2 and 8), or just plain text if you prefer. The JavaScript code in the browser can read that data and put it to work immediately.

That's how Ajax works — it uses JavaScript in the browser and the `XMLHttpRequest` object to communicate with the server without page refreshes, and handles the XML (or other text) data sent back from the server. In Chapter 3, I explain how all these components work together in more detail.

This also points out what you'll need to develop Web pages with Ajax. You'll add JavaScript code to your Web page to fetch data from the server (I cover JavaScript in Chapter 2), and you'll need to store data and possibly write server-side code to interact with the browser behind the scenes. In other words, you're going to need access to an online server where you can store the data that you will fetch using Ajax. Besides just storing data on the server, you might want to put code on the server that your JavaScript can interact with. For example, a popular server-side language is PHP, and many of the examples in this book show how you can connect to PHP scripts on Web servers by using Ajax. (Chapter 10 is a PHP primer, getting you up to speed on that language if you're interested.) So you're going to need a Web server to store your data on, and if you want to run server-side programs as well, your server has to support server-side coding for the language you want to work with (such as PHP).

What Can You Do with Ajax?

The technology for Ajax has been around since 1998, and a handful of applications (such as Microsoft's Outlook Web Access) have already put it to use. But Ajax didn't really catch on until early 2005, when a couple of high-profile Web applications (such as Google Suggest and Google Maps, both reviewed later in this chapter) put it to work, and Jesse James Garrett wrote his article coining the term Ajax and so putting everything under one roof.

Since then, Ajax has exploded as people have realized that Web software can finally start acting like desktop software. What can you do with Ajax? That's what the rest of this chapter is about.

Searching in real time with live searches

One of the truly cool things you can do with Ajax is live searching, where you get search results instantly, as you enter the term you're searching for. For example, take a look at `http://www.google.com/webhp?complete=1&hl=en`, the page which appears in Figure 1-2. As you enter a term to search

for, Ajax contacts Google behind the scenes, and you see a drop-down menu that displays common search terms from Google that might match what you're typing. If you want to select one of those terms, just click it in the menu. That's all there is to it.

You can also write an Ajax application that connects to Google in this way behind the scenes. Chapter 4 has all the details.

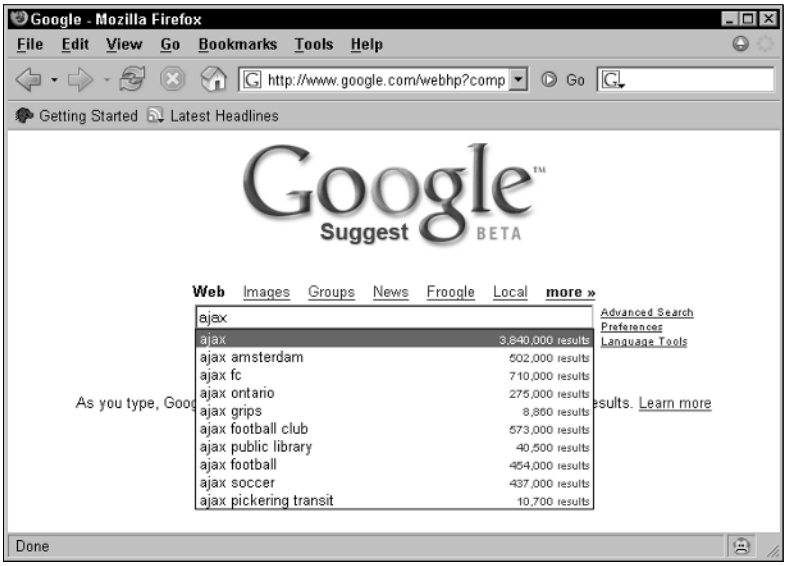

Figure 1-2: A Google live search.

Getting the answer with autocomplete

Closely allied to live search applications are autocomplete applications, which try to guess the word you're entering by getting a list of similar words from the server and displaying them. You can see an example at `www.paper mountain.org/demos/live`, which appears in Figure 1-3.

As you enter a word, this example looks up words that might match in a dictionary on the server and displays them, as you see in Figure 1-3. If you see the right one, just click it to enter it in the text field, saving you some typing.

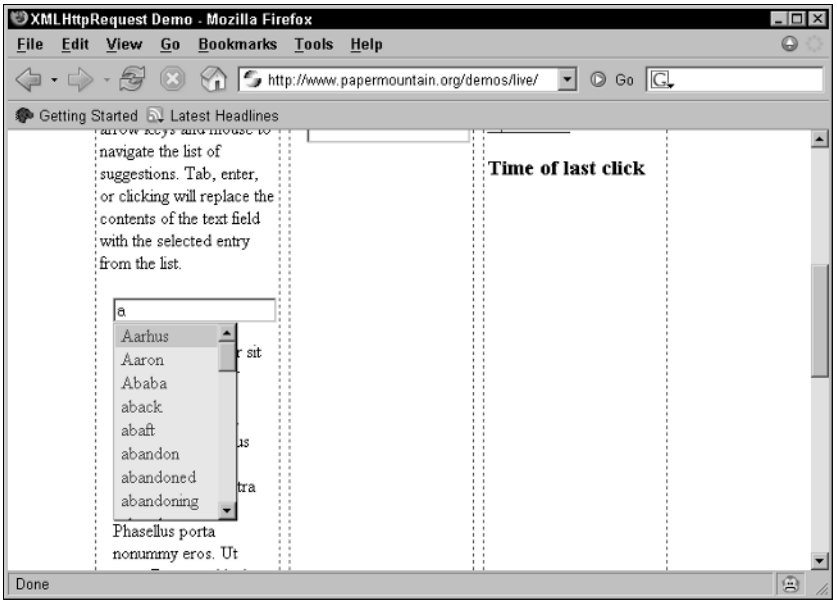

Figure 1-3: An autocomplete example.

Chatting with friends

Because Ajax excels at updating Web pages without refreshing the displayed page, it's a great choice for Web-based chat programs, where many users can chat together at the same time. Take a look at `www.plasticshore.com/projects/chat`, for example, which you can see in Figure 1-4. Here, you just enter your text and click the Submit button to send that text to the server. All the while, you can see everyone else currently chatting — no page refresh needed.

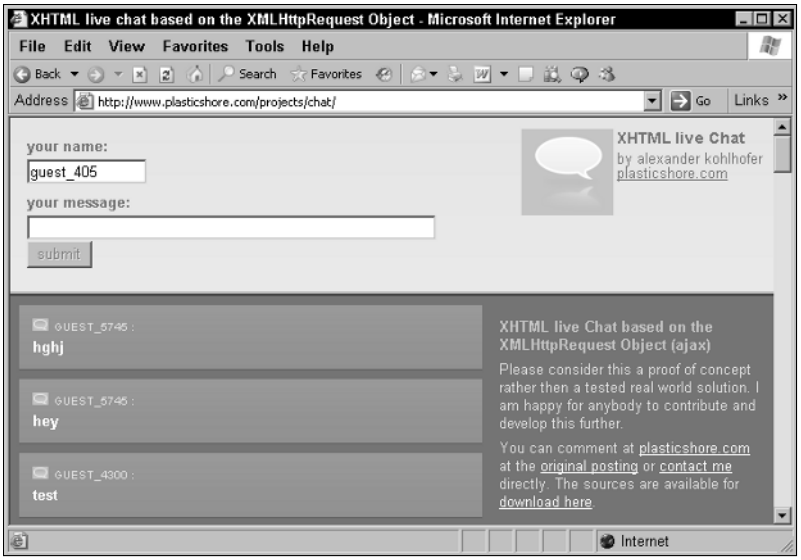

Figure 1-4: An Ajax-based chat application.

There are plenty of Ajax-based chat rooms around. Take a look at `http://treehouse.ofb.net/chat/?lang=en` for another example.

Dragging and dropping with Ajax

At the beginning of this chapter, I mention a drag-and-drop shopping cart example. As shown in Figure 1-5, when the user drags the television to the shopping cart in the lower-right, the server is notified that the user bought a television. Then the server sends back the text that appears in the upper left, "You just bought a nice television." You find out how to create this shopping cart in Chapter 6.

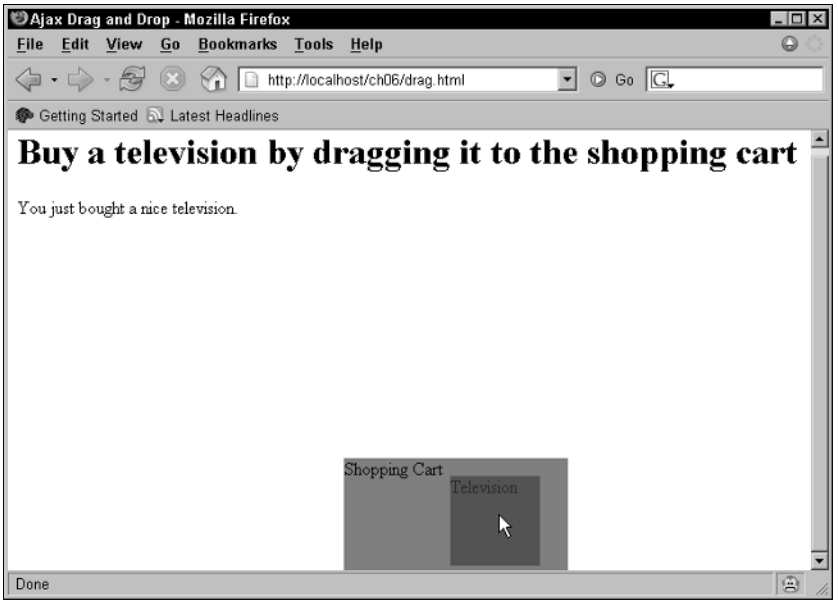

Figure 1-5: Drag-and-drop shopping.

Gaming with Ajax

Here's a cute one — a magic diary that answers you back using Ajax techniques, as shown in Figure 1-6. You can find it at `http://pandorabots.com/pandora/talk?botid=c96f911b3e35f9e1`. When you type something, such as "Hello," the server is notified and sends back an appropriate response that then appears in the diary, such as "Hi there!"

Or how about a game of chess, via Ajax? Take a look at `www.jesperolsen.net/PChess`, where you can move the pieces around (and the software on the server can, too) thanks to Ajax.

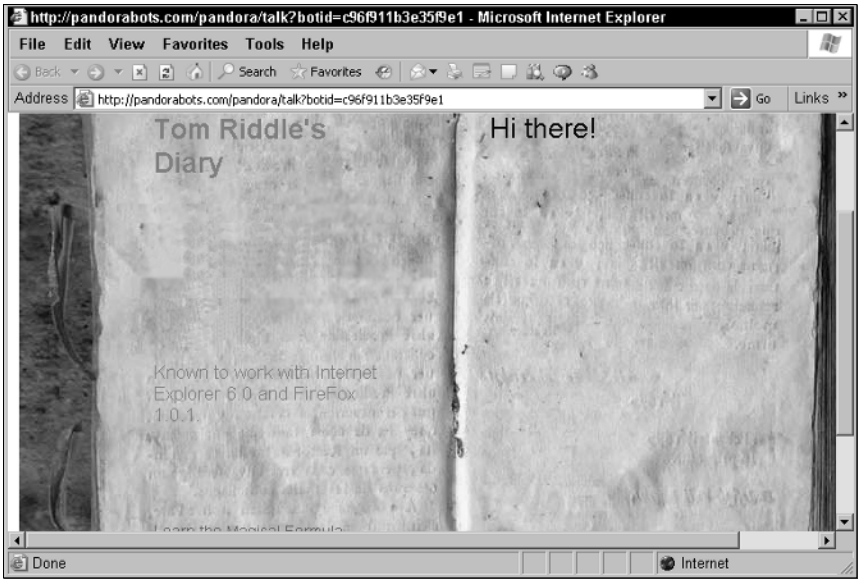

Figure 1-6: An interactive Ajax-enabled diary.

Getting instant login feedback

Another Internet task that can involve many annoying page refreshes is logging in to a site. If you type the wrong login name, for example, you get a new page explaining the problem, have to log in on another page, and so on. How about getting instant feedback on your login attempt, courtesy of Ajax? That's possible, too. Take a look at `www.jamesdam.com/ajax_login/login.html`, which appears in Figure 1-7. I've entered an incorrect username and password, and the application says so immediately. You'll see how to write a login application like this in Chapter 4.

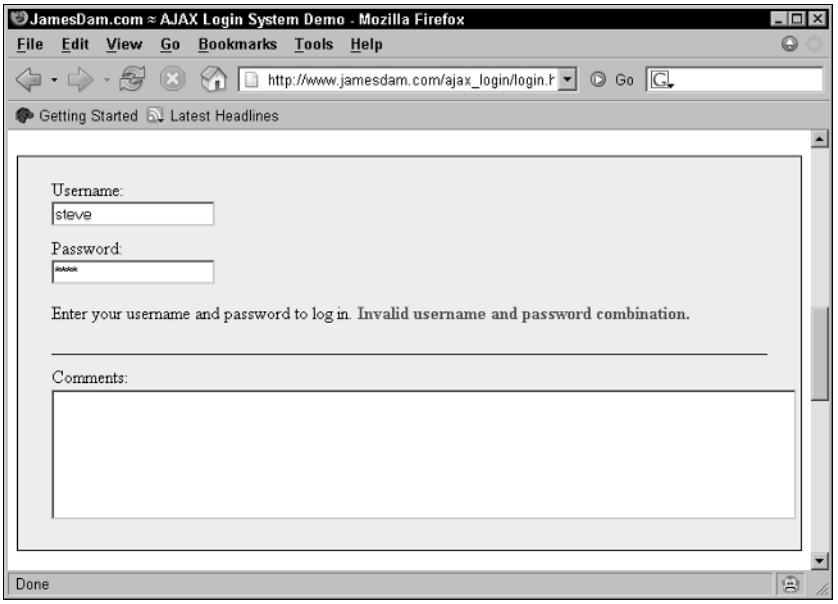

Figure 1-7: Ajax makes correcting login mistakes easier.

Ajax-enabled pop-up menus

You can grab data from the server as soon as the user needs it using Ajax. For example, take a look at the application in Figure 1-8, which I explain how to build in Chapter 9. The pop-up menus appear when you move the mouse and display text retrieved from the server using Ajax techniques. By accessing the server, Ajax allows you to set up an interactive menu system that responds to the menu choices the user has already made.

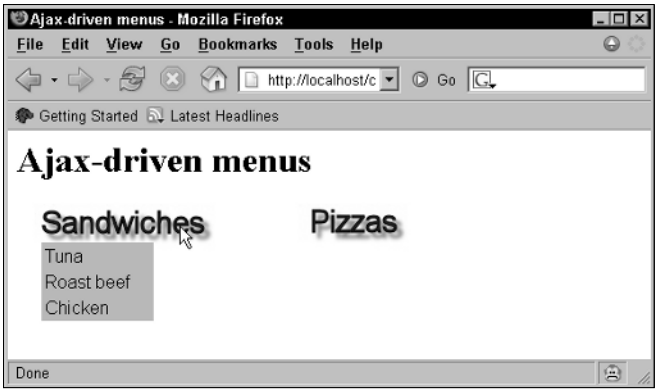

Figure 1-8: Ajax-enabled pop-up menus.

Modifying Web pages on the fly

Ajax excels at updating Web pages on the fly without page refreshes, and you can find hundreds of Ajax applications doing exactly that. For example, take a look at the Ajax rolodex at `http://openrico.org/rico/demos.page?demo=ricoAjaxInnerHTML.html`, shown in Figure 1-9. When you click someone's name, a "card" appears with their full data.

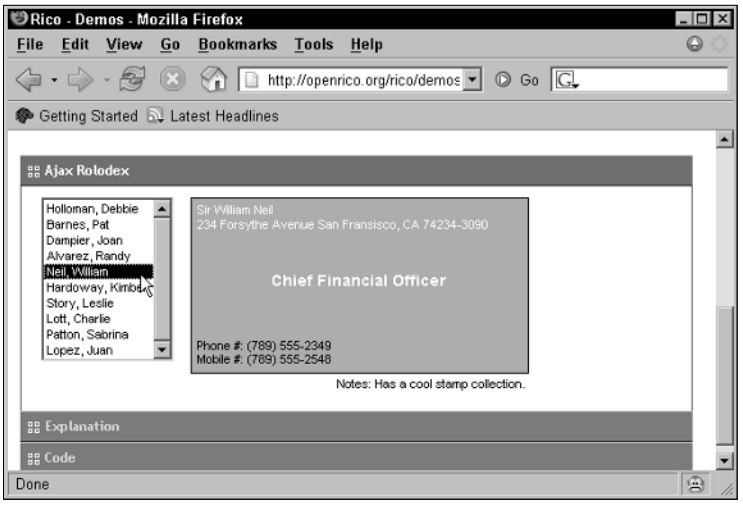

Figure 1-9: An Ajax rolodex.

You can see another example at `http://digg.com/spy`. This news Web site uses Ajax techniques to update itself periodically by adding new article titles to the list on the page.

Updating the HTML in a Web page by fetching data is a very popular Ajax technique, and you see a lot of it in Chapters 3 and 4.

Google Maps and Ajax

One of the most famous Ajax application is Google Maps, at `http://maps.google.com`, which you can see at work in Figure 1-10, zooming in on South Market Street in Boston.

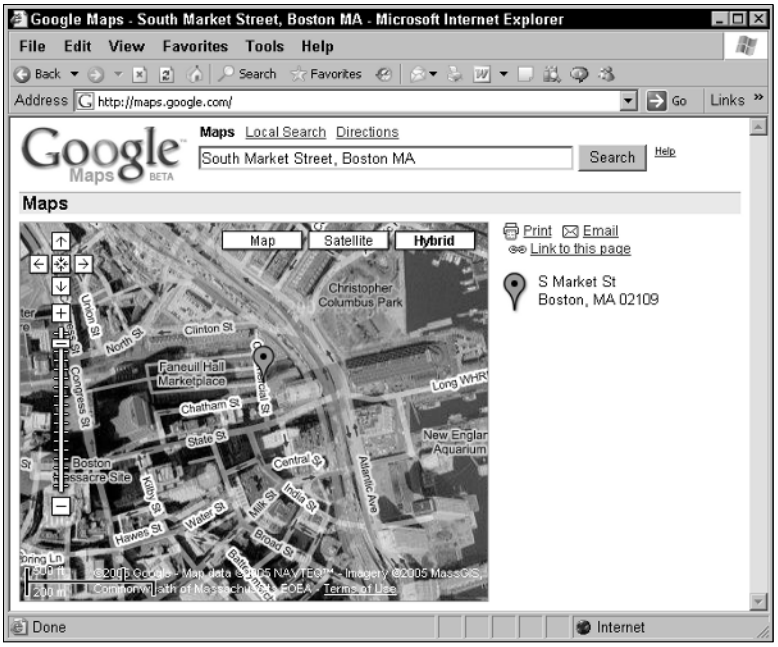

Figure 1-10: Using Google maps.

See that marker icon near the center of the map? The location for that marker is passed to the browser from the server using Ajax techniques, and the Ajax code in the browser positions the marker accordingly. Ajax at work again!

When Is Ajax a Good Choice?

The examples I show in the preceding section are just the beginning — dozens more, including those you can write yourself, appear in later chapters. Got a Web application that asks the user to move from page to page and therefore needs to be improved? That's a job for Ajax.