Really Know Your Tools

FROM THE EDITOR

Really Know Your Tools

Great programmers all possess deep knowledge of their primary tools. If you're not expert in yours, invest the time you need. Here's a start.

by Andrew Binstock

October 16, 2019

One of the benefits of my work is that it opens access for me to great developers, true domain experts, technical innovators, and wizards. In conversations with them, I've often been able to discuss what makes great programmers great. Their observations boil down to traits that are too infrequently prized instead of the convenient answer which is "clean code and write a lot of tests." For real greatness, it turns out, the question must be examined more deeply and less glibly.

From what I've seen personally and gathered from experts, great developers share several key traits. These are (in no particular order): the patience to fully understand the problem they're trying to solve; the ability to hold a lot of information in their heads at one time; the ability to move quickly from small to large and back down again; and finally, deep knowledge of their primary tools. Of these traits, the first and last are ones you can easily teach yourself. The other two rely more on innate abilities although, of course, they too can be acquired through discipline and hard work.

The deep knowledge of tools is a more difficult challenge today than it was 20 years ago. Modern full-stack developers might be regularly relying on dozens of tools, which makes it improbable that they will master them all in depth. However, I carefully point out that the *primary* tools are the ones that must be fully commanded.

Let's start at the very beginning: Can you type completely fluently? If you can't type without looking at your hands, you've lazily put yourself at a huge disadvantage. Not only will you work more slowly, but in ways very difficult to recognize, you'll be dissuaded from doing small things that require more typing. You'll avoid explorations that involve more than a few lines; you'll be dissuaded from writing the small script to automate a task you do repeatedly; and so forth. Even if your typing is imperfect (Can you type any digit without looking down? How about `!=`?), the time spent perfecting the muscle memory will repay itself with speed and the ability to focus entirely on the screen rather than the keyboard.

Let's move on to software tools. How well do you know your editor or IDE? Many developers are passably proficient in a small set of commands—the ones they use frequently and nothing else. For anything beyond entering code, compiling, and running tests, they are forced to rely on menus and mice. The drawbacks to this limited knowledge are the same as for imperfect typing: You work more slowly, in invisible ways you are discouraged from exploring new things, and you are constantly subject to micro-interruptions as you try to complete straightforward tasks.

A review of *The Pragmatic Programmer* will appear in the next issue.

What constitutes sufficient proficiency with your editor or IDE? In the 20th anniversary edition of *The Pragmatic Programmer*, David Thomas and Andrew Hunt lay out what they feel are the requirements of true fluency in your development environment. Can you do all these things *from the keyboard*?

- Move the caret and make selections by character, word, line, and paragraph
- Move the caret by syntactic units, such as matching delimiters, functions, modules, and so on
- Re-indent code following changes
- Comment and uncomment blocks of code with a single command
- Undo and redo changes
- Split the editor window into multiple panels and navigate between them
- Navigate to a particular line number
- Sort selected lines
- Search for both strings and regular expressions and repeat previous searches
- Temporarily create multiple cursors and edit the text at each one
- Display compilation errors in the current project
- Run the current project's test

I would add to this list: Be able to commit code and push it to a central repository, compile your code, and step through problematic code in the debugger.

If you answered "no" to any of these, you're in good company (including me, for sure). But most great programmers will know how to do all these things with ease—or nearly all of them. Thomas and Hunt prescribe steps by which you can acquire the necessary skills—the principal one being "Don't touch that mouse!" This forces you to look up and learn the commands.

They take their suggestions one step further: Identify tasks that you do immediately outside of your editor or IDE and then find plugins that automate or facilitate those tasks. In this way, your expertise in your principal tool expands to cover more work.

If you've ever pair-programmed with a great programmer, you've surely seen the speed and facility in action: the writing of code and tests followed by immediate refactoring with windows popped open to capture a reminder note or run a utility. It's a joy to watch (and, in fact, can be watched on various social media outlets). Be that person. Know your tools.

Andrew Binstock

Andrew Binstock (javamag_us@oracle.com, @platypusguy) is the editor in chief of *Java Magazine*. Previously, he was the editor of *Dr. Dobb's Journal*. He co-founded the company behind the open-source iText PDF library, which was acquired in 2015. His book on algorithm implementation in C went through 16 printings before joining the long tail. Previously, he was the editor in chief of *UNIX Review* and, earlier, the

founding editor of the *C Gazette*. He lives in Silicon Valley with his wife. When not coding or editing, he studies piano.

Share this Page