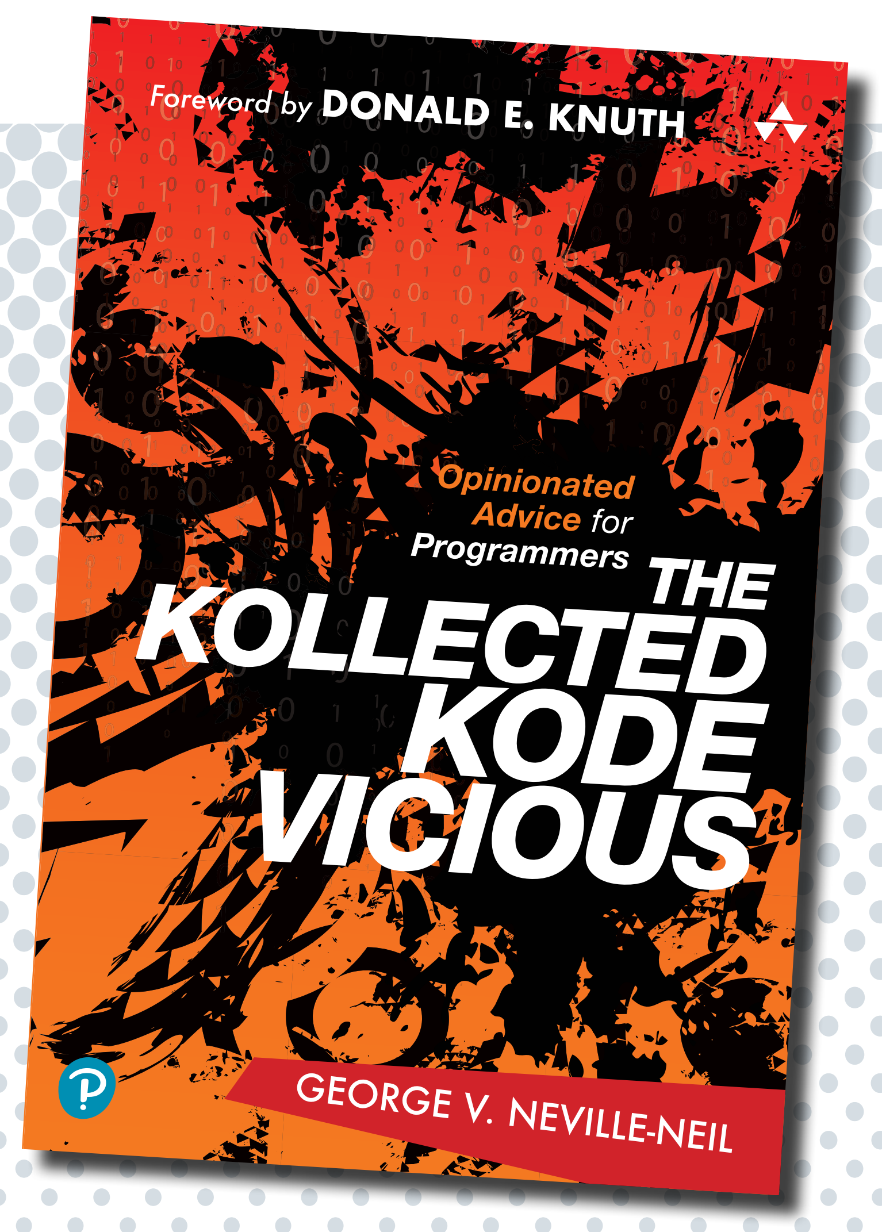
A review of *The Kollected Kode Vicious*

BY BENEDICT REUSCHLING


Foreword by DONALD E. KNUTH
Opinionated Advice for Programmers
THE KOLLECTED KODE VICIOUS
GEORGE V. NEVILLE-NEIL

Dear Practical Ports,
Should I set aside some time to read *The Kollected Kode Vicious* by George V. Neville-Neil instead of a book about a new programming language — one that I need for my job? I don't have much time for reading, so I need to choose carefully. I also don't see how someone using a pseudonym like KV instead of their real name can be taken seriously, especially when giving such opinionated advice to other people.

— A Skeptical Reader Without Much Time These Days

Dear Skeptical,
Have you ever come across the name Alice Addertongue or Silence Dogood? Or maybe Poor Richard rings a bell? These were pseudonyms used by none other than Benjamin Franklin. Pseudonyms and caricatures were very commonly used during Franklin's time, even though it was often clear who was behind the writing. Not only did the different caricatures allow Franklin to offer readers (of his own newspaper) a different perspective on a wide variety of subjects, but they also permitted him to use hyperbole, sarcasm, and whimsy to entertain readers and shape their opinions. So, writing under an assumed name and persona is not new, and, actually, as you're writing to Practical Ports, you're also participating in the scenario.

Before we get to the book, let's address your problem of setting aside time for reading books.

"Everybody has time. Stop watching fscking Lost!"
— Gary Vaynerchuk, in 2008 at Web 2.0 Expo NY
https://www.youtube.com/watch?v=EhqZ0RU95d4

But, ok, that aside, I can understand that time can be hard to come by these days. Plus, there are only so many books you can read (bonus points if they are good books!). Doomscrolling the news or social media does not count as reading, nor does getting caught in the endless Youtube loop. Books are still relevant sources of knowledge and entertainment — perhaps now more than ever — which is why I've devoted time to reviewing this one. Even with the sea of electronic information that is the Internet, there is timeless wisdom to be absorbed and pleasure to be had when turning physical pages.

Now, to your question of whether the *Kode Vicious* book is worth some of your apportioned time? For the uninitiated KV reader, I should explain that this book is a collection and grouping by theme of KV's columns from the *ACM Queue* column of the same name. The author wrote the columns over a number of years and *kollected* the best ones for the book, organizing them by topic and adding short intro pieces. How he got one of the giants of Computer Science, Donald E. Knuth, to write the foreword, will perhaps forever remain the author's secret. Talk about being knighted!

I'm a huge fan of quotes at the beginning of chapters and there are plenty of them here that encapsulate what you're about to read in the columns. (Example: *Oh Bullwinkle, that trick never works!* – Rocky J. Squirrel) If you have never read a KV column, know that they follow the revered question-and-answer/advice column format — a format so old that it goes back to Ben Franklin's deconstruction and reconstruction approach, and then, of course, to Socrates. The letters/questions in *The Kollected Kode Vicious* are framed as coming from a confused or puzzled individual seeking KV's advice. The questions are realistic and relevant and often resemble what I hear from students. Some of the later columns clearly address actual correspondence while some of the earlier columns are ghostwritten. In all cases, the answer/advice picks up the question and explains, defuses, deconstructs, rearranges, or discourages assumptions while shedding new light on the topic. Opinions are offered that are generally well intentioned and provide the sought-after advice. Many of these letters (if not all) could end with the words often used in Zen stories: "and thus the student was enlightened."

Checking out the table of contents reveals the following groupings:

- **The Kode at Hand** — deals with everyday annoyances and musings that programmers (like you) must deal with — from allocating too much memory, exception handling (or the lack of it) to proper logging and coding style discussions. This chapter will appeal mostly to programmers and is not for occasional computer users who wonder why the Internet is not working.
- **Koding Konundrums** — considers more philosophical questions like what makes a good programming language, avoiding spaghetti code that endlessly includes one file after the other, why tests are important, and meta-topics like code scanners and debugging strategies. All are valuable to read, and if you've programmed for several years, you'll definitely find a familiar story in there. If you ask yourself why the things are the way they are now, then the next chapter will enlighten you.
- **Systems Design** — looks at the choices (mostly bad ones) that people have made in designing systems we use every day or to program the ones we'll have to shake our fist at in the future. Evergreens like authentication vs. encryption, cross-site scripting, phishing and infections (the latter dealing with computer systems, mind you), and UI design will have you frequently nodding your head in agreement and shaking it at the prime examples of bad design. I will leave it up to you to find out why Java is listed in that chapter.
- **Machine to Machine** — discusses latency, failures to scale, protocol design, and the ever-growing list standards. if you know the author or know of him, it will not surprise you that this is not about networks connecting these machines. And if you think this is upsetting, then wait until you've read the next section…

- **Human to Human** — ponders how to name your hosts, leaving your pride out of it, code interview questions, and bikeshed coloring. People bring a lot of interesting traits to computing — if only they were good ones.

In each of these chapters, using stories that reveal insight and experience, the author cautions and educates. Although often presented with a grumpy undertone (perhaps from having seen too many of these instances), there is not one page where you get the feeling things are hopeless. Instead, there is a lot of practical advice mixed with personal preferences and recommendations and it's always imparted with wit and good humor.

I can imagine two excellent uses for this book. One is for light entertainment reading, and the other is for use as a reference book. The columns are brief enough to read during a short break (with a beverage of choice) or before going to sleep (nightmares notwithstanding).

Keep the book where you'll be able to grab it quickly so that, when necessary, you can re-read the relevant section about the problem you're facing. Another thought is to give it to your colleagues and peers when an argument ensues or when you have a pending design decision. "Let's turn to the Vicious book to remind ourselves of what not to do," can be uttered when discussion seems to be stuck and could use a fresh perspective. Both seasoned and new colleagues will find something interesting and relevant in these columns as they nod in agreement (been there, done that) or smile about an amusing anecdote.

Now, returning to your question of whether you should fill your limited reading time with The Kollected Kode Vicious over a book on a new programming language, I would say, "yes." I think you should read KV's book, but don't expect to find much advice like "use this clever code snippet to make your code faster" or "using Emacs as your editor will increase your productivity thousandfold." Another programming language may look good on a resume, but at the end of the day, these things are mere tools of the trade. There is so much more to what we do than writing code! This is what the book will explain to you as it offers a broad view of the joys and sorrows of the industry. Your mileage may vary, but I think it will be worth your while to read this book and internalize its teachings.

Let me close with another quote:

"Last year, a foolish monk.
This year, no change."
— Japanese Zen poet Ryokan Taigu

PP

BENEDICT REUSCHLING is a documentation committer in the FreeBSD project and member of the documentation engineering team. He serves on the board of directors of the FreeBSD Foundation as vice president. In the past, he served on the FreeBSD core team for two terms. He administers a big data cluster at the University of Applied Sciences, Darmstadt, Germany. He's also teaching a course "Unix for Developers" for undergraduates. Benedict is one of the hosts of the weekly bsdnow.tv podcast.