What you should already have to get the most out of this book...

- A familiarity with the basic workings of your computer, your operating system, and the Internet
- A good understanding of HTML and web page development
- Basic web server knowledge and administration may be helpful

Some books that may help with the basics...

- Sams Teach Yourself Windows Vista All in One
  This book will teach you the basics of using your computer—and Windows Vista—effectively

- Sams Teach Yourself Mac OS X Leopard All in One
  Covers all the most important topics for the reader who wants to get up and running as quickly as possible

What this book will help you learn...

- How to make JavaScript, HTML, XML, and PHP work together to create Ajax effects
- How to build better, more interactive interfaces for your web apps

Related titles...

- Sams Teach Yourself HTML and CSS in 24 Hours
  Explains the basics for creating a web page with HTML and putting it online

- Sams Teach Yourself JavaScript in 24 Hours
  A good introduction to creating interactive web sites with JavaScript

Where you may want to go from here...

- Enhance your Ajax skills to build more complex web-based applications
- Expand your web development abilities by learning other languages and development technologies

Possible titles to look for...

- Sams Teach Yourself PHP, MySQL and Apache All in One
  A great introduction to using PHP with MySQL and Apache

- Sams Teach Yourself Django in 24 Hours
  A detailed tutorial on creating websites with the Django framework
Contents at a Glance

Introduction ........................................................................................................... 1

Part I: Web Basics Refresher ............................................................................. 7
  CHAPTER 1  Workings of the Web ................................................................. 9
  2  Writing and Styling Pages in HTML and CSS ..................................... 21
  3  Anatomy of an Ajax Application ............................................................ 41

Part II: Introducing Web Scripting with JavaScript .......................................... 51
  CHAPTER 4  Creating Simple Scripts in JavaScript .................................. 53
  5  Working with the Document Object Model (DOM) ........................... 71
  6  Using Variables, Strings, and Arrays ..................................................... 81
  7  Using Functions and Objects ................................................................. 103
  8  Controlling Flow with Conditions and Loops .................................... 117
  9  Using Built-In Functions and Libraries ............................................... 135

Part III: Introducing Ajax ................................................................................. 147
  CHAPTER 10 The Heart of Ajax—the XMLHttpRequest Object ............. 149
  11 Talking with the Server .......................................................................... 157
  12 Using the Returned Data ........................................................................ 167
  13 Our First Ajax Application ..................................................................... 175

Part IV: Server-side Scripting with PHP ........................................................... 185
  CHAPTER 14 Getting to Know PHP ............................................................. 187
  15 Variables .................................................................................................. 195
  16 Flow Control ............................................................................................ 215
  17 Functions .................................................................................................. 223
  18 Using Classes ........................................................................................... 231
## Part V: More Complex Ajax Technologies

**CHAPTER 19** Returning Data as Text .............................................................. 239  
20 AHAH—Asynchronous HTML and HTTP .................................................. 247  
21 Returning Data as XML ............................................................................. 257  
22 Web Services and the REST and SOAP Protocols ..................................... 271  
23 A JavaScript Library for Ajax ..................................................................... 285  
24 Ajax Gotchas .............................................................................................. 295

## Part VI: Ajax Tools and Resources

**CHAPTER 25** The prototype.js Toolkit ............................................................ 305  
26 Using Rico ................................................................................................. 315  
27 Using Script.aculo.us .................................................................................. 325  
28 Using XOAD ............................................................................................... 331  

**APPENDIX** JavaScript, PHP, and Ajax Websites .......................................... 339  
**GLOSSARY** .................................................................................................. 343  

Index ............................................................................................................... 349
# Table of Contents

Introduction

## Part I: Web Basics Refresher

### CHAPTER 1: Workings of the Web

- A Short History of the Internet ................................................................. 9
- The World Wide Web .............................................................................. 10
- Introducing HTTP .................................................................................. 14
- The HTTP Request and Response ......................................................... 15
- HTML Forms .......................................................................................... 17
- Summary ............................................................................................... 20

### CHAPTER 2: Writing and Styling Pages in HTML and CSS

- Introducing HTML .................................................................................. 21
- Elements of an HTML Page ..................................................................... 23
- A More Advanced HTML Page .............................................................. 27
- Some Useful HTML Tags ........................................................................ 29
- Adding Your Own Style ......................................................................... 30
- Defining the Rules ................................................................................ 31
- Add a Little class ................................................................................... 31
- Applying Styles ..................................................................................... 33
- Formatting Text with Styles ................................................................. 35
- Adding Lines ........................................................................................ 39
- Summary ............................................................................................... 40

### CHAPTER 3: Anatomy of an Ajax Application

- The Need for Ajax .................................................................................. 41
- Introducing Ajax .................................................................................... 43
- The Constituent Parts of Ajax ............................................................... 45
- Putting It All Together ......................................................................... 47
- Summary ............................................................................................... 49
## CHAPTER 7: Using Functions and Objects 103

- Using Functions ................................. 103
- Introducing Objects ......................... 108
- Using Objects to Simplify Scripting .......... 109
- Extending Built-in Objects .................. 112
- Summary ........................................ 115

## CHAPTER 8: Controlling Flow with Conditions and Loops 117

- The if Statement ............................. 118
- Using Shorthand Conditional Expressions .... 121
- Testing Multiple Conditions with if and else ... 122
- Using Multiple Conditions with switch .......... 124
- Using for Loops .............................. 125
- Using while Loops ............................ 128
- Using do...while Loops ..................... 128
- Working with Loops ......................... 129
- Looping Through Object Properties ........... 131
- Summary ........................................ 134

## CHAPTER 9: Using Built-In Functions and Libraries 135

- Using the Math Object ..................... 135
- Working with Math Functions ............... 137
- Using the with Keyword .................... 139
- Working with Dates ......................... 140
- Using Third-Party Libraries ............... 143
- Other Libraries .............................. 144
- Summary ........................................ 145

## Part III: Introducing Ajax 147

## CHAPTER 10: The Heart of Ajax—the XMLHttpRequest Object 149

- Introducing XMLHttpRequest ............... 149
- Creating the XMLHttpRequest Object ........ 150
- Summary ........................................ 156
**CHAPTER 11: Talking with the Server** 157
- Sending the Server Request .................................................. 157
- Monitoring Server Status ...................................................... 162
- The Callback Function ......................................................... 163
- Summary ........................................................................... 166

**CHAPTER 12: Using the Returned Data** 167
- The responseText and responseXML Properties ..................... 167
- Parsing responseXML .......................................................... 171
- Providing User Feedback ...................................................... 172
- Summary ........................................................................... 174

**CHAPTER 13: Our First Ajax Application** 175
- Constructing the Ajax Application ........................................... 175
- The HTML Document .......................................................... 176
- Adding JavaScript .............................................................. 177
- Putting It All Together .......................................................... 180
- Summary ........................................................................... 184

**Part IV: Server-side Scripting with PHP** 185

**CHAPTER 14: Getting to Know PHP** 187
- PHP Basics ......................................................................... 187
- Your First PHP Script ........................................................... 190
- Summary ........................................................................... 194

**CHAPTER 15: Variables** 195
- Understanding Variables ....................................................... 195
- Data Types ........................................................................ 197
- Working with Numbers ........................................................ 198
- Numeric Data Types ............................................................. 200
- Numeric Functions ............................................................... 200
- Working with Strings ............................................................ 202
- Formatting Strings ............................................................... 203
# Sams Teach Yourself Ajax, JavaScript, and PHP All in One

## CHAPTER 21: Returning Data as XML 257

Adding the “x” to Ajax ................................................................. 257
The responseXML Property .......................................................... 258
Project—An RSS Headline Reader ........................................... 262
Summary .................................................................................... 269

## CHAPTER 22: Web Services and the REST and SOAP Protocols 271

Introduction to Web Services .................................................... 271
REST—Representational State Transfer .................................... 272
Using REST in Practice .............................................................. 274
Web Services Using SOAP .......................................................... 278
The SOAP Protocol .................................................................... 279
Using Ajax and SOAP ............................................................... 282
Reviewing SOAP and REST ....................................................... 283
Summary .................................................................................... 284

## CHAPTER 23: A JavaScript Library for Ajax 285

An Ajax Library ........................................................................... 285
Reviewing myAHAHlib.js ............................................................ 286
Implementing Our Library .......................................................... 287
Using the Library ...................................................................... 290
Summary .................................................................................... 293

## CHAPTER 24: Ajax Gotchas 295

Common Ajax Errors ................................................................... 295
The Back Button ........................................................................ 296
Bookmarking and Links .............................................................. 297
Telling the User That Something Is Happening ......................... 297
Making Ajax Degrade Elegantly ................................................ 297
Dealing with Search Engine Spiders ........................................ 298
Pointing Out Active Page Elements ......................................... 299
Don’t Use Ajax Where It’s Inappropriate .................................. 299
Security .......................................................................................... 300
Test Code Across Multiple Platforms ...................................................... 300
Ajax Won’t Cure a Bad Design ............................................................ 301
Some Programming Gotchas ............................................................... 301
Summary .......................................................................................... 302

Part VI: Ajax Tools and Resources 303

CHAPTER 25: The prototype.js Toolkit 305
Introducing prototype.js ........................................................................ 305
Wrapping XMLHttpRequest—the Ajax Object ........................................ 308
Example Project—Stock Price Reader .................................................... 311
Summary .......................................................................................... 313

CHAPTER 26: Using Rico 315
Introducing Rico ................................................................................ 315
Rico’s Other Interface Tools .................................................................. 320
Summary .......................................................................................... 324

CHAPTER 27: Using Script.aculo.us 325
Downloading the Library ........................................................................ 325
Including the Files .............................................................................. 326
Using Effects ...................................................................................... 326
Building the Script .............................................................................. 327
Summary .......................................................................................... 329

CHAPTER 28: Using XOAD 331
Introducing XOAD .............................................................................. 331
XOAD HTML .................................................................................... 334
Advanced Programming with XOAD .................................................... 337
Summary .......................................................................................... 338
About the Author

Phil Ballard, the author of *Sams Teach Yourself Ajax in 10 Minutes*, graduated in 1980 with an honors degree in electronics from the University of Leeds, England. Following an early career as a research scientist with a major multinational, he spent a few years in commercial and managerial roles within the high technology sector, later working full time as a software engineering consultant.

Operating as “The Mouse Whisperer” (http://www.mousewhisperer.co.uk), Ballard has spent recent years involved solely in website and intranet design and development for an international portfolio of clients.

Michael Moncur is a freelance webmaster and author. He runs a network of websites, including the Web’s oldest site about famous quotations, online since 1994. He wrote *Sams Teach Yourself JavaScript in 24 Hours* and has also written several bestselling books about networking, certification programs, and databases. He lives with his wife in Salt Lake City, Utah.
We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we’re doing right, what we could do better, what areas you’d like to see us publish in, and any other words of wisdom you’re willing to pass our way.

You can email or write me directly to let me know what you did or didn’t like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book’s title and author as well as your name and phone or email address. I will carefully review your comments and share them with the author and editors who worked on the book.

E-mail: webdev@samspublishing.com

Mail: Mark Taub
Associate Publisher
Sams Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.
Introduction

Over the last decade or so, the World Wide Web has grown in scope from being a relatively simple information repository to becoming the first stop for many people when seeking entertainment, education, news, or business resources.

Websites themselves need no longer be limited to a number of static pages containing text and perhaps simple images; the tools now available allow the development of highly interactive and engaging pages involving animations, visual effects, context-sensitive content, embedded productivity tools, and much more.

The list of technologies available for producing such pages is broad. However, those based on Open Source licenses have become, and remain, highly popular due to their typically low (often zero) entry cost, and to the huge resource of user-contributed scripts, tutorials, tools, and other resources for these tools and applications available via the Internet and elsewhere.

In this book, we give a detailed account of how to program fluid, interactive websites using server- and client-side coding techniques and tools, as well as how to combine these to produce a slick, desktop-application-like user experience using Ajax.

The programming languages used in this book include the ubiquitous JavaScript (for client-side programming) and the immensely popular open-source PHP language (for server-side scripting, and available with the majority of web-hosting packages). The nuts and bolts of Ajax programming are described in detail, as well as the use of several advanced open-source frameworks that contain ready-written code for quickly building state-of-the-art interactive sites.

The CD that accompanies this book provides all the tools required on your journey through learning to program in PHP, JavaScript, and Ajax.

What Is Ajax?

Ajax stands for Asynchronous JavaScript And XML. Although strictly speaking Ajax is not itself a technology, it mixes well-known programming techniques in an uncommon way to enable web developers to build Internet applications with much more appealing user interfaces than those to which we have become accustomed.
When using popular desktop applications, we expect the results of our work to be made available immediately, without fuss, and without our having to wait for the whole screen to be redrawn by the program. While using a spreadsheet such as Excel, for instance, we expect the changes we make in one cell to propagate immediately through the neighboring cells while we continue to type, scroll the page, or use the mouse.

Unfortunately, this sort of interaction has seldom been available to users of web-based applications. Much more common is the experience of entering data into form fields, clicking on a button or a hyperlink and then sitting back while the page slowly reloads to exhibit the results of the request. In addition, we often find that the majority of the reloaded page consists of elements that are identical to those of the previous page and that have therefore been reloaded unnecessarily; background images, logos, and menus are frequent offenders.

Ajax promises us a solution to this problem. By working as an extra layer between the user’s browser and the web server, Ajax handles server communications in the background, submitting server requests and processing the returned data. The results may then be integrated seamlessly into the page being viewed, without that page needing to be refreshed or a new one being loaded.

In Ajax applications, such server requests are not necessarily synchronized with user actions such as clicking on buttons or links. A well-written Ajax application may already have asked of the server, and received, the data required by the user—perhaps before the user even knew she wanted it. This is the meaning of the asynchronous part of the Ajax acronym.

The parts of an Ajax application that happen “under the hood” of the user’s browser, such as sending server queries and dealing with the returned data, are written in JavaScript, and XML is an increasingly popular means of coding and transferring formatted information used by Ajax to efficiently transfer data between server and client.

We’ll look at all these techniques, and how they can be made to work together, as we work through the chapters.

**Who This Book Is For**

This volume is aimed primarily at web developers seeking to build better interfaces for the users of their web applications and programmers from desktop environments looking to transfer their applications to the Internet.
It also proves useful to web designers eager to learn how the latest techniques can offer new outlets for their creativity. Although the nature of PHP, JavaScript, and Ajax applications means that they require some programming, all the required technologies are explained from first principles within the book, so even those with little or no programming experience should be able to follow the lessons without a great deal of difficulty.

**How To Use This Book**

All the technologies—including a refresher of WWW basics—are explained from first principles, so that even non-programmers or those unfamiliar with these languages should be able to follow the development of the concepts with little problem.

The book is divided into parts, each dedicated to a particular technology or discussion topic. Within each part, the chapters each specialize in a given aspect or subtopic. It should therefore be easy to follow the instructional flow of the book by a quick look through the table of contents.

However, if you are already a competent programmer in one or more of the technologies used—in PHP for instance, or in JavaScript—then feel free to speed-read or skip the sections that you don’t need.

To try out many of the examples you’ll need access to a web server that supports PHP, and a means to upload files into your web space (probably FTP). Most web hosts include PHP in their hosting packages, or can do so on request at minimal or no cost.

Alternatively, the CD that accompanies this book contains everything required to set up a web serving environment on your own computer. This package is called XAMPP, and it contains everything you need to develop fully functional, interactive websites like those described in this book, ready to be deployed to a web-based server at a later date if you so choose. Look out for the boxes marked “On the CD” as you work through the book.

**Conventions Used In This Book**

This book contains special elements as described by the following:

These boxes highlight information that can make your programming more efficient and effective.
Setting Up Your Workspace

While you can write the code in this book using just a simple text editor, to run the examples you’ll need a computer (with Windows, Mac, or Linux operating system) running a modern browser such as Internet Explorer or Firefox.


You will also need to load files on to a web server—if you already have a web host that supports PHP, you can use your web space there. Alternatively, the accompanying CD has everything you need to set up your own web server for private use, either on your own PC or another on your network.
What’s on the CD

The accompanying CD contains everything you could need to get the best from this book. Included on the CD you’ll find

- **XAMPP**, a complete open source compilation you can use to easily install the Apache web server, PHP language, and MySQL database manager on your computer. Versions are provided for Linux, Mac, and Windows environments.

- **jEdit**, a Java-based programmer’s editor that’s perfect for creating or modifying code. The CD includes files for Java, Mac, or Windows.

- A selection of **open source frameworks** for developing sophisticated web applications. Programming examples based on some of these frameworks are presented towards the end of the book.
CHAPTER 3

Anatomy of an Ajax Application

What You’ll Learn in This Chapter:

- The Need for Ajax
- Introducing Ajax
- The Constituent Parts of Ajax
- Putting It All Together

In this chapter you will learn about the individual building blocks of Ajax and how they fit together to form the architecture of an Ajax application. Subsequent chapters will examine these components in more detail, finally assembling them into a working Ajax application.

The Need for Ajax

In the following parts of the book, we shall discuss each of the core components in detail. Before discussing the individual components, though, let’s look in more detail at what we want from our Ajax application.

Traditional Versus Ajax Client-Server Interactions

Chapter 1 discussed the traditional page-based model of a website user interface. When you interact with such a website, individual pages containing text, images, data entry forms, and so forth are presented one at a time. Each page must be dealt with individually before navigating to the next.

For instance, you may complete the data entry fields of a form, editing and re-editing your entries as much as you want, knowing that the data will not be sent to the server until the form is finally submitted.

Figure 3.1 illustrates this interaction.
After you submit a form or follow a navigation link, you then must wait while the browser screen refreshes to display the new or revised page that has been delivered by the server.

As your experience as an Internet user grows, using this interface becomes almost second nature. You learn certain rules of thumb that help to keep you out of trouble, such as “don’t click the Submit button a second time,” and “don’t click the Back button after submitting a form.”

Unfortunately, interfaces built using this model have a few drawbacks. First, there is a significant delay while each new or revised page is loaded. This interrupts what we, as users, perceive as the “flow” of the application.

Furthermore, a whole page must be loaded on each occasion, even when most of its content is identical to that of the previous page. Items common to many pages on a website, such as header, footer, and navigation sections, can amount to a significant proportion of the data contained in the page.

Figure 3.2 illustrates a website displaying pages before and after the submission of a form, showing how much identical content has been reloaded and how relatively little of the display has actually changed.

This unnecessary download of data wastes bandwidth and further exacerbates the delay in loading each new page.

**By the Way**

Bandwidth refers to the capacity of a communications channel to carry information. On the Internet, bandwidth is usually measured in bps (bits per second) or in higher multiples such as Mbps (million bits per second).
The Rich User Experience

The combined effect of the issues just described is to offer a much inferior user experience compared to that provided by the vast majority of desktop applications.

On the desktop, you expect the display contents of a program to remain visible and the interface elements to respond to commands while the computing processes occur quietly in the background. As I write this chapter using a word processor, for example, I can save the document to disk, scroll or page up and down, and alter font faces and sizes without having to wait on each occasion for the entire display to be refreshed.

Ajax allows you to add to your web application interfaces some of this functionality more commonly seen in desktop applications and often referred to as a rich user experience.

Introducing Ajax

To improve the user’s experience, you need to add some extra capabilities to the traditional page-based interface design. You want your user’s page to be interactive, responding to the user’s actions with revised content, and be updated without any interruptions for page loads or screen refreshes.

To achieve this, Ajax builds an extra layer of processing between the web page and the server.
CHAPTER 3: Anatomy of an Ajax Application

This layer, often referred to as an Ajax Engine or Ajax Framework, intercepts requests from the user and in the background handles server communications quietly, unobtrusively, and asynchronously. By this we mean that server requests and responses no longer need to coincide with particular user actions but may happen at any time convenient to the user and to the correct operation of the application. The browser does not freeze and await the completion by the server of the last request but instead lets the user carry on scrolling, clicking, and typing in the current page.

The updating of page elements to reflect the revised information received from the server is also looked after by Ajax, happening dynamically while the page continues to be used.

Figure 3.3 represents how these interactions take place.

A Real Ajax Application—Google Suggest

To see an example of an Ajax application in action, let’s have a look at Google Suggest. This application extends the familiar Google search engine interface to offer the user suggestions for suitable search terms, based on what he has so far typed.

With each key pressed by the user, the application’s Ajax layer queries Google’s server for suitably similar search phrases and presents the returned data in a drop-down box. Along with each suggested phrase is listed the number of results that would be expected for a search conducted using that phrase. At any point the user has the option to select one of these suggestions instead of continuing to type and have Google process the selected search.

Because the server is queried with every keypress, this drop-down list updates dynamically as the user types—with no waiting for page refreshes or similar interruptions.

Figure 3.4 shows the program in action. You can try it for yourself by following the links from Google’s home page at http://www.google.com/webhp?complete=1&hl=en.
Next let’s identify the individual components of such an Ajax application and see how they work together.

Google has presented other Ajax-enabled applications that you can try, including the *gmail* web mail service and the *Google Maps* street mapping program. See the Google website at http://www.google.com/ for details.

**The Constituent Parts of Ajax**

Now let’s examine the components of an Ajax application one at a time.

**The XMLHttpRequest Object**

When you click on a hyperlink or submit an HTML form, you send an HTTP request to the server, which responds by serving to you a new or revised page. For your web application to work asynchronously, however, you must have a means to send HTTP requests to the server without an associated request to display a new page.

You can do so by means of the XMLHttpRequest object. This JavaScript object is capable of making a connection to the server and issuing an HTTP request without the necessity of an associated page load.

In following chapters you will learn what objects are, see how an instance of this object can be created, and see how its properties and methods can be used by JavaScript routines included in the web page to establish asynchronous communications with the server.
As a security measure, the XMLHttpRequest object can generally only make calls to URLs within the same domain as the calling page and cannot directly call a remote server.

Chapter 5, “Working with the Document Object Model” will introduce the concept of objects in general, and this subject will be expanded in Chapter 7 “Using Functions and Objects.”

Chapter 10, “The Heart of Ajax”—the XMLHttpRequest Object, discusses how to create an instance of the XMLHttpRequest object and reviews the object’s properties and methods.

**Talking with the Server**

In the traditional style of web page, when you issue a server request via a hyperlink or a form submission, the server accepts that request, carries out any required server-side processing, and subsequently serves to you a new page with content appropriate to the action you have undertaken.

While this processing takes place, the user interface is effectively frozen. You are made quite aware of this, when the server has completed its task, by the appearance in the browser of the new or revised page.

With asynchronous server requests, however, such communications occur in the background, and the completion of such a request does not necessarily coincide with a screen refresh or a new page being loaded. You must therefore make other arrangements to find out what progress the server has made in dealing with the request.

The XMLHttpRequest object possesses a convenient property to report on the progress of the server request. You can examine this property using JavaScript routines to determine the point at which the server has completed its task and the results are available for use.

Your Ajax armory must therefore include a routine to monitor the status of a request and to act accordingly. We’ll look at this in more detail in Chapter 11, “Talking with the Server.”

**What Happens at the Server?**

So far as the server-side script is concerned, the communication from the XMLHttpRequest object is just another HTTP request. Ajax applications care little about what languages or operating environments exist at the server; provided that the client-side Ajax layer receives a timely and correctly formatted HTTP response from the server, everything will work just fine.
It is possible to build simple Ajax applications with no server-side scripting at all, simply by having the XMLHttpRequest object call a static server resource such as an XML or text file.

Ajax applications may make calls to various other server-side resources such as web services. Later in the book we’ll look at some examples of calling web services using protocols such as SOAP and REST.

In this book we’ll be using the popular PHP scripting language for our server-side routines, but if you are more comfortable with ASP, JSP, or some other server-side language, go right ahead and use it in your Ajax applications.

**Dealing with the Server Response**

Once notified that an asynchronous request has been successfully completed, you may then utilize the information returned by the server.

Ajax allows for this information to be returned in a number of formats, including ASCII text and XML data.

Depending on the nature of the application, you may then translate, display, or otherwise process this information within the current page.

We’ll look into these issues in Chapter 12, “Using the Returned Data.”

**Other Housekeeping Tasks**

An Ajax application will be required to carry out a number of other duties, too. Examples include detecting error conditions and handling them appropriately, and keeping the user informed about the status of submitted Ajax requests.

You will see various examples in later chapters.

**Putting It All Together**

Suppose that you want to design a new Ajax application, or update a legacy web application to include Ajax techniques. How do you go about it?

First you need to decide what page events and user actions will be responsible for causing the sending of an asynchronous HTTP request. You may decide, for example, that the action of moving the mouse cursor over an image will result in a request being sent to the server to retrieve further information about the subject of the picture, or that the clicking of a button will generate a server request for information with which to populate the fields on a form.
JavaScript can be used to execute instructions on occurrences such as these, by employing event handlers. The details of how will be covered in detail in the following chapters. In your Ajax applications, such methods will be responsible for initiating asynchronous HTTP requests via XMLHttpRequest.

Having made the request, you need to write routines to monitor the progress of that request until you hear from the server that the request has been successfully completed.

Finally, after receiving notification that the server has completed its task, you need a routine to retrieve the information returned from the server and apply it in the application. You may, for example, want to use the newly returned data to change the contents of the page’s body text, populate the fields of a form, or pop open an information window.

Figure 3.5 shows the flow diagram of all this.

In Chapter 13, “Our First Ajax Application,” you’ll use what you have learned to construct a complete Ajax application.
**Ajax Frameworks**

While it is essential for a complete understanding of Ajax to understand what role each of the individual components plays, it is thankfully not necessary to rewrite all of your code for each new application. Your Ajax code can be stored as a reusable library of common Ajax routines, ready to be reused wherever they may be needed. There are also many commercial and open-source frameworks that you can use in your projects to do the “heavy lifting.”

We shall look at both of these techniques later in the book, where we develop our own JavaScript library for Ajax, and also consider several of the more popular open-source libraries.

**Summary**

This chapter discussed the shortcomings of the traditional web interface, identifying specific problems we want to overcome. We also introduced the various building blocks of an Ajax application and discussed how they work together.

In the following chapters we shall look at these components in more detail, eventually using them to build a complete Ajax application.

SYMBOLS

&& (And operator), 120
* (multiplication operator), 198
@ characters, PHP methods, 253
\ (backslashes)
  escaping strings, 202
  \n character sequence, newline characters, 192
[] (braces)
  code indentation rules, 216
  loop syntax, 126
  use in conditional statements, 216
[] (brackets), use in conditional statements, 216
$ (dollar sign)
  $ SERVER global array variable, 320
  $() function, 306
  $F() function, 307
  variables, 195
= (equal sign)
  = (assignment operator), 119
  == (equality operator), 119, 203
! (Not operator), 120
< (less than sign)
  <ajax-response> elements, Rico, 317-319s
  <div> ... <div> elements, 176
  <div> containers, 179
  <response> elements, Rico, 317-319
  <script> ... <script> elements, 177
— (minus sign), 84
  — (decrement operator), 199
  - (subtraction operator), 198
% (modulus operator), 199
. (period), 71
|| (Or operator), 119-120
A Badly Formatted Script That Displays the Date and Time (Listing 1.3), 192

abortion method, 154
active page elements, designing, 299
addition (+) operator, 198
AHAH (Asynchronous HTML and HTTP). See also HTML; HTTP
advantages of, 248
callAHAH() functions, 250-251
myAHAHlib.js, 249-251
metatag information, retrieving from URL, 252-253
responseText property, 255
responseAHAH() functions, 250-251
Ajax
application examples, 44
application flow, example of, 47-48
client-server interaction, 41-44
inappropriate situations for using, 299
objects
Ajax.PeriodicalUpdater class, 310
Ajax.request class, 308
Ajax.Updater class, 309-310
AjaxEngine objects, 316-317
Ajax Engines, 44, 316-317
<ajax-response> elements, Rico, 317-319
alert() function, 68
alt attribute (image tags), 26
Amazon.com REST API, 275-278
anchor objects, 77
anchor tags (HTML), 27
anchors, 77
And operator (&&), 120
Apache Web Server website, 11
appendChild() method, 259-261
applications, designing, 299
arguments, 104, 225
arithmetic operators
addition (+), 198
compound operations, 199-200
division (/), 199
modulus (%), 199
multiplication (*), 198
subtraction (-), 198
ARPAnet, Internet development, 9
array function, 207
arrays, 94
accessing, 207
assigning values to, 95
associative, textual key names, 208
contents, searching, 209
creating, 95, 207
declaring, 95, 207
elements, accessing, 96
function of, 206-209
index values, 207
length property, 95
looping through
foreach loop, 207
while loop, 207-208
sorting, 98-100
string arrays, 96-98
array_search function, array manipulation, 209
ASCII text, server responses, 47
assigning values to
arrays, 95
strings, 89-90
variables, 84
assignment operator (=), 119
associative arrays, textual key names, 208
asterisk (*), multiplication operator, 198
asynchronous server communications, 44
asynchronous server requests, 46, 157-162
at sign (@), PHP methods, 253

B
Back button, 79, 296
background property, 36, 39
background-color property, 38
backs slashes (\)
         escaping strings, 202
     \n character sequence, newline characters, 192
bandwidth, defining, 42
best practices, 67
body tags (HTML), 24-25
bookmarks, troubleshooting, 297
Boolean data types, 197
Boolean operators. See logical operators
Boolean values, 87, 216
braces ({})
         code indentation rules, 216
loop syntax, 126
use in conditional statements, 216
brackets ([]), use in conditional statements, 216
break statement, escaping from infinite loops, 130
breaking loops, 222
browsers
         availability of, 13
         caches
callAjax() functions, 160-162
GET requests, 301
server requests, 160, 162
callAHAH() functions, 250-251
callAjax() function, 159
browser caches, 160-162
launching, 165
callback functions, 162-163
AHAH, 250-251
basic application creation example, 179-180
JavaScript libraries, 288-290
launching, 165
myAJAXlib.js, 291
RSS headline readers, creating, 266-267
calling functions, 105-106
callRICO() function, 318
capitalization in strings, 205

How can we make this index more useful? Email us at indexes@samspublishing.com
case sensitivity

case sensitivity, 65
strings, 205
variables, 196

ceil function, rounding number functions, 200

center tags (HTML), 28
CERN (Conseil Europeen pour le Recherche Nucleaire), Internet development, 10
change() function, 335-336
character strings, split() method, 245
charAt method, responseText property, 169
charAT() method, 93
child nodes, adding to DOM, 259
child objects, 109
childNodes property, 261

cinematic effects (Rico), 324
classes (OO programming), 232
appearance of, 232-233
constructors, 234
definitions, 232-233
functions, 232
inheritance, 232
methods, 232-234
object instances, creating, 233
private methods, 232
public methods, 232
third-party, 232-236
when to use, 232

client-server interactions versus Ajax, 41-42

client-side programming, defining, 14

closing tags (?), 189
code
braces ({}), indentation rules, 216
comments, 193
functions, uses for, 223
modular, 224
platform tests, troubleshooting, 300
color, style sheets, 38
color property, 36-39

combining
conditions, 119-121
values of strings, 89
words, use of underscore characters, 196

comments, 66
code, 193
HTML, 25
Using Comments in a Script (Listing 1.4), 193

comparison operators, strings, 203
compound operators, 199-200
concatenation operators, strings, 202
conditional expressions, 118-119
conditional operators, 119

conditional statements, 62, 215
Boolean values, 216
braces ({}), 216
brackets ([],), 216
logical operators, 217-218
multiple condition branches, 218-219

operators, 216-217
switch statement, 219-220

conditions, combining, 119-121

constructors
class methods, 234
functions, 110
continue statement, 130

converting
case of strings, 91-92
data types, 88
date formats, 143, 213

count function, array manipulation, 209

CreateAttribute method, 261
createElement() method, 260-261
createTextNode() method, 259-261

CSS (Cascading Style Sheets), 30, 39
custom objects, 72

data types, 86-87

Boolean, 197
converting between, 88
double, 197
gettype function, 198
integer, 197
NULL values, 200
numeric, 200
querying, 198
settype function, 198
string, 197
data() function, 178

date and time, displaying, 54-60
date command, 189
date formats
  converting, 143, 213
  listing of, 213
  storage overview, 209-210
  Unix timestamp, 210
date function, 210-211, 224
Date object, 56, 140-141
Date.parse() method, 143
Date.UTC() method, 143
decimal numbers, rounding, 136
declaring
  arrays, 95, 207
  variables, 82, 196
decrement operator (—), 199
decrementing variables, 84
default argument values, functions, 226-227
defining
  functions, 104
    multiple parameters, 105
    simple example, 224-225
  objects, 110
DELETE requests, 273-274
developer’s tokens, 276
displaying
  dates and times, 54-60
  error messages, 60
Displaying the System Date and Time (Listing 1.1), 190-191
dissecting strings
  sublen function, 206
  subpos function, 206
  substr function, 205-206
<div> ... </div> elements, 176
<div> containers, 179
division operator (/), 199
DNS (Domain Name Service) servers, 14
do loops, 221
do...while loops, 128
doAjax function, 289-293
DOCTYPE elements, 23
document object, 74
  methods, 76
  properties, 75
document.write statement, 56
Dojo library, 144
dollar sign ($)
  $ SERVER global array variable, 320
  $( ) function, 306
  $F( ) function, 307
  variables, 195
DOM (Document Object Model)
  appendChild() method, 259
  child nodes, adding to, 259
  createElement() method, 260
  createTextNode() method, 259
document methods
  table, 261
elements, deleting, 267
  getElementsByTagName method, 258
else clause, multiple condition branches, 218-219
else keyword, 121-124
elseif keyword, multiple condition branches, 218-219
email
  gmail web mail service (Google), 45
  Internet development, 10
  return values, mail function example, 226
email_validation_class (third-party), 234-235

double data types, 197
double quotes (“”), strings, 197, 202
downloading Script.aculo.us library, 325

document properties table, 261
objects, 72
  document, 74-76
  hierarchy, 73
  properties, 73
E

echo command, 189
  A Badly Formatted Script That Displays the Date and Time (Listing 1.3), 192
  browser, outputting to, 191-192
  Using echo to Send Output to the Browser (Listing 1.2), 191-192
else clause, multiple condition branches, 218-219
else keyword, 121-124
elseif keyword, multiple condition branches, 218-219
email
  gmail web mail service (Google), 45
  Internet development, 10
  return values, mail function example, 226
email_validation_class (third-party), 234-235

How can we make this index more useful? Email us at indexes@samspublishing.com
Engines

**Engines (Ajax), 44**

**equal sign (=)**

= (assignment operator), 119

== (equality operator), 119, 203

**error handling**

application design, 301

Back button codes, 296

bookmarks, 297

browser caches, 301

code, platform tests, 300

GET requests, 301-302

JavaScript libraries, 293

links, 297

page design, 299

Permission Denied errors, 302

POST requests, 302

security, 300

spiders, 298

unsupported browsers, 297-298

user feedback, 297

**error messages, 60**

**escape characters (\), strings, 202**

**escaping infinite loops, 130**

**eval() function, JavaScript libraries, 288-290**

**event handlers**

basic application creation example, 180

example of, 67-68

myAJAXlib.js, calls for, 291

**exclamation point (!), Not operator, 120**

**explicit newline characters, \n, 192**

**expressions**

operators, precedence rules, 85-86

use in variables, 196-197

**F**

**$F() function, 307**

**feedback (user)**

basic application creation example, 182-183

JavaScript libraries, 293

server requests, 172-173

troubleshooting, 297

**firstChild property, 261**

**float widths, string formatting, 204**

**floor function, rounding number functions, 200**

**flow control, 117**

conditional statements, 215

Boolean values, 216

logical operators, 217-218

multiple condition branches, 218-219

operators, 216-217

switch statement, 219-220

if statement

conditional expressions, 118

logical operators, 119-121

**loops**

breaking out of, 222

do, 221

for, 221-222

nested, 222

while, 220-221

**font-family property, 36-38**

**font-size property, 36**

**font-style property, 36**

**font-weight property, 36**

**for loops, 125, 221-222**

**for statement, 63, 125-128**

**for...in loops, 131-133**

**foreach loops, looping through arrays, 207**

**Form objects, prototype.js, 307**

**formatting strings**

format codes, 204

printf function, 203

sprintf function, 204-205

**Forth programming language, 127**

**Forward button, creating, 79**

**Frameworks (Ajax), 44**

**FTP (File Transfer Protocol), Internet development, 10**

**Fuchs, Thomas, 144**

**function calls, 62**

**functions, 62, 103**

$(), 306

$F(), 307

alert(), 68

arguments, 104, 225

array, 207

darray manipulation, 208-209
href property

href property (window objects), 78
HTML (Hypertext Markup Language), 21. See also AH AH, HTTP
  attributes, adding to, 25
  basic application creation example, 176
  color values, 26
  common tags table, 29-30
  containers, 25
  defining, 22
  <div> ... <div> elements, 176
  <div> containers, 179
  <hr>, 39
  hyperlinks, 27
  loading, 23
  myAjAXlib.js, 291
  responseText property, 242-243
  RSS headline readers, creating, 263
  saving, 23
  <script> ... <script> elements, 177
  seville.html document example, 28
  tags, 22-23
    anchor tags, 27
    body tags, 24-25
    center tags, 28
    event handlers, 64
    head tags, 24
    metatags, 251-253
    table tags, 27-29
  title tags, 24
testpage.html document example, 22
tool requirements, 22
XOAD, 334
  change() function, 335-336
  XOAD
    HTML::getElementById() method, 335-336
  XOAD
    HTML::getElementByTag Name() method, 336-337
  word processors, 22
HTTP (Hypertext Transfer Protocol), 10. See also AH AH; HTML
  server response status codes, 163
  SOAP requests, sending, 281
hyperlinks, HTML, 27
hypertext, Internet development, 10

id values, 173
if statement
  conditional expressions, 118
  logical operators
    And, 120
    else keyword, 121
    Not, 120
    Or, 119
testing multiple conditions, 122-124
images
defining, 26
tags
  alt attribute, 26
  src attribute, 26
include keyword, library function files, 229
include once keyword, library function files, 229
increment operator (++), 84, 132, 199
incrementing variables, 84
indenting code, braces (), 216
index values, assigning arrays, 207
indexOf() method, 94, 169
infinite loops, 129-130, 220
inheritance, classes, 232
initial expression, 125
instances (objects), creating, 111
class objects, 233
XMLHttpRequest objects, 151-153
integer data types, 197
Internet, development of, 9-10
Internet Explorer 6.0, security settings, 58
in_array function, array manipulation, 209
IP addresses, defining, 14
J

JavaScript libraries
Back button codes, 296
callback functions, 288-290
doAjax functions, 289-293
error handling, 293
eval() function, 288-290
GET requests, 288
myAHAHlib.js, 286-287
myAJAXlib.js, 289-290
callback functions, 291
event handler calls, 291
GET requests, 291
HTML pages, 291
PHP scripts, 291
responseText
properties, 291
usage example, 291-292
XML data, retrieving, 292
POST requests, 288, 293
prototype.js
$( ) function, 306
$F( ) function, 307
Ajax.PeriodicalUpdater
class, 310
Ajax.request class, 308
Ajax.Updater class,
309-310
download website, 305
Form objects, 307
getElements() method,
307
Rico, 315-324
serialize() method, 307
Stock Price Reader build
time example, 311-312
Try.these() function, 308
web applications, adding
to, 306
user feedback, 293
XMLHttpRequest instances,
creating, 287
join() method, 100
JSON (JavaScript Object
Notation), 309, 332

K - L

keywords, 139-140
keywords metatag, 251-253

large clock display, adding to
time and date script, 58-60
lastChild property, 261
lastIndexOf() method, 94, 169
length of arrays, calculating, 95
length property, 91, 95
less than sign (<)
<ajax-response> elements,
Rico, 317-319
<div> … <div>
elements, 176
<div> containers, 179
<response> elements, Rico,
317-319
<script> … <script> ele-
ments, 177
levels (DOM), 74

libraries
JavaScript
callback functions,
288-290
doAjax functions, 289-293
error handling, 293
eval() function, 288, 290
GET requests, 288
myAHAHlib.js, 286-287
myAJAXlib.js, 289-292
POST requests, 288, 293
prototype.js, 305-312,
315-324
user feedback, 293
XMLHttpRequest instances, 287
open source libraries,
Rico, 315
AjaxEngine instances,
316-317
callRICO() function, 318
cinematic effects, 324
drag-and-drop, 320-323
multiple page element
updates, 317
<response> elements, 317-319
usage example, 318
web applications, adding
to, 316
third-party libraries
Prototype, 143
Script.aculo.us, 144,
325-327
Yahoo! UI Library, 144

How can we make this index more useful? Email us at indexes@samspublishing.com
library file functions, creating, 229
link objects, 76-77
links
  style sheets, 38
troubleshooting, 297
underlining, 38
local variables, 83, 227-228
localtime variable, 56
location object, 78-79
location.reload() method, 79
location.replace() method, 79
logical operators
  And (&&), 120
  conditional statements, 217-218
  Not (!), 120
  Or (||), 119-120
loops, 63
  breaking out of, 222
  continue statement, 130
do loops, 221
for loops, 221-222
for statement, creating with, 125-128
for...in loops, 131-133
foreach loops, 207
infinite loops, 129-130, 220
nested loops, 222
while loops, 207-208, 220-221
while statement, creating with, 128
Lynx text-based web browsers, 13

M

mail function
  default argument values, 226-227
  return values, 226
margin-left property, 40
Math object, 135-136
Math.random() method, 137-139, 160
mathematical functions, 201
metatags
  keywords, 251-253
  myAHAHlib.js, 252-253
methods, 72, 109
  abort, 154
  appendChild(), 259-261
  charAt(), 93, 169
  classes, 233-234
  constructors, 234
  CreateAttribute, 261
  createElement(), 260-261
  createTextNode(), 259-261
  getAllResponseHeaders, 154
  getElementByld(), 173, 180, 258
  getElementsByTagName, 179
  getElements(), prototype.js, 307
  GetElementsByld, 261
  getElementsByName(), 171-172, 258, 261
  getResponseHeader, 154
  HasChildNodes, 261
  history.back(), 78
  history.forward(), 78
  history.go(), 78
  indexOf(), 94, 169
  join(), 100
  lastIndexOf(), 94, 169
  location.reload, 79
  location.replace, 79
  Math.random(), 137-139, 160
  open, 154-155
  registerDraggable, 320
  registerDropZone, 320
  RemoveChild, 261
  send, 154-155
  serialize(), prototype.js, 307
  setRequestHeader, 154-156
  sort(), 98-100
  split(), 97, 245
  substring(), 93, 169
  toLowerCase(), responseText
  property, 169
  toUpperCase(), responseText
  property, 169
  XMLHttpRequest object, 154
  XOAD
    HTML::getElementById(),
    335-336
  XOAD
    HTML::getElementsByNam
    e(), 336-337
Microsoft typography website, 36
minus sign (-), 84
  — (decrement operator), 199
  - (subtraction operator), 198
mktime function, creating timestamps, 212
MochiKit library, 145
modular code, 224
modulus operator (%), 199
Mosaic, Internet development, 10
multiplatform code tests, 300
multiple conditions
   conditional statements, 218-219
testing, 122-124
multiple scripts, order of operation, 64
multiplication (*) operator, 198
myAHAHlib.js, 249-251, 286-287
   metatag information, retrieving from URL, 252-253
   responseText property, 255
myAJAXlib.js, 289-290
   callback functions, 291
event handler calls, 291
GET requests, 291
HTML pages, 291
PHP scripts, 291
responseText properties, 291
usage example, 291-292
XML data, retrieving, 292

naming conventions, 65, 195-196
NaN (non a number), 88
navigation tools, creating
   Back/Forward buttons, 79
nested loops, 222
newline characters, \n, 192
nextSibling property, 261
nodeName property, 261
nodes (DOM)
   child nodes, 259
document methods table, 261
node methods table, 261
node properties table, 261
nodeType property, 261
nodeValue property, 261
Not operator (!), 120
null value, 87, 200
numeric arrays, sorting, 98-100
numeric data types, 200
numeric functions
   random, 201
   rounding numbers, 200-201

O

object hierarchy (DOM), 73
object-oriented programming, see
   OO (object-oriented) programming
objects, 108
   Ajax
      Ajax.PeriodicalUpdater
t      class, 310
      Ajax.request class, 308

Ajax.Updater class, 309-310
AjaxEngine, instances in Rico, 316-317
built-in, 72, 112-114
child objects, 109
creating, 108, 111
defining, 110
document, 74
   methods, 76
   properties, 75
DOM, 72
Form, prototype.js, 307
instances, creating, 111, 151-153, 233
location, 78-79
methods, 72
naming conventions, 65
   properties, 71, 108
XMLHTTPRequest
   basic application creation example, 177-178
   callAjax() function, 159
   instances, creating, 151-153
   JavaScript libraries, creating, 287
   methods
      open, 155
      send, 155
   methods, list of, 154
   properties, list of, 154
responseAjax() function, 159
server requests, 157-165

How can we make this index more useful? Email us at indexes@samspublishing.com
status property, 164
statusText property, 164
uses of, 150
XMLHTTPRequest, readyState property, 162-163
onBlur event handler, 165
onLoad() event handler, basic application creation example, 180
onreadystatechange property, 154
OO (object-oriented) programming
advantages of, 232
classes
appearance of, 232-233
constructors, 234
definitions, 232-233
functions, 232
inheritance, 232
methods, 232-234
objects, instance creation, 233
private methods, 232
public methods, 232
third-party, 232-236
PHP Classes website, 231
PHP functionality, 231
PHP.net website
resources, 233
when to use, 232
open method, 154-155
open source libraries, Rico, 315
AjaxEngine instances, 316-317
callRICO() function, 318
cinematic effects, 324
drag-and-drop, 320-323
multiple page element updates, 317
<response> elements, 317-319
usage example, 318
web applications, adding, 316
operators, 85
+= operator, 84
arithmetic
addition (+), 198
division (/), 199
modulo (%), 199
multiplication (*), 198
subtraction (-), 198
assignment (=), 119
compound, 199-200
conditional statements, 216-217
decrement (—), 199
equality (==), 119, 203
increment (++), 199
logical operators
And (&&), 120
conditional statements, 217-218
Not (!), 120
Or (||), 119-120
precedence rules, 85-86
P
parentNode property, 261
parseFloat() function, 88
parseInt() function, 88
parsing, responseXML property, 172
percent sign (%), modulus operator, 199
period (.), 71
Permission Denied errors, troubleshooting, 302
PHP (Hypertext Preprocessor), 187
$ SERVER global array variable, 320
?php tag, 189-190
methods, 253
running locally from PC, 190
scripts
basic application creation example, 178-179
myAJAXlib.js, 291
XOAD, 331
cache handling, 338
client controls, customizing, 338
downloading/installing, 332
events, 338
header() function, 334
JSON, 332
runServer() function, 334
simple page example, 332-334
How can we make this index more useful? Email us at indexes@samspublishing.com
prototype.js

Q - R

quotation marks
  double quotes (" ")
  strings, 197
  variables, 202
  single quotes (')
  strings, 202
  variables, 197

question mark (?)
  ?php tag, 189-190
  closing tags, 189

random numbers
  generating, 136
    example script, 137-139
    rand function, 201
    srand function, 201

readyState property, 154, 162-163

<response> elements, 317-319
  usage example, 318
  web applications, adding to, 316
serialize() method, 307
Stock Price Reader build example, 311-312
Try,these() function, 308
web applications, adding to, 306

public methods (classes), 232
PUT requests, 273-274

recommend web browsers, 54
registerDraggable method, 320
registerDropZone method, 320
RemoveChild method, 261
require keyword, library function files, 229
require once keyword, library function files, 229
reserved words, 66
<response> elements, Rico, 317-319
responseAHAH() functions, 250-251
responseAjax() function, 159, 163
responseText property, 154, 239
  character strings, 240-242
  formatted data, 244-245
  HTML, 242-243
  manipulation methods list, 169-170
  myAHAHlib.js, 255
  myAJAXlib.js, 291
  null values, 168
  returned text, 240-242
  values, displaying, 168-169
responseXML property, 154, 170
  parsing, 172
  stored values, 258
  web pages, adding elements to, 259-261

REST (Representational State Transfer)
  Amazon.com REST API, 275-278
  articles, uploading, 275
DELETE requests, 273-274
  example of, 273
GET requests, 273-276
POST requests, 273-275
  principles of, 272
PUT requests, 273-274
SOAP versus, 283
stateless operations, 274

return keyword, 107
return values, functions
  failure, 225-226
  mail function example, 226
  success, 225-226

returning
  single characters from strings, 93
  time in UTC, 142

Rico, 315
  AjaxEngine instances, 316-317
  callRICO() function, 318
  cinematic effects, 324
drag-and-drop, 320-323
  multiple page element updates, 317
<response> elements, 317-319
usage example, 318
web applications, adding to, 316

rounding decimal numbers, 136
rounding number functions
  ceil, 200
  floor, 200
  round, 201
RSS
feeds, 262	headline readers, creating, 262-265
callback functions, 266-267
HTML page, 263
server scripts, 268-269
runServer() function, 334

S
scope of variables, 82
<script> ... <script> elements, 177
Script.aculo.us library, 144, 325-327
scripts
A Badly Formatted Script That Displays the Date and Time (Listing 1.3), 192
adding to HTML documents, 57
comments, adding, 66
creating, required tools for, 54
date and time, displaying, 55-60
Displaying the System Date and Time (Listing 1.1), 190-191
flow control
conditional statements, 215-220
loops, 220-222
library functions, including, 229
order of operation, 64
random numbers, generating, 137-139
Using Comments in a Script (Listing 1.4), 193
Using echo to Send Output to the Browser (Listing 1.2), 191-192
search engine spiders, troubleshooting, 298
security
IE 6.0, settings for, 58
troubleshooting, 300
XMLHTTPRequest objects, 46
semicolons (;) 56, 61, 189
send method, 154-155
serialize() method, prototype.js, 307
server-side programming, defining, 12
servers
asynchronous communications, 44
requests
asynchronous requests, 46
basic application creation example, 178
browser caches, 160-162
callback functions, 162
GET requests, 159
monitoring status of, 162-163
progress notifications, 172-173
setRequestHeader method, 154-156
settype function, 198
shorthand conditional expressions, 121
single quotes (‘) strings, 202
variables, 197
single-line comments, 193
sizeof() function, 244
slashes (/)
/ (division operator), 199
/*...*/ (use in multiple-line comments), 193
// (use in single-line comments), 193
How can we make this index more useful? Email us at indexes@samspublishing.com
SOAP (Simple Object Access Protocol), 278

development of, 279

namespaces, 280

requests

    Ajax usage example, 282
code example, 281

    components of, 279-280

    HTTP, sending via, 281

REST versus, 283

specification information

    website, 279

sort() method, 98-100

sorting

    numeric arrays, 98-100

    string arrays, 98

spiders (search engine), troubleshooting, 298

split() method, 97, 245

splitting strings, 97

sprintf function, string formatting, 204-205

srand function, random number generation, 201

src attribute (image tags), 26

statements, 61

    conditional, 62

    function calls, 62

    termination (;), 189

status property, 154, 164

statusText property, 154, 164

Stephenson, Sam, 143

Stock Price Reader build example, 311-312

storing date formats, 209-210

string arrays

    creating, 96-97

    sorting, 98

string data types, 197

string objects, creating, 89

strings, 56, 87

    assigning values to, 89-90

    capitalization in, 205

    case sensitivity, 91-92, 205

    comparing, 203

    concatenation operator, 202

    dissecting

        sublen function, 206

        subpos function, 206

        substr function, 205-206

    escape characters (\), 202

    formatting

        format codes, 204

        printf function, 203

        sprintf function, 204-205

    function of, 202

    length of, calculating, 91

    length property, 91

    quotation marks

        double (“ “), 197, 202

        single (‘), 197, 202

    returning single characters from, 93

    splitting, 97

    substrings, 92-94

    variables, 197

strtolower function, string capitalization, 205

strtotime function, 213

strtoupper function, string capitalization, 205

style sheets

    adding, 33-34

    class attribute, 31-33

    declarations, 31

    embedded, 30

    inline, 30

    linked, 30

    links, 38

    precedence, 34-35

    rules, 31

    text, 36

    <style> tag, 36

substr function, string dissection, 206

subpos function, string dissection, 206

substr function, string dissection, 205-206

substring() method, 93, 169

substrings

    index values, 92

    locating, 94

subtraction (–) operator, 198

switch statement, 124

    conditional statements, 219-220

    syntax, 125

syntax

    case sensitivity, 65

    comments, 66

    naming conventions, 65

    reserved words, 66

    switch statement, 125
table tags (HTML), 27, 29
tags
  ?php, 189-190
closing (?), 189
HTML tags
  anchor tags, 27
  body tags, 24-25
  center tags, 28
  head tags, 24
  table tags, 27-29
title tags, 24
image tags, 26
metatags
  keywords, 251-253
  myAHAHlib.js, 252-253
processing instructions, 189-190
<style> tag, 36
testing
  color, 39
date and time script, 58
  multiple conditions, 122-124
text-align property, 36
text-based web browsers, 13
text-decoration property, 36-38
text-indent property, 36
third-party classes, 234-236
third-party libraries
  Prototype, 143
  Script.aculo.us, 144
time
displaying, 54-60
  zones, 142
time and greeting example, 123-124
time function, locating timestamps, 210
time.php script, date and time display, 190-191	
timestamps
  converting date formats to, 213
  creating (mktime function), 212
date function, 210-211
  server requests, 162
time function, 210
title tags (HTML), 24
toLocalString() function, 142
toLowerCase() method, 91, 169
toUpperCase() method, 91, 169
toUTCString() function, 142
troubleshooting
  application design, 301
  Back button codes, 296
  bookmarks, 297
  browser caches, 301
code, platform tests, 300
GET requests, 301-302
links, 297
page design, 299
Permission Denied errors, 302
POST requests, 302
security, 300
spiders, 298
unsupported browsers, 297-298
user feedback, 297
Try.these() function, 308
underscore characters, combining words, 196
Unix timestamp format
  best uses, 210
drawbacks, 210
ease of use, 210
mktime function, 212
  starting value, 210
unsupported browsers, troubleshooting, 297-298
URL (Uniform Resource Locators), creating RSS headline readers, 262-265
  callback functions, 266-267
  HTML page, 263
  server scripts, 268-269
user feedback
  basic application creation example, 182-183
  JavaScript libraries, 293
  server requests, 172-173
troubleshooting, 297
Using echo to Send Output to the Browser (Listing 1.2), 191-192
UTC (Universal Time Coordinated), 54, 142
utctime variable, 56

variables, 55
  arguments, 104
  assigning values to, 84

How can we make this index more useful? Email us at indexes@samspublishing.com
variables

declaring, 82, 196
decrementing, 84
dollar sign ($), 195
expressions, 85, 196-197
fixed values, 196
global
  creating, 83
  scope of, 227-228
incrementing, 84
invalid names, 196
local, 83, 227-228
naming, 82, 195
  case sensitivity, 196
  conventions, 65, 196
operators, precedence rules, 85-86
scope of, 82
  global, 227-228
  local, 227-228
strings, 197
underscore characters, word combinations, 196
valid names, 196
values, 195
verifying date and time script, 58

W

W3C (World Wide Web Consortium), 74, 279, 301
web browsers
  availability of, 13
  caches
    callAjax() functions, 160-162
GET requests, 301
server requests, 160-162
defining, 13
graphics browsers, 13
Lynx text-based browsers, 13
style sheet properties, 38
text-based browsers, 13
unsupported browsers, troubleshooting, 297-298
web server interaction, 10
web pages
  defining, 11
  elements, adding via
    responseXML property, 259-261
  id values, 173
  server-side scripting, 188
web servers
  defining, 11
  server-side scripting of web pages, 188
  web browser interaction, 10
web services
  example of, 272
REST
  Amazon.com REST API, 275-278
  articles, uploading, 275
  DELETE requests, 273-274
  example of, 273
GET requests, 273-276
lists of available articles, reading, 274-275
particular articles, retrieving, 275
POST requests, 273-275
  principles of, 272
  PUT requests, 273-274
  SOAP versus, 283
  stateless operations, 274
SOAP, 278
  development of, 279
  namespaces, 280
  requests, 279-282
  REST versus, 283
  specification information website, 279
websites
  Apache Web Server website, 11
  GNU.org, date formats, 213
  JSON, 309
  Lynx text-based web browsers, 13
  Microsoft typography website, 36
  PHP Classes, 231, 234
  PHP.net
    array functions, 208
    mathematical function resources, 201
    online manual documentation, 223
    OO programming resources, 233
    string functions listing, 205
  prototype.js download website, 305
  W3C, 279, 301
while loops, 220-221
arrays, looping through, 207-208
example of, 128
whitespace, 66
with keyword, 139-140
word processors, HTML, 22

X

XML (Extensible Markup Language)
data, retrieving, 292
responseXML property
stored values, 258
web pages, adding elements to, 259-261
RSS headline readers, creating, 262-265
callback functions, 266-267
HTML page, 263
server scripts, 268-269
server responses, 47
XMLHttpRequest objects, 45
basic application creation example, 177-178
callAjax() function, 159
instances, creating, 151-153
JavaScript libraries, creating, 287
methods, list of, 154
open method, 155
properties, list of, 154
readyState property, 162-163
responseAjax() function, 159
security, 46
send method, 155
server requests, 46
browser caches, 160-162
sending, 157-159
status, monitoring, 162-163
timestamps, 162
server-side scripts, 46
status property, 164
statusText property, 164
uses of, 150
XOAD (XMLHTTP Object-oriented Application Development), 331
cache handling, 338
client controls, customizing, 338
Controls class, 338
downloading/installing, 332
events, 338
header() function, 334
HTML, 334
change() function, 335-336
XOAD
HTML::getElementById() method, 335-336
XOAD
HTML::getElementsByTag Name() method, 336-337
JSON, 332
runServer() function, 334

Y - Z

Yahoo! UI Library, 144