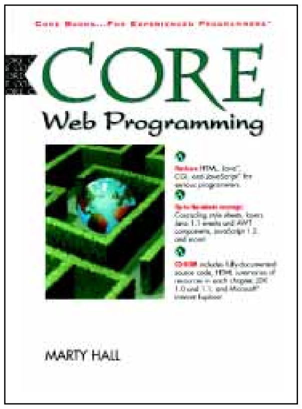

Webspace Engineering

Ben Benokraitis

CORE WEB PROGRAMMING

Marty Hall, The Johns Hopkins University Applied Physics Laboratory
Published by Prentice Hall PTR, Upper Saddle River, NJ, 1998, 1277 pp., $49.95

Assume that you're interested in obtaining funding for a project to power ships with cold fusion technology. To convince your prospective sponsor of the worthiness of this project, you decide to develop a computer simulation touting the benefits of the revolutionary approach. You have heard a little about object-oriented programming and its usefulness in developing simulation systems, so you explore this possibility. At the Applied Physics Laboratory, you need look no further than Marty Hall (and his merry band of software engineers) to help you win the day and the funding. For the rest of us, we have his book.

And a fine book it is. But don't be misled into thinking that the flash in the simulation comes easily. Besides the science you're concerned about, the computing science in the simulation must be coordinated meticulously. As the renowned computer scientist Allen Newell once said, "Science is in the details." Well, there are enough details (including a few "daemons") in Marty Hall's book to classify his treatise as a science many times over.

It is clear that Marty Hall enjoys his craft. Despite the length of the book, the clarity, enthusiasm, and humor of the author never subside. From examples that exhort the reader to invest in cold fusion technology to poking fun at software engineers (these must be geeks) who fail to recognize hex input as integer input, the chapters can be read quite quickly. But assimilating all the information will take longer. The book is written for experienced programmers, and if you are not among them, this book may not be for you. In any case, it will give you a better appreciation for what a hacker does and laughs at daily (see the explanations of hacker and cracker on page 1029; yes, page 1029.)

According to the author, the purpose of the book was to provide a text for a course called "Distributed Development on the World Wide Web" offered through The Johns Hopkins University's part-time graduate program in computer science. Because no single book was available for this purpose, the author set out to write his own book to cover most of what a professional programmer needs to know in Web programming. The book is divided into four parts dealing with the major topics of the original course: Part I, The HyperText Markup Language (HTML); Part II, Java; Part III, CGI Programming; and Part IV, JavaScript.

From the beginning, the author encourages the reader not just to read, but to do and try (page xxxiv). The course was designed to have "a lot of hands-on projects," and this flavor is carried over into *Core Web Programming*. At one point, the author advises, "If you're wise, you won't sit down and read these Java chapters straight through, engrossing though they may be. <SARCASTIC> No doubt it will be difficult to tear yourself away, but you've got to do it. </SARCASTIC> . . . I suggest installing Java as soon as possible, reading a little, practicing a little, reading a bit more. . . ."

(page 235). Here the author invents his own <SARCASTIC> tag to get his point across.

Installing Java was not an idle suggestion. The accompanying CD-ROM contains the Java Development Kit, version 1.02 (Windows 95/NT, MacOS); the Java Development Kit, version 1.1 (Windows 95/NT); the WinEdit text editor (Windows 95/NT); Microsoft Internet Explorer, version 3; and an HTML guide of all the resources provided in the book, including the documented source code. Given the many versions and releases of the various Web browsers, HTML, Java, and JavaScript considered in the book, the author does an amazing job in explaining their features and pitfalls.

The five chapters in Part I deal with HTML, particularly HTML 3.2. They cover the design of Web pages, block-level elements, text-level elements, frames, and cascading style sheets. You will learn how to develop a fairly sophisticated Web page of your own, including tables, forms, and multimedia formats. Throughout Part I, as well as the entire book, couplets of sample code listings are provided with associated output screens. To summarize and to highlight significant elements, the author has strategically placed Core Approaches, Notes, Tips, and Warnings throughout the book. These are quite useful when the text is being read again (you will have to do this, as the author admits) to obtain a quick overview of the material. Also useful are the introductory paragraph and summary of each chapter; each summary also introduces the next topic.

The unifying portion of the book is Part II, which describes Java. Part II contains Chapters 6 through 15, half of the 20 chapters in the book. These chapters discuss getting started with Java; object-oriented programming in Java; basic Java syntax; applets, graphical applications, and basic drawing; handling mouse and keyboard events; windows; arranging windows using layout managers; graphical user interface controls; concurrent programming using Java threads; and client-server programming in Java.

Programmers appreciate that Java "can do Windows and take out the garbage," but is Java all that it's advertised to be? After describing the features of Java, the author "debunks" some of the myths about Java in Chapter 6. Although he is undoubtedly a proficient Java developer, the author takes the following position (page 233):

> . . . I think the carpenter's model fits the software world better. Using this analogy, the various technical approaches are tools in the software developer's toolbox. Clearly some tools are more broadly applicable than others, and certain tools are well suited to certain jobs. OOP [Object-Oriented Programming] is a useful and broadly applicable tool, and it should be in a central place in the developer's toolkit. But functional programming, structured programming, rule-based programming, divide-and-conquer approaches, greedy algorithms, and the like are also useful tools, and the expert craftsman should be skilled with them as well. OOP is complementary to some of them, independent of others, and occasionally in conflict with some. Choosing OOP as the underlying structure for Java was a wise choice, but once in a while the object-oriented viewpoint will get in the way. Rejecting a useful technique in Java simply because it "doesn't fit with the object-oriented philosophy" is, well, heresy.

This reasoned view should make even skeptical programmers who haven't used Java at least consider expanding their toolkit. And there is no better way than by reading (nay, practicing) the material provided in *Core Web Programming.*

In Part II (Chapter 7: Object-Oriented Programming in Java), the author begins a ship simulation and then enhances it (Chapter 14: Concurrent Programming Using Java Threads). Very few books on Java treat threads at all. The author again provides enough examples and working code to make the concepts understandable. Indeed, students interested in operating systems would be well rewarded in turning to Marty Hall's treatment of user-level threads. Moreover, most of the "flash" a sponsor would be interested in seeing in a simulation is covered in Chapters 9 through 13, which deal with graphical programming. Among other topics, Chapter 15 discusses how to connect to different relational databases using Java DataBase Connectivity (JDBC).

Parts III and IV cover CGI Programming (Chapters 16–18) and JavaScript (Chapters 19–20). CGI (Common Gateway Interface) Programming allows a Web page to connect to a remote system, while JavaScript is a scripting language that runs on Web pages, allowing you to generate HTML dynamically or to take some action when a monitored event occurs. An interesting application developed in Part III lets you interactively select various font, color, and indentation values to reconfigure the "style" of a Web page (first introduced in Chapter 5: Cascading Style Sheets). To mention just one application in Part IV, JavaScript is used to store and examine "cookies," which track user preferences and activity while connected to a Web site.

You'll find many other interesting examples throughout the book, including advertising generation (AdBuilder), politically correct resumes (WonderWidget), buttons (ImageButtons), and databases (ShowTable), as well as a discussion of privacy and security issues in a number of cases.

Here's the rub (or "daemon"). There are so many topics to consider, it's not clear what should be covered first. The author himself struggles with the situation when deciding to cover window usage before discussing buttons, text areas, and the like, which are things that go into a window. The best approach would be to separate the 20 chapters into threads and to process them concurrently (ha!). Another is to read the text like a hypertext document, skipping around forward and backward. If you read the book closely, it's filled with both forward and backward references, all of which should probably be followed like hypertext links.

In Chapter 8, the author introduces the Java Hipness Factor (JHF) but fails to give a scale and range of JHF values. No matter. In that respect, both of this reviewer's thumbs are up.

Professor Ben Benokraitis
Chair, Department of Computer and Information Sciences
Shepherd College
Shepherdstown, WV 25443
vjbenok@shepherd.wvnet.edu