BOOKS

Book Reviews

MARK LAMOURINE AND RIK FARROW

Learning Angular 2
Pablo Deeleman
Packt Publishing 2016, 326 pages
ISBN 978-1-78588-207-4

Reviewed by Mark Lamourine

If there's one thing I've discovered from my attempt to learn client-side Web programming, and Angular 2 in particular, it's that I'm glad I'm not a Web programmer. Creating a Web app these days, even with frameworks to standardize many of the constructs and behaviors, requires the use of at least four languages (I count JavaScript, HTML, CSS, and templating as distinct languages). In many cases, one or more of these are interlaced in a single file. Web design frameworks take some of the burden by providing a well-defined set of tools for the developer. Angular is Google's attempt to create a JavaScript framework to assist in and standardize the creation of single-page client-side Web applications.

Learning Angular 2 doesn't really help my impression much. The book is based on RC1 of Angular 2 and was published in May 2016. Angular 2 went to first release (2.0) in September, after six more release candidates. While Google is promising increment-only releases (using "semantic versioning"), this doesn't give me warm fuzzies to start. I was less happy when, on downloading the sample code from GitHub and trying to follow the installation process, I found that the required libraries were already advanced and out of sync with each other. An experienced JavaScript coder and NPM user would have solved this in moments, but it took me an hour or so merely to get to where I could start the actual samples in the book.

Once I got past this, the text runs in a fairly typical way. In Chapter 2, Deeleman introduces TypeScript, a superset of ECMAScript 6. The language itself is very straightforward. It should be safe as it is managed by Microsoft and is the source language that Google selected for Angular 2 itself. But again, I'm a bit disconcerted by the explanation for the existence of TypeScript, which is essentially that none of the standards bodies could agree on what client-side scripting should look like, so Microsoft took it on themselves to decide. It may be a good thing, and it's not Deeleman's fault in any case, but it doesn't instill confidence in a new learner.

Deeleman's "hello world" example is a Pomodoro timer. He explains that this is a kind of work-tracking device to help break down tasks. He guides the reader through the creation of a simple app in Chapter 3, and the rest of the book extends the application with new features. I like how he presents his code samples, offering a complete file or feature first, then breaking down the parts and explaining how they interact or relate. I prefer this to a style that presents small, digestible but apparently unrelated fragments and then composes them at the end.

There are places where the narrative gets lost, though. Deeleman states that he expects the reader to have a comprehensive understanding of JavaScript, but it feels at times as if he's presenting incomplete or circular definitions for terms: "Angular 2 defines directives as components without views. In fact, a component is a directive with a view." But there's very little time spent on what a component is and why that is significant.

Deeleman manages to cover the major points and features of Angular 2: component design, composition, standard directives (logic for producing and laying out the custom content that components present), HTML templating language, and client-server communications. He doesn't go deep into the theory or philosophy behind how and why these elements work together the way they do. His approach is mechanical, but it is effective on that level.

Packt tends to publish early books, and it seems sometimes that they spend less effort on editorial work than some of the more prestigious imprints. In an environment where frameworks like Angular can come and go in a publishing cycle, this makes some sense. They provide for a market of readers eager to learn new things that more conservative publishers might pass over or miss completely. Having lamented the thin supply of books on Go, Docker, and Kubernetes, I appreciate what they do. Readers should be aware, though, of what they are getting.

If you're an experience client-side Web developer looking for a self-tutorial on Angular 2 to supplement the documentation already on line, *Learning Angular 2* will serve. Anyone hoping to learn to design Web apps from scratch will have to work harder to grasp the context and operations that Deeleman leaves out.

Mastering Angular 2 Components
Gion Kunz
Packt Publishing, 2016, 352 pages
ISBN 978-1-78588-464-1

Reviewed by Mark Lamourine

My first experience learning Angular 2 was a challenge at least in part because of my own inexperience with client-side Web development, but I didn't want to stop with a single try. My reading of *Mastering Angular 2 Components* gets the benefit of that experience.

This book is also based on Angular 2 RC1 and was released in June 2016, so the same risks apply regarding bit rot as applied with *Learning Angular 2*, but I didn't have any problems preparing the working environment this time.

Kunz begins by introducing terminology and tooling, and he spends significant time both defining terms and explaining why they matter and how they relate. While I understand classes and decorators from other languages, I appreciated the paragraph or two he gave to each, explaining how they are defined and used in TypeScript and how this relates to ECMAScript and JavaScript standards. I'm still not comforted much by the state of language development for Web programming, but at least I now better understand the technical aspects of the decisions.

Kunz alternates well between developer and application user realms, which clarifies the reasoning and the choices that the Angular 2 developers made when designing the framework. He has peppered the text with diagrams to help clarify the relationships between components and directives and how these are related to views and templates. Chapter 7, "Components for the User Experience," makes clear who we are actually writing our apps for. The composition of complete services, both the presentation and logic, seems natural and meshes well with Kunz's exposition of the language and framework features that Angular 2 provides to the developer.

I especially liked the section which treats CSS and how the CSS elements are bound back to the HTML to influence the visual presentation. It is easy for a coder to treat visual presentation as subsidiary to data structures and logic (I am guilty). Kunz spends time showing how the design of the templates and data bindings in Angular 2 components can be influenced by the intended presentation and why it is important to consider the presentation hooks during development of the components.

I also appreciated his clear treatment of how data, both input and output, is bound to HTML template elements. It is an aspect of Web programming that had confounded me for some time. He shows how to create structures to present data both as text and graphically, using both CSS and SVG to create dynamic visual elements: graphs, charts, sliders, and interactive controls.

The final significant topic that piqued my interest is a section on the interactions between client and server, including timing and response mechanisms. While others have described the syntax necessary to create and respond to triggers and data exchange, Kunz is the first I have seen to clearly diagram the sequence of real-time events and communications that result from these coded elements.

As an experienced developer in other realms, I was comfortable working through each step of the learning process as presented by *Mastering Angular 2 Components*. I think this is one I'm going to come back to as I work on my own first Web service.

The Practice of System and Network Administration, Volume 1, 3rd Edition
Tom Limoncelli, Christine Hogan, Strata R. Chalup
Addison-Wesley, 2016, 1168 pages
ISBN 978-0-321-91916-8

Reviewed by Mark Lamourine

It's been a decade since the release of the second edition of *The Practice of System and Network Administration*. In that time, the character of system administration has changed and expanded in ways few of us anticipated. Each edition has been a comprehensive survey of the aspects of system administration in its era. Since the release of the second edition, configuration management has become commonplace, virtualization has moved from the desktop to the datacenter and the cloud, software development has accepted the tenets of Agile processes, and software revision control has become a public service. Containers have been rediscovered, though I don't think we can see yet what the results will be. (There is a Volume 2 which deals specifically with cloud administration. This is not that book.)

From the first, *TPoSaNA* (I generally avoid acronyms and abbreviations, but I make an exception for a title this long) has been an encyclopedia of the profession. It is a welcome anomaly in the sea of technical tutorials and references. You're not going to learn to be a system administrator by reading it, but you can become a better one by scanning it and then keeping it handy for those times when you're not sure what to think or do. More than once I have pulled it out to show to a colleague or manager when I have needed an authority to back me up in some point of discussion, and it has proved very useful in educating managers in the scope of the work their people are expected to do.

The updates start with the table of contents. As an encyclopedia, it isn't surprising that this book has as many sections as most books have chapters (11). Thirty-two of the 56 chapters are new or updated (indicated by a marker on the title line). I liked the fact that I could thumb through and so easily find the places I needed to reread.

In this edition, the authors have dropped their 1st edition conceit of offering "The Basics," "The Standard," and "The Icing" levels of support for each topic. They still open with a clear discussion of the topic scope and goals, but then use a more conventional approach to the detailed discussion. Often they justify or illustrate their choices with anecdotes from their own work, showing how the problems arise in the real world and how they responded. Every chapter is peppered with references to other resources and ends with a set of exercises, which are really prompts for readers to think about what they've just read in the context of their own work environments. This works well to help readers relate the new ideas to their own work.

Most people would expect a book on system administration to include operating system and software installation and configuration management. Many would expect to see guidelines for help-desk management. I think many (non-sysadmin) people would be surprised to see power and air-conditioning management under the umbrella of system administration. I know I have welcomed the chapters on how to manage time, not just to do the job well but to remain sane and happy. I haven't needed the chapters on hiring and firing, but I know technical people who have grown to lead or manage groups of developers or admins and appreciated it. In a technical field, it's just not something you have to think about…until you do.

For people who call themselves system administrators, I can't recommend having a handy copy of this third edition of *TPoSaNA* highly enough. For managers of system administrators, I recommend it even more highly. If you've been in the profession for long, there's likely a lot here you already have heard, but this book is the perfect starting resource for those times when you or your colleagues find yourself having to extend yourselves to Get the Job Done.

The Hardware Hacker: Adventures in Making and Breaking Hardware

Andrew "bunnie" Huang
No Starch Press, 2017, 416 pages
ISBN: 978-1-59327-758-1

Reviewed by Rik Farrow

Based on the title and subtitle of this book, I thought Huang was writing about, well, hardware hacking. But a lot of the book is about manufacturing hardware in China for small production runs. It wasn't long after I realized that the title poorly describes the content that I got over my disappointment, however.

bunnie Huang is not just a brilliant hardware designer, he's a great writer, too. His style is conversational, clear, and concise, and I found myself wishing that more people could write like Huang. He explains that he started visiting factories in China to support the manufacturing of the Chumby, a dedicated MP3 player with touch screen he designed in the early noughts. Huang describes just how fascinating he found the mega-bazaars of Shenzhen, China. But his real focus is the factories, how important it is to find the right factory and to communicate clearly what you want them to do.

Huang uses a bill-of-materials for a bicycle safety light as an example. While the device just requires eight parts, getting it manufactured correctly requires an entire page full of detailed information. Almost as an aside, I learned what RoHS means (Restriction of Hazardous Substances) and how one region's standard can mean safer products for everyone.

Huang's writing does tend to wander, but always in directions that I found fascinating. For example, he gets to visit a factory that makes zippers, starting with zinc/aluminum ingots. He asks the same question I might have about why one processing line required a human to align zipper pulls while other lines did not. The answer is subtle, a tiny tab found on most zippers. Yet this diversion helps to illustrate an important point about the manufacturing process: that what might be important to a designer, lack of the tiny tabs, causes problems for manufacturing.

Huang uses several of the projects he has worked on to illustrate the problems that a hardware hacker, intent on actually going on to produce product runs in the low thousands, will encounter. One issue is fake parts, parts that appear authentic but are actually just the casing with no electronics inside. Other issues appear truly ridiculous but are no less real. When Huang and a partner required a spiral notebook where the spiral had to be non-conducting for Chibitronics, the manufacturer didn't understand what that meant. Huang bought a pair of volt-ohm meters and taught the manufacturer how to use them to test the spiral bindings for conductance.

The chapter on fake parts, something that can be a problem when manufacturing in China, is the closest Huang comes to hardware hacking in my mind. Huang provides photos of SD cards, then has them stripped down to the chips hidden inside the epoxy resin. He does talk about some of his other hardware designs: for example, the hacker's laptop he co-designed and manufactured, but manufacturing is still the largest theme.

Toward the end of the book (Chapter 10), Huang veered off into an area I felt he couldn't possible handle: biology and bioinformatics. Huang compares a metabolic diagram to an Apple II schematic, then imagines DNA and RNA as configuration bits. I was wrong about this chapter, as Huang manages his comparisons brilliantly, using hardware as a way to explain genes, proteins, and amino acids. He goes on to describe the flu virus and how it manages to continue to evade vaccine designers using just 3.2 KB of "data" in its genome.

I found Huang's book both easy and fun to read. If you are curious about the manufacturing culture in China, including its own version of "open source," I recommend reading this book.