Contents at a Glance

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

Part I: Start Programming Today

HOUR 1  Hands-On Programming ........................................ 5
        2  Process and Techniques ..................................... 21
        3  Designing a Program ....................................... 39
        4  Getting Input and Displaying Output ....................... 57
        5  Data Processing with Numbers and Words ................. 75
        6  Controlling Your Programs ................................ 95
        7  Debugging Tools ............................................. 113

Part II: Programming Fundamentals

HOUR 8  Structured Techniques ..................................... 131
        9  Programming Algorithms ................................ 145
       10  Having Fun with Programming ............................ 173
       11  Advanced Programming .................................... 187

Part III: Object-Oriented Programming with Java

HOUR 12  Programming with Java .................................. 201
       13  Java’s Details ............................................ 221
       14  Java Has Class ........................................... 239
       15  Applets and Web Pages .................................. 255

Part IV: Other Programming Languages

HOUR 16  HTML5 and CSS3 .......................................... 269
       17  JavaScript and AJAX ..................................... 285
       18  Scripting with PHP ....................................... 301
       19  Programming with C and C++ ............................ 333
       20  Programming with Visual Basic 2012 ...................... 359
       21  C# and the .NET Framework .............................. 375
Part V: The Business of Programming

HOUR 22 How Companies Program ........................................... 389
23 Distributing Applications ................................................. 407
24 The Future of Programming ............................................... 413
Index ................................................................................. 423
Table of Contents

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

Part I: Start Programming Today

HOUR 1: Hands-On Programming ......................................... 5
  Get Ready to Program .................................................. 5
  What a Computer Program Does ...................................... 6
  Common Programming Misconceptions .............................. 7
  Many Programs Already Exist ......................................... 9
  Programmers Are in Demand ......................................... 9
  The Real Value of Programs ......................................... 9
  Users Generally Don’t Own Programs ............................ 10
  Giving Computers Programs ....................................... 10
  Your First Program .................................................... 13
  Clarifying Comments .................................................. 14
  Entering Your Own Program ....................................... 16
  Summary ................................................................. 18
  Q&A .................................................................. 19
  Workshop ................................................................. 19

HOUR 2: Process and Techniques ......................................... 21
  Understanding the Need for Programs ............................ 21
  Programs, Programs, Everywhere ................................. 24
  Programs as Directions .............................................. 25
  Summary .................................................................. 35
  Q&A .................................................................. 36
  Workshop ................................................................. 36

HOUR 3: Designing a Program ............................................. 39
  The Need for Design .................................................... 39
  User-Programmer Agreement ....................................... 40
Summary ......................................................... 183
Q&A ............................................................. 184
Workshop ....................................................... 184

HOUR 11: Advanced Programming ......................... 187
JavaScript’s Programming Weakness ....................... 187
Writing Cookies .............................................. 190
Reading Cookies ............................................ 192
Deleting Set Cookies ........................................ 193
Summary ....................................................... 198
Q&A ............................................................. 198
Workshop ....................................................... 199

Part III: Object-Oriented Programming with Java

HOUR 12: Programming with Java .......................... 201
Introducing Java .............................................. 202
Java Provides Executable Content ......................... 204
Seamless Execution ......................................... 206
Multi-Platform Executable Content ....................... 206
Java Usage Summary ....................................... 208
You'll Start with Standalone Java ......................... 209
Java's Interface .............................................. 210
Security Issues .............................................. 211
Give Java a Spin ............................................ 212
Java Language Specifics .................................... 213
Get Ready to Begin ......................................... 217
Summary ....................................................... 218
Q&A ............................................................. 218
Workshop ....................................................... 218

HOUR 13: Java’s Details ....................................... 221
Defining Java Data ........................................... 221
Operators ...................................................... 227
Contents

Programming Control ........................................... 230
From Details to High-Level .................................... 235
Summary ............................................................ 236
Q&A ................................................................. 236
Workshop .......................................................... 236

HOUR 14: Java Has Class ........................................ 239
Using NetBeans to Run a Java Program .......................... 239
Going GUI ......................................................... 244
Java and OOP ....................................................... 245
Overview of Classes ............................................... 246
Do You Understand OOP? ....................................... 248
Methods Do the Work in Classes ................................. 249
Summary ............................................................ 251
Q&A ................................................................. 252
Workshop .......................................................... 252

HOUR 15: Applets and Web Pages ............................... 255
About Writing Java Applets ...................................... 255
Creating a Java Applet ........................................... 256
Placing the Applet Inside the Web Page ......................... 262
Viewing the Applet Inside the Web Page ....................... 264
Summary ............................................................ 265
Q&A ................................................................. 266
Workshop .......................................................... 266

Part IV: Other Programming Languages

HOUR 16: HTML5 and CSS3 ..................................... 269
HTML Programming ............................................... 269
A Simpler Example ............................................... 274
A Quick HTML Primer .......................................... 275
Using CSS to Control How Your Text Looks .................. 278
Including Graphics in Your Website with HTML ............... 281
Fundamental Differences Between C and C++ .......................... 347
Introducing C++ Objects ..................................................... 348
Things to Come ................................................................. 354
Summary .............................................................................. 355
Q&A ..................................................................................... 356
Workshop ................................................................. 356

**HOUR 20: Programming with Visual Basic 2012** ........................................ 359
Reviewing the Visual Basic Screen ................................................ 359
Creating a Simple Application from Scratch ................................. 361
Other Visual Basic Programming Considerations ............................. 368
Your Next Step ...................................................................... 371
Summary .............................................................................. 371
Q&A ..................................................................................... 371
Workshop ................................................................. 372

**HOUR 21: C# and the .NET Framework** .................................................. 375
Understanding the Purpose of .NET ............................................. 375
The Common Language Runtime ................................................. 376
The Framework Class Library ...................................................... 377
Parallel Computing Platform ..................................................... 378
Dynamic Language Runtime ...................................................... 378
The C# Language ..................................................................... 378
Summary .............................................................................. 387
Q&A ..................................................................................... 387
Workshop ................................................................. 388

**Part V: The Business of Programming**

**HOUR 22: How Companies Program** .................................................. 389
Data Processing and Information Technology Departments .......... 389
Computer-Related Jobs ............................................................. 393
Job Titles .............................................................................. 394
Structured Walkthroughs ............................................................ 400
<table>
<thead>
<tr>
<th>Chapter Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Putting a Program into Production</td>
<td>401</td>
</tr>
<tr>
<td>Consulting</td>
<td>403</td>
</tr>
<tr>
<td>Summary</td>
<td>404</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>404</td>
</tr>
<tr>
<td>Workshop</td>
<td>404</td>
</tr>
<tr>
<td><strong>HOUR 23: Distributing Applications</strong></td>
<td>407</td>
</tr>
<tr>
<td>Issues Surrounding Software Distribution</td>
<td>407</td>
</tr>
<tr>
<td>Using Version Control</td>
<td>410</td>
</tr>
<tr>
<td>Summary</td>
<td>410</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>410</td>
</tr>
<tr>
<td>Workshop</td>
<td>411</td>
</tr>
<tr>
<td><strong>HOUR 24: The Future of Programming</strong></td>
<td>413</td>
</tr>
<tr>
<td>Some Helpful Tools</td>
<td>413</td>
</tr>
<tr>
<td>Will Programming Go Away?</td>
<td>416</td>
</tr>
<tr>
<td>Your Ongoing Training Needs</td>
<td>419</td>
</tr>
<tr>
<td>Summary</td>
<td>420</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>421</td>
</tr>
<tr>
<td>Workshop</td>
<td>421</td>
</tr>
<tr>
<td><strong>Index</strong></td>
<td>423</td>
</tr>
</tbody>
</table>
About the Authors

**Greg Perry** is a speaker and writer in both the programming and applications sides of computing. He is known for bringing programming topics down to the beginner's level. Perry has been a programmer and trainer for two decades. He received his first degree in computer science and then earned a Master's degree in corporate finance. Besides writing, he consults and lectures across the country, including at the acclaimed Software Development programming conferences. Perry is the author of more than 75 other computer books. In his spare time, he gives lectures on traveling in Italy, his second-favorite place to be.

**Dean Miller** is a writer and editor with more than 20 years of experience in both the publishing and licensed consumer products businesses. Over the years, he has created or helped shape a number of bestselling books and series, including *Sams Teach Yourself in 21 Days*, *Sams Teach Yourself in 24 Hours*, and the *Unleashed* series, all from Sams Publishing. He has written books on C programming and professional wrestling, and is still looking for a way to combine the two into one strange amalgam.
Dedication

To my wife and best friend, Fran Hatton, who’s always supported my dreams and was an incredible rock during the most challenging year of my professional career.
—Dean

Acknowledgments

Greg: My thanks go to all my friends at Pearson. Most writers would refer to them as editors; to me, they are friends. I want all my readers to understand this: The people at Pearson care about you most of all. The things they do result from their concern for your knowledge and enjoyment. On a more personal note, my beautiful bride, Jayne; my mother Bettye Perry; and my friends, who wonder how I find the time to write, all deserve credit for supporting my need to write.

Dean: Thanks to Mark Taber for considering me for this project. I started my professional life in computer book publishing, and it is so gratifying to return after a 10-year hiatus. I’d like to thank Greg Perry for creating outstanding first and second editions upon which this version of the book is based. It was an honor working with him as his editor for the first two editions and a greater honor to co-author this edition. I can only hope I did it justice. I appreciate the amazing work the editorial team of Elaine Wiley, Anne Goebel, and the production team at Pearson put into this book. On a personal level, I have to thank my three children, John, Alice, and Maggie and my wife Fran for their unending patience and support.
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 to let us 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 we cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail we receive, we might not be able to reply to every message.

When you write, please be sure to include this book’s title, edition number, and authors, as well as your name and contact information.

Email: feedback@samspublishing.com

Mail: Sams Publishing
ATTN: Reader Feedback
800 E. 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.
Learning how to program computers is easier than you might think. If you approach computers with hesitation, if you cannot even spell PC, if you have tried your best to avoid the subject altogether but can do so no longer, the book you now hold contains support that you can depend on in troubled computing times.

This 24-hour tutorial does more than explain programming. This tutorial does more than describe the difference between JavaScript, C++, and Java. This tutorial does more than teach you what programming is all about. This tutorial is a training tool that you can use to develop proper programming skills. The aim of this text is to introduce you to programming using professionally recognized principles, while keeping things simple at the same time. It is not this text’s singular goal to teach you a programming language (although you will be writing programs before you finish it). This text’s goal is to give you the foundation to become the best programmer you can be.

These 24 one-hour lessons delve into proper program design principles. You’ll not only learn how to program, but also how to prepare for programming. This tutorial also teaches you how companies program and explains what you have to do to become a needed resource in a programming position. You’ll learn about various programming job titles and what to expect if you want to write programs for others. You’ll explore many issues related to online computing and learn how to address the needs of the online programming community.

**Who Should Use This Book?**

The title of this book says it all. If you have never programmed a computer, if you don’t even like them at all, or if updating the operating system of your phone throws you into fits, take three sighs of relief! This text was written for you so that, within 24 hours, you will understand the nature of computer programs and you will have written programs.

This book is aimed at three different groups of people:

- Individuals who know nothing about programming but who want to know what programming is all about.
Companies that want to train nonprogramming computer users for programming careers.

Schools—both for introductory language classes and for systems analysis and design classes—that want to promote good coding design and style and that want to offer an overview of the life of a programmer.

Readers who seem tired of the plethora of quick-fix computer titles cluttering today’s shelves will find a welcome reprieve here. The book you now hold talks to newcomers about programming without talking down to them.

What This Book Will Do for You

In the next 24 hours, you will learn something about almost every aspect of programming. The following topics are discussed in depth throughout this 24-hour tutorial:

- The hardware and software related to programming
- The history of programming
- Programming languages
- The business of programming
- Programming jobs
- Program design
- Internet programming
- The future of programming

Can This Book Really Teach Programming in 24 Hours?

In a word, yes. You can master each chapter in one hour or less. (By the way, chapters are referred to as “hours” or “lessons” in the rest of this book.) The material is balanced with mountains of shortcuts and methods that will make your hours productive and hone your programming skills more and more with each hour. Although some chapters are longer than others, many of the shorter chapters cover more detailed or more difficult issues than the shorter ones. A true attempt was made to make each hour learnable in an hour. Exercises at the end of each hour will provide feedback about the skills you learned.
Conventions Used in This Book

This book uses several common conventions to help teach programming topics. Here is a summary of those typographical conventions:

- Commands and computer output appear in a special monospaced computer font. Sometimes a line of code will be too long to fit on one line in this book. The code continuation symbol (➥) indicates that the line continues.

- Words you type also appear in the monospaced computer font.

- If a task requires you to select from a menu, the book separates menu commands with a comma. Therefore, this book uses File, Save As to select the Save As option from the File menu.

In addition to typographical conventions, the following special elements are included to set off different types of information to make it easily recognizable.

TRY IT YOURSELF

The best way to learn how to program is to jump right in and start programming. These Try it Yourself sections will teach you a simple concept or method to accomplish a goal programmatically. The listing will be easy to follow and then the programs’ output will be displayed along with coverage of key points in the program. To really get some practice, try altering bits of the code in each of these sections in order to see what your tweaks accomplish.

NOTE

Special notes augment the material you read in each hour. These notes clarify concepts and procedures.

TIP

You’ll find numerous tips that offer shortcuts and solutions to common problems.

CAUTION

The cautions warn you about pitfalls. Reading them will save you time and trouble.
You have this book because you want to go farther than the typical computer user. You don’t just want to use computers; you want to write programs for them. You want the skills necessary to make computers do exactly what you want.

This 24-hour tutorial shows that you don’t have to be a wizard to become a proficient (or even an expert) programmer. By the time you finish this one-hour lesson, you will have entered your first computer program, run it, and looked at the results. This chapter springboards you into programming so you can get a feel for programming right away. Once you’ve had this hands-on experience, subsequent hours will explain more of the background you need to understand how to design and write programs.

The highlights of this hour include:

▶ Learning what a program does
▶ Understanding the truth behind common programming myths
▶ Mastering source code concepts
▶ Entering and running your first program

Get Ready to Program

You’ll find this hour divided into these two areas:

▶ An introduction to the nature of programming
▶ A hands-on practice with your first programming language

Most new programmers want to start programming right away. Yet, some background is necessary as well. This hour attempts to take care of both demands by offering an introductory background look at programming and then quickly jumping head-first into hands-on programming. If you went straight into a new programming language with absolutely no background first, you might miss the entire point of programming in the first place. So, the next few pages bring all readers up to a level playing field so that everyone who takes this 24-hour course can begin to write programs right away.
What a Computer Program Does

Most people today have some understanding of a computer’s purpose. For example, a computer can help people balance their books or track their inventory. If you want to begin programming, chances are you have probably been using a computer for some time. Nevertheless, as a future programmer, you should review some fundamental computing concepts before mastering the ins and outs of a new computer language.

At its simplest level, a computer processes data. Figure 1.1 shows this as the primary goal of all computer programs. Many businesses call their computer programming departments data processing departments because computers process data into meaningful information. You may not have considered the difference between the words data and information before, but there is a tremendous difference to a computer professional. Raw data consists of facts and figures, such as hundreds of days of stock prices. A program might process that data into meaningful information, such as a line chart that shows the overall trend of the stock prices over time. It is the computer program that tells the computer what to do. Sometimes, a program might simply process data from another program without showing any output for the user. The processed data is still information because the program’s output, stored on the disk, is changed in some way. For example, a program that closes monthly account balances may collect data from various accounting systems in a company and combine, balance, and close that data, resetting the data for the following month.

A program is a list of detailed instructions that the computer carries out. The program is the driving force behind any job that a computer does. The computer cannot do anything without a program. It is the job of the programmer to design and write programs that direct the computer to take raw data and transform that data into meaningful information for the end-user. The end-user (usually just called the user) of the computer is generally the nontechnical, nonprogramming person who needs the results (the information) that the program provides.

You, as the programmer, are responsible for guiding the computer with the programs you write. Learning to program computers takes a while, but it is certainly rewarding. Computer programming offers the advantage of instant feedback, unlike a lot of other jobs you can train for.
Common Programming Misconceptions

This text aims directly at the heart of the matter: Computers are easy to use and easy to program. A computer is nothing more than a dumb machine that “knows” absolutely nothing. You must supply the program that tells the computer what to do. A computer is like a robot that waits on your every command and acts out your instructions exactly as you give them. Sometimes your program instructions are incorrect. If they are, the computer goes right ahead and attempts them anyway.

CAUTION
Don’t fear computer programming. Computers are tools to help you get your job done. You can learn to program.

Many misconceptions about computers exist and stem from a lack of understanding about how computers work and what computers are physically capable of doing. This book wants to shoot down the myths and improve your understanding of these machines. You’ll be programming computers in no time. The computer is nothing more than a tool that helps you do certain types of work. The computer itself is not bad or good. A hammer is a tool you can use for good (to build houses) or for bad (to break things). A computer in the wrong hands can be used for bad purposes, but that isn’t the computer’s fault any more than it is the hammer’s fault if someone misuses it.

The next three sections attack the three most popular computer myths. Have you heard any of them? Did you think some were true?

Myth 1: Only Math Experts Can Program Computers

Thank goodness this is a myth and not reality—thousands of people would be out of work (including most computer book authors!). Computers would be elitist machines used by only the best engineers and scientists; the casual user could not master them. Computers would still be beneficial in some areas but they would not provide the benefits that so many people can enjoy.

Not only can you be poor at math—you don’t even have to like math or have the desire to learn math to be a good computer programmer. The computer does all the math for you; that’s one of its jobs. There are countless expert computer programmers in the world who cannot tell you the area of a circle or the square root of 64. Relax if you thought this myth was reality.

Programming can provide beneficial side effects. It turns out that, as you become a better programmer, you may find your math skills improving. Developing programming skills tends to improve your overall capability for logical thinking, which underlies many skills in math as well. Therefore, being better in math might be a result of programming but it’s not a prerequisite.
TIP
People who favor logic puzzles, crosswords, anagrams, and word-search games seem to adapt well to programming, but again, liking these gaming activities is not a programming prerequisite. You will find that you can learn to program computers, and actually become extremely good at it, without liking math, being good at math, or having any flair at all for puzzles or word games.

Myth 2: Computers Make Mistakes
You might have heard the adage, “To err is human, but to really foul things up takes a computer!” This might be accurate, but only in that a computer is so very fast that it duplicates a person’s mistakes rapidly.

Computers do not make mistakes—people make mistakes. If you have heard a bank teller tell you that $24 was incorrectly deleted from your savings account because “the computer program made an error,” the teller probably has no idea what really happened. People program computers, people run them, and people enter the data that the computer processes.

The odds of a computer randomly fouling up a customer’s bank balance are minute. Computers simply do not make random mistakes unless they are programmed incorrectly. Computers are finite machines; when given the same input, they always produce the same output. That is, computers always do the same things under the same conditions. Your job, as you learn to program, will be to reduce the chance of computer mistakes.

When a computer malfunctions, it does not make a simple mistake; rather, it really messes things up. When a computer fails, it typically breaks down completely, or a storage device breaks down, or the power goes out. Whatever happens, computers go all out when they have a problem and it is usually very obvious when they have a problem. The good news is that computers rarely have problems.

Before people invented computers, banks kept all their records on ledger cards. When a teller found a mistake (possibly one that the teller had made), do you think the teller said, “The ledger card made a mistake”? Absolutely not. Computers can have mechanical problems, but the likelihood of small mistakes, such as an incorrect balance once in a while, is just too small to consider. Such mistakes are made by the people entering the data or by (gulp) the programmers.

Myth 3: Computers Are Difficult to Program
Computers are getting easier to use, and to program, every day. If you used a microwave, drove a car, or used an iPod recently, then chances are good that you used a computer when you did. Yet, did you know you were using a computer? Probably not. The makers of computers have found ways to integrate computers into your everyday life to monitor and correct problems that might otherwise occur without them.
Of course, if you are reading this book, you want to learn enough about computers to write your own programs. Writing computer programs does take more work than using a microwave oven's computerized timer functions. The work, however, primarily involves getting down to the computer's level and learning what it expects.

Not only are computers getting easier to program every day, but you have more opportunities to learn about them than ever before. Cable television channels are loaded with educational shows about using and programming computers. Books and videos on the subject are all around you. The Internet itself contains scores of classes on all aspects of computers and other topics. There is probably a computer programming class now in session somewhere within 15 minutes of your house as you read this.

**Many Programs Already Exist**

Although there are many programs already written for you to use, sometimes you need a program that fills a specific need and you cannot find one that does exactly what you want. When you are done with this book, you will know exactly what you need to design and write your own programs.

**Programmers Are in Demand**

Look at help-wanted websites. You'll find that there is a shortage of computer programmers. Amidst the requests for Java programmers, C++ programmers, Ruby on Rails programmers, mobile app developers, systems analysts, senior systems analysts, object-oriented programmers, systems programmers, HTML coders, and application programmers, you may find yourself lost in a sea of uncertainty and TLAs (three-letter acronyms) that might, at first, seem hopeless. Do not fret; this book will help direct you toward areas of programming that might be right for you.

Hour 22's lesson, "How Companies Program," explores the chain of computer jobs and describes what each type of programming job is all about. If you are just starting out, you probably won't be able to go to work as the most senior-level programmer, but you will be surprised at the salary your programming skills can bring you.

**The Real Value of Programs**

Although individual computer programs are going down in price, companies and individual computer owners invest more and more in programs every year. Not only do people purchase new programs as they come out, but they update the older versions of programs they already have.
Businesses and individuals must factor in the cost of programs when making computer decisions. Whereas an individual usually buys a computer—called hardware because the machine isn’t changed often or easily—and is done with the hardware purchasing for a while, the purchasing of programs—the software—never seems to end, because software changes rapidly. As a future programmer, this is welcome news because you have a secure career. For the uninformed computer purchaser, the cost of software can be staggering.

A business must also factor in the on-staff and contract programmers and the time needed to write the programs it uses. More information on the programming and support staff appears in the next section.

**Users Generally Don’t Own Programs**

When a company purchases software, it most often purchases a software license. If a company wants to buy a word-processing program for 100 employees, legally it must purchase 100 copies of the program, or at least buy a site license that allows the company to use the software on more than one machine. When a company buys a program, it does not own the program. When you buy a record, you do not own the music; you have only purchased the rights to listen to the music. You cannot legally alter the music, record it, give away recordings of it, and most importantly, you cannot sell recordings that you make of it. The same is true for software that you buy. The license for individual software grants you permission to use the software on one computer at any one time.

**Giving Computers Programs**

Figure 1.2 shows the code for a simple program that creates a web page with an interactive button, all created by a computer programmer. As a matter of fact, that computer programmer is about to be you after you create the code using JavaScript. JavaScript is a programming language that enables you to improve websites by letting you customize the look and feel of your site’s presentation, increasing interaction with your website users, and validating any data they enter. You may think JavaScript is specifically for web masters and site designers, but it’s a simple language and learning its basics can help you learn basic programming concepts.

Another advantage to JavaScript is that most programming languages need you to install an interpreter or compiler on your computer in order to execute your code. With JavaScript (and HTML), you only need a text editor. You type in your code, save it with an .html extension, and then open the saved file with your web browser.
A computer is only a machine. To give a machine instructions, your instructions might be fairly readable, as Figure 1.2’s are, but the code (another name for a program’s instructions) must be fairly rigid and conform to a predefined set of rules and regulations according to the programming language you use. Therefore, to write a program in the JavaScript programming language, you must conform to JavaScript’s rules of proper command spelling and placement. This programming language grammar is called syntax. (And you thought syntax was just a levy placed on cigarettes and liquor!)

Despite its name, JavaScript is not tied to the Java programming language (which is covered in Part III, “Object-Oriented Programming with Java,” of this book). In fact, the name has caused some confusion as computer fans thought that JavaScript was either an extension or spin-off of Java, but it got the name because Java was a hot new language at the time, and some thought the connection of the scripting language to the web development language would benefit the adoption and growth of JavaScript. Misleading name aside, the more you learn JavaScript,
the more easily you will be able to learn additional programming languages. Although computer languages have different goals, syntaxes, and reasons for being, they are often similar in structure.

**NOTE**

It is important to understand that the first section of this book is teaching you the basics of computer programming using the JavaScript language, and is not a complete JavaScript tutorial. This may seem like the same thing, but it is not. The aim of the book is to teach you basic computer programming techniques (with examples in JavaScript code) in the first half of the book and introduce you to a variety of programming languages and jobs in the second half. There are scores of excellent books, including *Teach Yourself JavaScript in 24 Hours*, also by Sams Publishing, that can take you through every in and out of the JavaScript language. That book will take you far more in-depth into the language than this book. Now that we’ve properly calibrated your expectations, let’s get back to beginning programming!

**Source Code**

Even after you make the effort to learn a computer language such as JavaScript, and after you go to the trouble of typing a well-formed and syntactically accurate program such as the one in Figure 1.2, your computer still will not be able to understand the program! The program you write is called **source code**. It is the source code that you write, manipulate, and correct. Your computer, on the other hand, can understand only **machine language**, a compact series of computer-readable instructions that make no sense to people. They make sense to some advanced computer gurus, but my general assertion stands that they don’t make sense to people.

Listing 1.1 shows machine language. Can you decipher any of it? Your computer can. Your computer loves machine language. Actually, it’s the only language your computer understands. And different computers understand their own version of a machine language so what works on one type of computer will not necessarily work on another. It’s best to stay away from machine language and let products such as JavaScript convert your higher-level language into machine language’s cryptic 1’s and 0’s. To convert source code such as your JavaScript program to machine language, you need an interpreter (or for other languages, a **compiler**).

**LISTING 1.1**  Machine language is extremely difficult for people to decipher.

```
01100100
10111101
10010011
10010100
00001111
01010101
11111110
```
All programs you run on your computer, phone, or tablet, including Microsoft Word, Internet Explorer, and programming languages, are already converted into machine language. That’s why you can click a program’s icon and the program begins immediately. No interpretation or compilation is required. By providing you with the machine language only, software vendors serve two purposes:

1. They give you programs that execute quickly without the intervening compiling step.
2. They ensure that you don’t change the source code, thereby protecting their intellectual property.

Your First Program

The first program that you write will be simple. You may not understand much of it, and that’s fine. The goal here is not to explain the program details, but to walk you through the entire process of the steps that are typical of computer programming:

1. Type a program’s source code and save it.
2. Load the code document, which is now an HTML document saved to your hard drive, with your web browser and see what happens.
3. If the page is not doing what you want and has errors, called bugs, you’ll need to fix those bugs in your source code and repeat these steps. Hour 7, “Debugging Tools,” explains how to locate and fix common bugs. You may need to do some bug hunting earlier than Hour 7 if you experience problems with your programs in these first few hours. If a program does not work as described, you will need to compare your source code very carefully to the code in the book and fix any discrepancies before saving it again.

Starting with JavaScript

You can write your JavaScript code in either your text editor or a word processor. If you choose the latter, you must make sure to save your files as text only. Save your programs with an .html extension, and then when you double click on the file, it should automatically open with your default web browser.

NOTE

Unlike word processors, your editor will not wrap the end of a line down to the subsequent line. It’s important that programming instructions (called statements) remain separate and stay on their own lines in many cases. Even if a programming language allows multiple program statements per line, you’ll find that your programs are more readable and easier to debug if you keep your statements one per line. Sometimes, a single statement can get lengthy and you may span long statements over two or three lines, but rarely would you want multiple statements to appear on the same line.
Clarifying Comments

The program presented in Figure 1.2 not only has program statements, but also some lines that do not do anything when the program is run by the interpreter, but have great value to anyone reading your code. These lines start with either a slash and asterisk (/*) or two consecutive slashes (//) and are known as comments.

You may be asking if there’s any difference between the slash and asterisk comment and the double-slash comments. Taking the second type first, the double-slash comment is known as a single-line comment. JavaScript will ignore all text to the right of the two slashes. As soon as you hit return, JavaScript will start paying attention to your code again. So if you want to keep commenting, you will need to start the next line with a fresh pair of slashes.

// The next five lines ask web visitors for their favorite
// color and Steinbeck novel

In the previous example, if you left off the two slashes, JavaScript would assume you were either typing color or calling a function named color and you might not get the results you were expecting.

When JavaScript encounters a comment that begins with the slash and asterisk, it will treat everything that follows as a comment until it sees a closing asterisk and slash (*/). This can be on the same line:

/* Next, the code calculates the area of the circle. */

Or it can go on for several lines:

/* This section is used to enter a contact
The user can enter a name, physical address,
up to 5 email addresses, home, work, and cell
phone numbers, and any additional comments
to help remember how they know the person. */

All languages support comments, which are also called remarks, although some languages use an apostrophe instead of the slashes. Although you’ll find many comments in programs, and you should put many comments in the programs you write, comments are not for the interpreter or for the computer. Comments are for people. In JavaScript, for example, the interpreter ignores all text to the right of the slashes or slash/asterisk. The computer knows that comments are for you and not for it.

Reasons for Comments

Comments help produce more understandable program listings. The programming industry considers the inclusion of comments to be one of the most important good programming habits you can develop. Here are common reasons programmers put comments in their programs:
- Comments can identify programmers—If you develop programs with others or write programs for a company, you should put your name and contact information in comments at the top of your programs. Later, if someone has a question about the program or wants you to change the program, they’ll know that you wrote the program and they can locate you.

- Before a section of tricky code, you can use comments to include your explanation of the code’s purpose. Later, if you or someone else needs to change the program, the change will be easier; reading the comments is much simpler than trying to figure out the goals of the tricky code from uncommented code.

**Placement of Comments**

When you use the double slashes (//), a comment can appear on the same line as code. The following code shows comments to the right of lines of other program instructions. Notice that double slashes precede the text. These comments explain what each line of code does.

```plaintext
if hours > 12          // Test the hour for AM/PM indicator
  hours = hours - 12  // Convert 24-hour time to 12-hour
  amOrPm$ = " PM"     // Set indicator that will print
```

**TIP**

Most programming languages are free-form, meaning that you can indent lines and add as many spaces and blank lines as you like. Doing so makes a program more readable. In Hour 7, you’ll learn how to use this whitespace (extra spaces and lines in code) to improve your programs’ readability by separating one program section from another and, thereby, making it easier for programmers to understand your code.

Sure, as a programmer, you will often understand code without the need to put a remark to the right of each line. But at the time you write a program, the program is clearest to you. When you later go back to edit a program, what may have been clear before is no longer clear because time has passed. A few comments will go a long way toward reminding you of your original intent.

However, do not overdo comments. Some comments are redundant, as the following shows:

```plaintext
document.write("Martha");   // Prints the word Martha
```

Such a comment is not helpful and serves only to cloud an already-understandable instruction. Don’t overuse comments, but use them generously when needed to clarify your code.
Entering Your Own Program

You’re now ready to type and run your own program from scratch. The program will be simple. The goal here is to get you used to using the editor.

Follow these steps to create your own program:

1. In your text editor, select File and then New File to create a brand-new document.

2. Type the program in Listing 1.2 into the editor. Be careful that you type the program exactly as you see it in the listing. If you make a typing mistake, you will not be able to run the program. The program is a little bit complicated, but if you type it as is, you should have a working program. Notice that both styles of comments are used in the program.

Listing 1.2  Type your first program into the editor.

```html
<!DOCTYPE html>
<html>
<head>
<script>
/* This is the function that gets
called when the user clicks the
button on the main page */

function displayAnswer() {
    document.write("Just 24 1-hour lessons!");
}
</script>
</head>
<body>
<h1>My First Program</h1>
<p id="demo">How long will it take for me to learn to program?</p>
<button type="button" onclick="displayAnswer()">How many hours?</button>
</body>
</html>
```
3. Save the program as text only, and give it a name with an .html ending. I named mine TYBeginProgram1.html. After you save the program, either double click the file’s icon in your explorer or open the file in your web browser. You should see the web page shown in Figure 1.3 to start and in Figure 1.4 when you click the button.

NOTE
The page you load into your web browser should have an .html extension (.htm works as well). Eventually, you will want to separate your JavaScript code into its own files, using .js files. For now, you can embed all your code into .html files.

FIGURE 1.3
With less than two dozen lines of code, you’ve created a simple web page, complete with a button for the visitor to click.
Once you click the button on your web page, you get a new page with more information. Awesome interactivity, right?

If your program contained an error, the page would not show up like the ones in Figures 1.3 and 1.4. You would have to work through your version line-by-line to see where the error might be.

**TIP**
In addition to checking for errors, it is also a smart idea to check your created pages in different browsers when possible to see the subtle difference in presentation from browser to browser.

**Summary**

Now that you know a bit more about the computer industry as a whole, you have a better idea of where programmers fit into the picture. Computers can do nothing without programs and programmers are responsible for providing those programs.

You already got a taste of programming this hour by typing your first JavaScript program and opening the resulting document in your browser. As you saw by typing your own program into an editor, entering and running a code is not painful at all. As a matter of fact, the quick feedback that programming offers is a rewarding experience. Programmers often like their profession very much because they can see the results of their work quickly.
Q&A

Q. Once I write a program, is my job finished?

A. It is said that a program is written once and modified many times. Throughout this 24-hour tutorial, you will learn the importance of maintaining a program after it is written. As businesses change and computing environments improve, the programs that people use must improve as well. Companies provide upgrades to software as new features are needed. The programs that you write today will have to change in the future to keep up with the competition as well as with new features that your users require.

Q. Can I enter other programming language listings in my editor as well?

A. You can write programs in other programming language, such as C, C++, and Java, in your text editor. However, you would need a compiler or interpreter (depending on the language) to convert the programming language’s code to machine language. A machine language program is your program’s instructions compiled to a language your computer can understand.

Workshop

The quiz questions are provided for your further understanding.

Quiz

1. What is the difference between data and information?
2. What is a program?
3. What is a programming language?
4. True or false: Computers never make mistakes.
5. Why should people not fear the computer replacing their jobs?
6. What do programmers use editors for?
7. True or false: Java and JavaScript are related to each other.
8. What filename extension should you use when you create a JavaScript program?
9. True or False: There’s no problem writing your programs in your typical word-processing program.
10. Tweak the program in Listing 1.2 so the dialog box asks, “Who wrote this program?” and the answer the user gets upon clicking the button is your name.

Answers

1. Data consists of raw facts and figures, and information is processed data that has more meaning.
2. A program is a set of detailed instructions that tells the computer what to do.
3. A programming language is a set of commands and grammar rules with which you write programs that give computers instructions.

4. False. A computer might make a mistake, but it’s rare that it does so. In the vast majority of cases where a computer is blamed, a person entered bad data or typed a bad program.

5. Computers increase jobs, not replace them. The information industry is one of the reasons for the past two decades of economic growth.

6. Programmers use editors to type programs into the computer.

7. False. JavaScript and Java only share a name; JavaScript was a scripting language that was developed at the same time Java was hot, so it was named in a way to take advantage of Java’s popularity.

8. If your code is going to stay in your HTML document, you can use .html or .htm, but when you create separate JavaScript files, use .js.

9. False. The default method of saving files in word processors will add formatting codes that will generate errors when you try to run your program. You can still use a word processor, but you must always remember to save your files as plain text.

10. Here is one possible solution:

```html
<!DOCTYPE html>
<html>
<head>
<script>
/* This is the function that gets called when the user clicks the button on the main page */

function displayProgrammer()
{
    document.write("Dean Miller did!");
}
</script>
</head>
<body>

<h1>My First Program</h1>
<p id="demo">Who wrote this program?.</p>

<button type="button" onclick="displayProgrammer()">And the answer is...</button>

</body>
</html>
```
Symbols

#include statement, 336-337
<h> tag, 278

A

absolute values
    arguments, 90
abstraction, objects, 346
accessing
    cookies via document.cookie, 189-190
    global variables, 324
    properties within a method, 327-328
accumulators, 146-148
    totals, 152-153
accuracy, debugging code, 114-119
ActiveX, 289

advanced programming, 187, 198
    cookies
        deleting, 193-198
        reading, 192-193
        writing, 190-191
    JavaScript programming
        weakness, 187-191
    AJAX (Asynchronous JavaScript and XML), 210, 285-289, 297
        ajaxRequest function, 292
        ajaxResponse function, 292
        frameworks, 288
        JavaScript client, 286
        libraries, 288
        creating, 291-292
        creating quizzes from, 292-296
        limitations, 288-289
        popular examples of, 287-288
        server-side script, 286-287
algorithms, 169
    accumulators, 146-148
    totals, 152-153
array variables, 149-150
    counters, 146-148
    nested loops, 169
    optimizing functions, 165-169
    searching arrays, 158-162
    sorting data, 155-158
    swapping values, 154-155
Amazon.com, 287
ambiguity, programming, 33-34
anchor tag (HTML), 281-282
Andromeda Strain, 114
animations
    Java applets, 205-206
API (application programming interface), 209
applets (Java), 203, 255, 265.
See also programs
    animations, 205-206
    appletviewer, 256
    Autocrazy, 212
    finishing, 257-261
    placing in web pages, 261-264
applets (Java)

security issues, 211
skeleton, 256-257
viewing inside web page, 264-265
writing, 255-261
appletviewer, 256
application programming interface (API), 209
applications. See also applets (Java); programs
  compilation and packaging, 408-409
  continuous integration and deployment, 416
distributing, 407, 410
  open source, distributing, 409
  RAD (rapid application development), 47
  version control, 410
  Visual Basic, creating from scratch, 361-368
arguments
  absolute values, 90
arithmetic operators
  Java, 228-229
  PHP, 313-314
Array data type (PHP), 312
array variables, 149-150
arrays, 87-88, 226
  elements, 87
  parallel, 150
  reserving space, 149-150
searching
  algorithms, 158-162
  binary searches, 161-162
  sequential searches, 159
ascending order, sorting data, 155
ASCII (American Standard Code for Information Interchange), 76
  nonprinting characters, 87
tables, 76-79, 86-87
assembly language, 334
assigning values, 61-63
assignment operators, PHP, 313
Asynchronous JavaScript and XML (AJAX). See AJAX (Asynchronous JavaScript and XML)
Autocrazy, 212
automated testing, code, 415-416

B
Backbone.js, 288
bandwidth
  web pages, 257
behavior, objects, adding to, 350-352
beta testing programs, 141
binary searches
  arrays, 161-162
bits, 76
Boolean data type (PHP), 312
Boolean literals, 222
Boolean variables, 225
bottom-up design, 48
branching programs, 133
breakpoints, setting, 127
bubble sort, 156-158
bugs
debugging tools, 113
  first ever, 113-114
logic errors, 115
runtime errors, 119
syntax errors, 115-118
built-in functions
  C programming language, 339-343
  PHP, 320-321
bytecode, 206-208
bytes, 76

C
C programming language, 333-336, 354-355
  #include statement, 336-337
  versus C++, 347-348
  command keywords, 334
  conversion characters, 340
  data formats, 337-338
  functions, 338
    built-in, 339-343
    main(), 336
    printf(), 339-341
    scanf(), 341-343
    writing general program, 343-345
  operators, 345
  requirements, 335
  statements, control, 345-346
  strings, 342
C# programming language, 375
  .NET Framework, 3v78-387
  Visual C#, 383-387
C++ programming language, 333, 346, 354-355
  versus C, 347-348
  I/O, 347-348
objects, 348-353
accessing members, 350
adding behavior to, 350-352
declaring variables, 349-350
working with class scope, 352-353
OOP (object-oriented programming), 346-347
requirements, 335
versus Java, 202
calculations, JavaScript, 79-83, 92-93
call stacks, 127
call-by-value languages, 249
calling functions, 321
capturing mouse events, 66-71
CASE (Computer-Aided Software Engineering), 417-418
case sensitivity, Java programs, 214
certification, 395-396
CGI (Common Gateway Interface), 206
character literals, 222
character strings, 61
character variables, 225-226
chargeback method, 392-393
charts, flowcharts, 51, 97, 133
check boxes, Java interface, 209
Chrome, debugging tools, 120
CIL (common intermediate language), CLR (common language runtime), 377
classes
Java, 239, 246-248, 251
inheritance, 247
methods, 249-251
.NET Framework, FCL (framework class library), 377-378
objects, 346
working with scope, 352-353
public, 215-216
CLR (common language runtime)
CIL (common intermediate language), 377
CTS (common type system), 377
.NET Framework, 376-377
code. See also programs
bugs
first ever, 113-114
logic errors, 115
runtime errors, 119
syntax errors, 115-118
bytecode, 206-208
comments, 14-15
placement, 15
reasons for, 14-15
debugging, 113, 120-129
Java, analyzing, 215-217
JavaScript, packaging into functions, 137-139
leading spaces, 314
printing, for loop, 102-105
profiling, 142
pseudocode, 51-52, 133
source, 12-13
spaghetti, 133
strings, 75-79v
ASCII table, 76-79
merging, 76-77
typing first program, 16-18
Visual Basic
finalizing, 366-368
modules, 360
whitespace, 15, 119-120
writing, 53, 119-120
JavaScript, 13

code listings
Accessing Class Properties within a Method, 327-328
Accessing Global Variables with the global statement, 324
Adding a different interactivity to your web photos with the onmouseover event, 179-180
Adding a photo rotator to your webpage is easy with JavaScript, 174-175
Adding interactivity to your web photos, 177
AJAX Library, 291
Alert sends output of calculations and other data to the screen, 62-63
Binary search can speed searching tremendously, 163-162
Building a simple informational function, 137-138
C is cryptic and requires several elements that other languages do not, 335
C program contains no errors but is extremely difficult to maintain, 119
C program with better spacing is far clearer to understand, 120
Class can have both data members and procedure methods, 246
Class with a Method, 327
Collecting your functions into a .js file, 139-140
Completed version of the Calculator program, 243
Completing the text-based Java applet, 258
Counting down from 10 to 1 is simple with a for loop, 106
Declaring a Function That Requires an Argument, 322
Declaring and Calling a Function, 321
Even a simple program can have an error in it, 116
Examples of setting your own styles using CSS to control font color, size, and typeface, 279-280
Examples of the <h> tag show how to change headline text size, 277-278
Function That Returns a Value, 322
General format for all HTML Web pages, 275-276
HTML File for the Quiz Example, 293
HTML file that carries a Java applet, 261-263
JavaScript calculations follow the order of operator precedence, 81-82
JavaScript File for the Quiz Example, 295-296
JavaScript script that performs a button change when the user points to the button, 71
Machine language is extremely difficult for people to decipher, 12
Merging two strings together is simple, 76
Move away from the text mode into windowed features, 244
NetBeans builds an outline of your code so you have less to type, 241-242
Number-guessing game can use a counter variable, 147-148
Our website for our small store is starting to get a bit robust, 181-182
PHP Script Incorporated into HTML, 304
Printing a message several times in a loop is efficient, 103-104
Printing results of calculations, 59
Printing without a for loop gets tedious, 102-103
Program outline that doesn’t use functions can be hard to maintain, 165-166
Program outline that uses functions is simple, 166-167
Program that uses functions for everything is easier to maintain and understand, 167-168
Proof That Your Object Exists, 326
Sales program stores data in arrays for each month, 150-151
Sales-reporting and averaging program can use an accumulator for sales totals, 152-153
scanf() and printf() for input and output, 343-344
Sequential search can help an inventory application, 161
Showing Object Properties, 327
Simple C# program to calculate the area of a circle when given the radius, 380-381
Simple HTML commands can produce attractive Web pages, 274-275
Simple Java program adds two numbers together, 240
Simple PHP Script, 303
storeinfo2.html, 139-140
Swing-based Java applet uses this program outline, 256-257
Teacher’s grade-printing program is easy to follow when you use a loop, 104-105
Type this to complete the main procedure that Forte began for you, 243
Type your first program into the editor, 16-17
Use an if statement when you want your program to make a decision, 96-97
Use for to control a counting loop, 101
Use main() to control the rest of the program, 343-345
Using if-else to compute an employee’s overtime, 232-233
Using prompt to get the value of a circle’s radius, 64
Using prompts to get a user’s first and last names, 66
Using the for Statement, 310
Variables Defined Outside Functions Are Inaccessible from Within a Function by Default, 323
while Statement, 308
XML File Containing the Quiz Questions and Answers, 294
You can control the grade program with a while loop, 107-108
You can nest several if statements to check a series of conditions, 99-100
You can sort a list of values with the bubble sort, 156-157
You can use prompt to simulate a cash register program for a small store, 67-68
You must supply detailed instructions when you want the computer to do something, 27-28
Your first Java program may look cryptic, 213-214
college degrees, 395
combined assignment operators, PHP, 314-315
command keywords, C programming language, 334
command tags (HTML), See tags (HTML)
comments, 14-15
double slashes (//), 15
Java code, 214
JavaScript, 14
PHP scripts, 305
placement, 15
reasons for, 14-15
Common Gateway Interface (CGI), 206
common intermediate language (CIL), CLR (common language runtime), 377
common language runtime (CLR), See CLR (common language runtime)
common type system (CTS), CLR (common language runtime), 377
companies
chargeback method, 392-393
computer-related jobs, 393-394
data processing departments, 389-393
IT (information technology) departments, 389-393
programmers, job titles, 394-400
programming, 389, 404
consultation, 403-404
ongoing training needs, 419-420
putting program into production, 401-403
structured walkthrough, 400-401
comparison operators
Java, 229
PHP, 316
compilation, applications, 408-409
compilers, 19
JIT (just-in-time compiler), 377
computer programs. See programs
Computer-Aided Software Engineering (CASE), 417-418
computers
mistakes, 8
programming languages, 33
concatenation operators, PHP, 314
conditional operators, Java, 230
constants, PHP, 319
constructs, 133

How can we make this index more useful? Email us at indexes@samspublishing.com
consultants, 403-404
context, devices, 260
continuous integration and deployment of applications, 416
contract programmers, 132, 393
control statement, C programming language, 345-346
controlling programs, 95, 109-111
comparing data with if statement, 95-98
looping statements, 96-109
relational tests, writing, 96-100
conversion characters, C programming language, 340
cookieName property (cookies), 191
cookies, 188-191
accessing, via document.
cookie, 189-190
deleting, 193-198
properties, 190-191
reading, 192-193
setting and reading, PHP, 329
writing, 190-191
cookieValue property (cookies), 191
counters, 146-148
counting loops, controlling, 101
Crichton, Michael, 114
CSS (cascading style sheets), 282
CSS3, 277-281
CTS (common type system), CLR (common language runtime), 377
data
comparing, if statements, 95-98
hiding, 211
keyboard, obtaining, 62-71
sorting, 59-61
data entry clerks, 396-397
data flows, defining, 43-50
data formats, C programming language, 337-338
data modeling, UML (Unified Modeling Language), 418
data processing, 6, 75
ASCII table, 86-87
methods, 87-91
performing math with JavaScript, 79-83
strings, 75-79
data processing departments, 6, 389-393
data types, PHP, 312-313
databases, interacting with, PHP, 329
debugging code, 113, 126-129
accuracy, 114-119
JavaScript, 120-124
JavaScript console log, 125-126
decision structured-programming construct, 134-135
decision symbols, 97
declaring functions, 321-322
decrement operators, Java, 227-228
defining PHP functions, 321-323
degrees (college), 395
deleting cookies, 193-198
deprecation, 279
descending order, sorting data, 155
designing programs, 39, 41-42
benefits, 39-40
bottom-up design, 48
defining output and data flows, 43-50
logic, 51-53
OOD (object-oriented design), 44
prototyping, 45-47
RAD (rapid application development), 47
top-down, 48-50
user-program agreements, 40-41
desk checking programs, 141
devices, context, 260
digital signatures, 211
distributing applications, 407, 410
issues, 407-409
DLR (dynamic language runtime), .NET Framework, 378
document.cookie, accessing cookies, 189-190
DOM (document object model), 286
domain property (cookies), 191
Donahue, John, 204
double slashes (/), comments, 15
do.while loop, 107-109
drawString() function, 215-217
drawString() method, 257
dynamic language runtime (DLR), "NET Framework, 378

E

EBCDIC, tables, 78
echo statement, 304
elements, arrays, 87
Ember.js, 288
empty strings, 64
encryption, 211
environment, IDEs (Integrated Development Environments), 415
errors. See bugs
escape sequences, Java, 222-223
events, mouse, capturing, 66-71
executable content, Java, 204-206
expires property (cookies), 191
extraction operators, 347

F

Facebook, 287
FCL (framework class library), "NET Framework, 377-378
fields, 42
Firefox, debugging tools, 120
first program, typing, 16-18
Float data type (PHP), 312
floating-point numbers, 82
floating-point variables, 225
flow control functions, PHP, 306-308
flowcharts, 51, 97, 133
for loop, 101-105, 234-235
controlling, 105-107
printing, 102-105
form data, processing, PHP, 328
format, Java statements, 230
Forte for Java, 217
FORTRAN programming language, 34
framework class library (FCL), "NET Framework, 377-378
frameworks, AJAX (Asynchronous JavaScript and XML), 288
functions
ajaxRequest, 292
ajaxResponse, 292
C programming language, 338
built-in, 339-343
main(), 336
printf(), 339-341
scanf(), 341-343
writing general program, 343-345
case sensitivity, 322
collecting into .js file, 140
creating, 137-139
drawString(), 215-217
JavaScript code, packaging into, 137-139
versus methods, 216
optimizing, 165-169
PHP, 320
built-in, 320-321
calling, 321
declaring, 321-322
defining, 321-323
returning values from user-defined, 322-323
variable scope, 323-324
printBR(), 322
user-defined, 321
returning values from, 322-323

g

garbage collection, CLR (common language runtime), 377
Git, 410
GitHub, 410
global variables, 224
Gmail (Google), 287
goto statement, 133
graphical user interface (GUI)
Java programs, 244-245
graphics. See images
GUI (graphical user interface)
Java programs, 244-245
Gutmans, Andi, 301

H

<h> tag, 278
hiding data, 211
Hopper, Grace, 114
horizontal scrolling, websites, 437
Hörnell, Karl, 212
HTML (Hypertext Markup Language), 34, 269, 275-276, 282
combining with PHP, 304-305
CSS3, 277-281
hyperlinks, 281-282
Java, executable content, 204-206
programming, 269-274
standardization, 271
tags
  <h>, 278
anchor, 281-282
command, 273-275
text formatting, 276-278
web pages
displaying, 273
including graphics in, 281-282
hyperlinks, 281-282
Hyperlink Markup Language (HTML). See HTML (Hypertext Markup Language)

IDEs (Integrated Development Environments), 415
if statements, 231-233
  comparing data, 95-98
  nesting, 100
  relational operators, 99
if-else statements, 231-233
images
  rotating, 173-179
  web pages, including in, 281-282
increment operators, Java, 227-228
inheritance
  Java classes, 247
  objects, 346
init() method, 216
initialization, Java variables, 224
input, 57, 71-73
inputting, strings, 64-68
insertion operators, 347
instructions, programs, 25-34
JavaScript, 27-28
Integer data type (PHP), 312
integer literals, 222
integer variables, 224-225
  PHP, automatically incrementing and decrementing, 315-316
Integration Development Environments (IDEs), 415
interfaces
  API (application programming interface), 209
  CGI (Common Gateway Interface), 206
  Java, 209-211
Internet-related jobs, 399-400
interpreters, 19
I/O (input/output), C++
programmng language, 347-348
IT (information technology)
departments, 389-393
Java, 12, 34, 201-204, 212-218, 221, 230, 235-236
applets, 203, 255, 265
  animations, 205-206
  appletviewer, 256
  finishing, 257-261
  placing in web pages, 261-264
  skeleton, 256-257
  viewing inside web page, 264-265
  writing, 255-261
versus C++, 202
call-by-value language, 249
case sensitivity, 214
classes, 239, 246-248, 251
  inheritance, 247
  methods, 249-251
code
  analyzing, 215-217
  comments, 214
development of, 202
  example program, 213-214
  executable content, 204-206
  Forte for Java, 217
  interface, 209-211
Java Virtual Machine (JVM), 207-208
  versus JavaScript, 210
JVM (Java Virtual Machine), 216-217
  literals, 222-223
loops
  for, 234-235
  while, 233-234
multi-platform executable content, 206-208
NetBeans, 217
OOP (object-oriented programming), 245-246, 248-249
coding, 262
operators, 227-230
  arithmetic, 228-229
  comparison, 229
  conditional, 230
  decrement, 227-228
  increment, 227-228
  primary math, 227
programs, GUI (graphical user interface), 244-245
running programs, NetBeans, 239-244
seamless execution, 206
security issues, 211
standalone, 209-210
statements
  format, 230
  if, 231-233
  if-else, 231-233
Swing, 209
usage, 208-209
variables, 223-226
  arrays, 226
  Boolean, 225
  character, 225-226
  floating-point, 225
  initialization, 224
  integer, 224-225
  string, 226
  web pages, 255
Java Virtual Machine (JVM), 207-208
JavaScript, 10-13, 34, 173, 183-184, 285, 378. See also
AJAX (Asynchronous JavaScript and XML)
  assigning values, 61-63
  calculations, 79-83, 92-93
  capturing mouse position, 179
  comments, 14
  debugging code, 120-124
  console log, 125-126
  versus Java, 210
jQuery, 288
  math operators, 79
  packaging code into functions, 137-139
  printing screen, 57-58
  programming weakness, 187-191
  programs, instructions, 27-28
  Prototype, 288
  repeating news tickers, adding to websites, 179-183
  rotating images on page, 173-179
  scriptlets, 66
  sorting data, 59-61
  writing code, 13
  XMLHttpRequest object, 289-291

JIT (just-in-time compiler), 377
job titles, programmers, 394-400
jQuery JavaScript library, 288
junior programmers, 397
JVM (Java Virtual Machine), 207-208

K-L
keyboard data, obtaining, prompt method, 62-71
keywords, C programming language, 334
languages. See programming languages
LANs (Local Area Networks), 399
leading spaces, 314
Lerdorf, Rasmus, 301
libraries
  AJAX (Asynchronous JavaScript and XML), 288
  creating, 291-292
  creating quizzes from, 292-296
  .NET Framework, FCL (framework class library), 377-378
licenses, software, 10
literals, Java, 222-223
Local Area Networks (LANs), 399
local variables, 323
logic, developing, 51-53
logic errors, 115
logical operators, PHP, 316-318
looping
  statements, 96-109
  structured-programming construct, 135-137
loops
  counting, controlling, 101
  do.while, 107-109
  for, 101-105
    controlling, 105-107
  Java
    for, 234-235
    while, 233-234
  nested, 169
  nesting, 107
  PHP, 308-310
    while, 107-109
lowercase strings, 88
Lunar Lander, 204

M
machine language, 12
main() function, 336
mainframe computers, 78
maintaining programs, 19
Management Information Systems (MIS), 416
math
  computer performance, 81-86
    performing with JavaScript, 79-83
  math operators
    Java, 227
    JavaScript, 79
Math.abs() method, 90
media queries, CSS (cascading style sheets), adding to web pages, 432
members, accessing objects, 350
merging strings, 76-77
messages, objects, 346
methods
  data processing, 87-91
  drawString(), 257
  versus functions, 216
  init(), 216
  Java classes, 249-251
  Math.abs(), 90
  numeric functions, 89-91
  open(), 289-290
  paint(), 216-217, 259
  PHP objects, 327-328
  prompt, obtaining keyboard data, 62-71
  resize(), 258
  send(), 290
  setColor(), 260-262
  string, 88-89
MIS (Management Information Systems), 416
mistakes, computers, 8
mobile web applications
  distribution, 409
modules, code, 360
mouse
  capturing position, 179
  capturing events, 66-71
N
names, sorting, 155
naming, PHP variables, 310-312
nested loops, 169
nesting
  if statements, 100
  loops, 107
.NET, 375-376
NetBeans, 217
  Java programs, running, 239-244
.NET Framework, 375-376, 387
C# programming language, 378-387
CLR (common language runtime), 376-377
DLR (dynamic language runtime), 378
FCL (framework class library), 377-378
parallel computing platform, 378
network-related jobs, 399-400
networks
  LANs (Local Area Networks), 399
  WANs (Wide Area Networks), 399
New Project selection screen (Visual Basic), 360
news tickers, websites, adding to, 179-183
nonprinting characters, ASCII codes, 87
NULL data type (PHP), 312
numbers, binary equivalents, 85
numeric functions, 89-91

O
Object data type (PHP), 312
object-oriented programming (OOP). See OOP (object-oriented programming)
objects, 347
abstraction, 346
behavior, adding to, 350-352
C++, 348-353
classes, 346
working with scope, 352-353
DOM (document object model), 286
inheritance, 346
members, accessing, 350
messages, 346
PHP, 324-325
creating, 325-326
methods, 327-328
properties, 326-327
polymorphism, 347
reuse, 347
variables, declaring, 349-350
XMLHttpRequest, 289-291
OOD (object-oriented design), 44
OOP (object-oriented programming), 44
abstraction, 346

C++ programming language, 346-347
Java, 245-246, 248-249
coding, 262
open() method, 289-290
open source software, distributing, 409
operators
C programming language, 345
extraction, 347
insertion, 347
Java, 227-230
arithmetic, 228-229
comparison, 229
conditional, 230
decrement, 227-228
increment, 227-228
primary math, 227
overloading, 354-355
PHP, 313-319
arithmetic, 313-314
assignment, 313
combined assignment, 314-315
comparison, 316
concatenation, 314
logical, 316-318
precedence, 318-319
relational, if statement, 99
output, 57, 71-73
defining, 43-50
keyboard data, obtaining, 62-71
printing screen, 57-58
programs, 42
sorting data, 59-61
overloading operators, 354-355
ownership, programs, 10

P
packaging applications, 408-409
paint() method, 216-217, 259
parallel arrays, 150
parallel computing platform, .NET Framework, 378
parallel testing programs, 141-142
parameters, programs, 42
path property (cookies), 191
PHP (PHP: Hypertext Preprocessor), 34, 301, 329
combining with HTML, 304-305
comments, 305
common uses, 328-329
constants, 319
data types, 312-313
echo statement, 304
flow control functions, 306-308
functions, 320
built-in, 320-321
defining, 321-323
returning values from user-defined, 322-323
variable scope, 323-324
loops, 308-310
objects, 324-325
creating, 325-326
methods, 327-328
properties, 326-327

How can we make this index more useful? Email us at indexes@samspublishing.com
operators, 313-319
arithmetic, 313-314
assignment, 313
combined assignment, 314-315
comparison, 316
concatenation, 314
logical, 316-318
precedence, 318-319
print() statement, 304
requirements, 302-303
structures, 303-308
variables, 310-312
integer, 315-316
placement, comments, 15
polymorphism, objects, 347
practicing debugging, 120-124
precedence, operators, PHP, 318-319
preprocessor directives, 336-337
primary math operators,
Java, 227
print() statement, 304
printBR() function, 322
printf() function, 339-341, 344
printing
for loop, 102-105
screen, JavaScript, 57-58
private keys, 211
procedures, Visual Basic, 369-370
profilers, 413-414
profiling code, 142
programmers,
contract, 132, 393
demand for, 9
job security, 402
job titles, 394-400
ongoing training needs, 419-420
programming, 5, 142-143
advanced, 187, 198
deleting cookies, 193-198
JavaScript programming
weakness, 187-191
reading cookies, 192-193
writing cookies, 190-191
algorithms, 145-146, 169
accumulators, 146-148, 152-153
array variables, 149-150
counters, 146-148
nested loops, 169
optimizing functions, 165-169
searching arrays, 158-162
sorting data, 155-158
swapping values, 154-155
ambiguity, 33-34
backlogs, 131-132
companies, 389, 404
consultation, 403-404
putting program into
production, 401-403
structured walkthrough, 400-401
difficulty, 8-9
first program, 13
future trends, 413-421
HTML (Hypertext Markup
Language), 269-274
math, 7-8
misconceptions, 7-9
OOP (object-oriented
programming), 44
profilers, 413-414
resource editors, 414
structured, 131-137
decision
structured-programming
construct, 134-135
looping
structured-programming
construct, 135-137
origins, 132
sequence
structured-programming
construct, 133-134
programming languages, 33-34.
See also AJAX (Asynchronous
JavaScript and XML); C
programming language; Java;
JavaScript; PHP; Visual Basic
AJAX (Asynchronous JavaScript
and XML), 285-289
assembly, 334
C, 333-336, 354-355
requirements, 335
C#, 375
C++, 333, 346, 354-355
requirements, 335
call-by-value, 249
compilers, 19
FORTRAN, 34
goto statement, 133
interpreters, 19
Java, 34, 201-204, 212-218,
221, 230, 235-236
analyzing code, 215-217
applets, 203
versus C++, 202
development of, 202
dependent program, 213-214
interface, 209-211
literals, 222-223
multi-platform executable content, 206-208
NetBeans, 217
OOP (object-oriented programming), 245-246, 248-249
operators, 227-230
seamless execution, 206
security issues, 211
standalone, 209-210
usage, 208-209
variables, 223-226
JavaScript, 10-13, 34, 285
writing code, 13
PHP, 34, 301, 329
requirements, 302-303
Visual Basic, 333, 359, 370-371

programs, 95. See also applets (Java); applications; code
branching, 133
comparing data with if statement, 95-98
controlling, 109-111
looping statements, 96-109
debugging, 126-129
designing, 39, 41-42
benefits, 39-40
bottom-up design, 48
defining output and data flows, 43-50
logic, 51-53
OOD (object-oriented design), 44
prototyping, 45-47
RAD (rapid application development), 47
top-down, 48-50
user-program agreements, 40-41
existing, 9
fields, 42
first, typing, 16-18
instructions, 25-34
JavaScript, 27-28
Java
case sensitivity, 214
running with NetBeans, 239-244
maintaining, 19
need for, 21-24
obtaining, 22
output, 42
obtaining keyboard data, 62-71
printing, 57-58
sorting data, 59-61
ownership, 10
parameters, 42
putting into production, 401-403
relational tests, writing, 96-100
saved instructions, 29-31
setting breakpoints, 127
software licenses, 10
source code, 12-13

tasks, 6
testing, 139-142
beta, 141
desk checking, 141
parallel, 141-142
ubiquity, 24-25
value, 9-10
writing, 119-120
prompt method, keyboard data, obtaining, 62-71
properties
cookies, 190-191
PHP objects, 326-327
Prototype JavaScript library, 288
prototyping, 45-47
pseudocode, 51-52, 133
public class, 215-216
purchasing programs, 22
Python, 378

Q-R

QA (quality assurance) testing, 415-416
RAD (rapid application development), 47
reading cookies, 192-193
relational operators, if statement, 99
relational tests, writing, 96-100
remarks. See comments
repeating news tickers, websites, adding to, 179-183
reserving space, arrays, 149-150
resize() method, 258
Resource data type (PHP), 312

How can we make this index more useful? Email us at indexes@samspublishing.com
resource editors, 414
recompute, objects, 347
rollover effect, web pages, 66
rotating images on page, 173-179
Ruby, 378
runtime errors, 119
sales totals, accumulators, 152-153
 saved instructions, programs, 29-31
scanf() function, 341-343
scope, classes, working with, 352-353
screen, printing, JavaScript, 57-58
scriptlets (JavaScript), 66
scripts
JavaScript, scriptlets, 66
PHP
 comments, 305
Simple PHP Script, 303
 structures, 303-308
searching arrays
 algorithms, 158-162
 binary searches, 161-162
Secure property (cookies), 191
security, Java, 211
send() method, 290
senior programmers, 397
sequence structured-programming construct, 133-134
sequential searches, arrays, 159
setColor() method, 260-262
skeleton, Java applet, 256-257
slashes, comments, 15
software. See applications
sorting data, 59-61
 algorithms, 155-158
 bubble sort, 156-158
source code, 12-13
space, arrays, reserving, 149-150
spaghetti code, 133
standalone Java, 209-210
statements
 #include, 336-337
 C programming language,
 control, 345-346
 echo, 304
 goto, 133
 if
 comparing data, 95-98
 Java, 231-233
 nesting, 100
 relational operators, 99
Java
 format, 230
 if, 231-233
 if-else, 231-233
 looping, 96-109
 print(), 304
String data type (PHP), 312
string methods, 88-89
string variables, 226
strings, 75-79
ASCII table, 76-79
C programming language, 342
character, 61
empty, 64
inputting, 64-68
making all capital or
 lowercase, 88
merging, 76-77
replacing parts, 88-89
structured programming, 131-137
constructs, 133
decision
 structured-programming construct, 134-135
looping
 structured-programming construct, 135-137
origins, 132
sequence
 structured-programming construct, 133-134
subroutines. See functions
Subversion, 410
Suraski, Zeev, 301
swapping values, variables, 154-155
Swing, 209, 256-257
syntax errors, 115-118
systems analysts, 398-399

 Tables, ASCII, 76-79
tags (HTML), 273-275
<h>, 278
anchor, 281-282
tasks, programs, 6
temporary variables, 154, 188
testing
 automated, 415-416
XMLHttpRequest object 437

programs, 139-142
beta, 141
desk checking, 141
parallel, 141-142
text, formatting in HTML
(Hypertext Markup Language), 276-278
text editors, 13
typing first program, 16-18
top-down design, programs, 48-50
totals, accumulators, 152-153
trailing spaces, 314
training, programming, ongoing
needs, 419-420
twips, 362

U
UML (Unified Modeling Language), 418
uppercase strings, 88
user sessions, setting and
reading, PHP, 329
user-defined functions, 321
returning values from,
322-323
user-program agreements,
programs, 40-41

V
values
assigning, 61-63
swapping, 154-155
variables
array, 149-150
global, 224
Java, 223-226
arrays, 226
Boolean, 225
center, 225-226
floating-point, 225
initialization, 224
integer, 224-225
string, 226
local, 323
object, declaring, 349-350
PHP, 310-312
functions, scope, 323-324
integer, 315-316
swapping values, 154-155
temporary, 154, 188
watch, 127
Visual Basic, 333, 359
applications, creating from
scratch, 361-368
code, finalizing, 366-368
environment, 359-361
language, 370-371
procedures, 369-370
property values, 367
twips, 362
Visual C#, 383-387

W
WANs (Wide Area Networks), 399
WANs (Wide Area Networks), 399
Web browsers, Java, usage,
208-209
Web pages
bandwidth, 257
displaying, 273
hyperlinks, 281-282
inclusion, graphics, 281-282
placingJava applets in,
261-264
rollover, 66
viewing applets, inside,
264-265
Websites
adding repeating news ticker
to, 179-183
while loop, 107-109, 233-234
whitespace, 15, 119-120
Wide Area Networks (WANs), 399
writing
applets, 255-261
C general program functions,
343-345
code, 53, 119-120
cookies, 190-191
relational tests, 96-100

X-Y-Z
XML (Extensible Markup
Language), 287, 294. See also
AJAX (Asynchronous JavaScript
and XML)
XMLHttpRequest object, 289-291