Book Reviews

ELIZABETH ZWICKY, WITH MARK LAMOURINE

Team Geek: A Software Developer's Guide to Working Well with Others

Brian W. Fitzpatrick and Ben Collins-Sussman

O'Reilly, 2012. 160 pp.
ISBN 978-1-449-30244-3

The Developer's Code: What Real Programmers Do

Ka Wai Cheung

Pragmatic Bookshelf, 2012. 141pp.
ISBN 978-1-934356-79-1

These make a nice pair; each is the programmer's equivalent of a glossy management book, with simple, good advice well packaged. Simple ideas are easy to read and easy to believe in and sadly very, very hard to implement. I like these ideas a lot.

Team Geek is a unified book in smoothly integrated chapters with pictures; *Developer's Code* is in short essays. Otherwise, they differ primarily as suggested by the subtitles. *Team Geek* is mostly about teamwork, as a team member or team leader; *Developer's Code* is mostly about subjects that are closer to the code itself.

In *Team Geek,* as always, I disagree with some of the details of the specifics (the compliment sandwich, like the direct order, has its place; the trick is recognizing that place on the rare occasions you encounter it). But that's inevitable, and the general outlines are spot-on. Most people underestimate the importance of being honest and nice.

Revolution in the Valley

Andy Hertzfeld et al.

O'Reilly, 2005. 290 pp.
ISBN 978-1-449-31624-2

The real story of the Macintosh. If you're at all interested in computing history, or "how it was done" stories, you should read this. It's a top-notch job of being as non-fictional as possible, and it beautifully captures the combination of insanity and exhilaration that goes into groundbreaking work.

This is not a story I was part of, and it's before I came to Silicon Valley, so in some sense it's not a world I know. And yet, at the same time, it is the world I knew then (a time when we saved the foam that one of our computers came in because it was handy for sleeping on when you had been programming too long and needed a nap). Furthermore, it is also the world of the most recent startup I was at. Having the stories told by the participants gives this a sense of truth that most tales of computing history don't possess.

This came in (reprinted) for review right around the time Steve Jobs died, and I put off looking at it partly because of the noise at the time. Although Steve Jobs appears, this is not a book about him, and it depicts him with the full complexity of feelings that people had at the time.

The Design of Design

Frederick P. Brooks, Jr.

Addison-Wesley, 2010. 448 pp.
ISBN 978-0201362985

I love the author's classic book *The Mythical Man Month* with a passion, and I was therefore prepared to adore this as well. Unfortunately, I did not. It's nice enough; there are interesting insights into a number of things, most of them not directly related to programming, and anybody who has used JCL is sure to enjoy the discussion of how it came to be the worst programming language ever (the author's description, although I wholeheartedly agree).

If you're the kind of person who wants to engage in meta-cognition about the nature of design as an interdisciplinary undertaking, this is an interesting resource. If you were hoping for a straightforward sequel to *The Mythical Man Month,* you are liable to be disappointed.

Understanding IPV6, Third Edition

Joseph Davies

Microsoft Corporation, 2012. 640 pp.
ISBN 978-0-7356-5914-8

Another case of dashed expectations. I want to like IPv6. I want to understand IPv6. This is not the book to help a person do either. It is a detailed and careful reference work for IPv6, particularly on Microsoft platforms. If that's what you need—packet layouts, protocol details, all the relevant registry settings—then this is a fine book.

I would recommend skipping the prose, however. This is the paragraph, in Chapter 1, at which I gave up:

"One must consider, however, that the Internet, once a pseudo-private network connecting educational institutions and United States government agencies, has become an indispensable worldwide communications medium that is an integral part of increased efficiency and productivity for commercial organizations and individuals, and it is now a major component of the world's economic engine. Its growth must continue."

Things get better, but not lighter, when the topic turns to protocol details.

—Elizabeth Zwicky

CoffeeScript: Accelerated JavaScript Development

Trevor Burnham

Pragmatic Bookshelf, 2011. 127 pp.
ISBN 978-1-93435-678-4

I hadn't heard of CoffeeScript until I saw these books on the shelf at my local book store. I leafed through them wondering what kind of cutesy software would have that name and found a complete new browser language to replace JavaScript.

CoffeeScript is an attempt to address some of the long-lamented flaws of the JavaScript syntax. It adopts the design philosophy of more recent scripting languages, especially Python and Ruby. At the same time, it maintains close ties to JavaScript, even allowing in-line JavaScript if needed.

CoffeeScript is actually a translated language. The "compiler" produces JavaScript suitable for inclusion in HTML pages or for execution by a Node.js interpreter. The translator is in fact written in CoffeeScript and can execute in a browser (both authors explain how, and then why you shouldn't). More typically, it runs in a Node.js interpreter, and the resulting JavaScript output is what is used in production. It irks me to call the CoffeeScript interpreter a "compiler," but I will follow CoffeeScript convention in this.

CoffeeScript: Accelerated JavaScript Development is not a lengthy tome. Burnham states in the preface that readers should have at least some experience with JavaScript. It might be even more help if they can code Ruby. It is implicit that they should have some background in software development as well. You'll see why in a moment.

The opening chapter glosses how to install Node.js and the CoffeeScript compiler. It gives an example of how to compile CoffeeScript source code to JavaScript. It also shows how to run it as an interpreter in interactive mode.

Only three language constructs (writ large) are explicitly covered: Functions in Chapter 2, Collections and Iteration in Chapter 3, and Classes and Modules in Chapter 4. Burnham relies on the code examples, the reader's experience, and the similarities to Ruby to help the reader along. (There is also a Cheat Sheet for JavaScripters in Appendix 3.) The final two chapters discuss using CoffeeScript with JQuery and writing server side code with Node.js.

In keeping with the "pragmatic" theme in the Pragmatic Programmers series, the preface includes the description of a project that is used to illustrate the new concepts. Each chapter contains a coding section based on that project. The chapters close with a set of example questions designed to highlight the problems that most JavaScript programmers encounter when learning CoffeeScript.

Burnham provides three appendices to his book. I've already mentioned the cheat sheet for converting JavaScript constructs to CoffeeScript and back. The other two are the annotated answers to the chapter example questions and a section listing six different ways to run CoffeeScript.

CoffeeScript is a great introduction to the CoffeeScript language for someone who is familiar with and frustrated by the warts of JavaScript. It will be a very welcome book for someone who must use JavaScript and is comfortable with Ruby.

Programming in CoffeeScript

Mark Bates

Pearson Education, 2012. 283 pp.
ISBN-13: 978-03-2182010-5

If you're a JavaScript coder, *Programming in CoffeeScript* is meant to sit in a handy place on your desk. The Pearson "Developer's Library" series promises reference works for professional developers, and Bates' book fills the bill.

This is the first book I can remember that has a test for the reader in the preface. Bates presents a fragment of JavaScript and tells the reader to put the book down and get comfort-

able with JavaScript before returning. I think that's probably prudent as well.

The preface also contains the mandatory "How to Install CoffeeScript" section, but Bates reduces it to four short paragraphs in which he explains that he can't do better in print than the writers on the CoffeeScript Web site.

The first half of *Programming in CoffeeScript* is, as you would expect, an exposition of the language syntax and programming constructs. Bates walks thoroughly and methodically from a chapter on literals and variables through to classes and inheritance. He follows what I think is a fairly new convention of combining the traditional chapters on loops and arrays into a single chapter entitled "Collections and Iterations." This comes from the Functional Programming school, but I find it reasonable to highlight the relationship between collections and their most common operations. He still includes a chapter on classic logic and control structures but again combines them in a way that makes sense to me.

Bates' explicit reliance on his reader's prior knowledge lets him avoid lengthy expositions that can bog down texts that try to teach both a language and the theory of programming. This makes for a crisp, smooth read and makes the book suited for use as a long-term reference.

Bates also uses the reader's prior knowledge and the CoffeeScript compiler's remarkably clean output to his advantage. Every sample of CoffeeScript code in the book is followed by the resulting JavaScript code and by the output it generates when executed. I can hear the horrified gasps of aging C and C++ programmers who have looked under the cover of cpp(1) and cfront(1) output and of all those who hate XML because they cut their teeth on machine-generated data streams. The JavaScript is both nicely formatted and actually maps cleanly back to the CoffeeScript source. I've tried it myself, and it's not just the result of good typesetting. There are some constructs that take an extra look (the section on Binding and the -> and => operators at the end of the chapter on Classes jumps to mind), but Bates highlights them well and covers them in more detail as needed. In general, the samples in both languages give the reader a cross-reference which helps to clarify both the intent of the example and the proper meaning and use of the CoffeeScript language. I learned a few things about JavaScript too.

It is in the second half of the book that Bates dives into the practicum of using CoffeeScript.

It seems (and I think this is a good thing) that it has become accepted that no language is complete without both a unit-testing framework and some form of task-based build tool. Again, CoffeeScript borrows from Ruby. Cake is included with CoffeeScript. The structures generally follow Ruby's Rake (which in turn builds on lessons from Maven, Make, and others). Bates provides a nice set of sample tasks and then indicates that he generally uses Rake himself.

Because CoffeeScript translates to JavaScript, you can use any of the JavaScript unit-test frameworks to test your CoffeeScript. Bates suggests a framework called Jasmine, which is modeled on Ruby's RSpec and which has native CoffeeScript support.

The book concludes with four chapters in which Bates builds a Node.js service on CoffeeScript. By itself this wouldn't be a big deal, but in the process Bates introduces a number of other components which will be needed for real applications. These include an application framework, database integration, and client-server communications with both JQuery and a CoffeeScript MVC framework named Backbone.js.

Programming in CoffeeScript is a great introduction to the CoffeeScript language and development ecosystem. It should also get more than its share of dog-earing from regular use.

—Mark Lamourine