Mentoring VS-Coders as an Emacsian

Jeremy Friesen

[2023-10-03 Tue 09:47]

Contents

Mentoring VS Coders as an Emacsian

Presenter Jeremy Friesen

Pronouns he/him

Event URL `https://emacsconf.org/2023/talks/mentor/`

Audience EmacsConf 2023

Presentation Date and Time December 3, 2023 (2023-12-03) at 2:35 Post Meridian (pm) Eastern Daylight Savings Time (EDT)

Subjects Emacs, Mentoring, Visual Studio Code: Microsoft Sponsored Text Editor (VS Code)

Socials **Blog** Take on Rules (takeonrules.com)

Mastodon dice.camp/@takeonrules

Email jeremy@jeremyfriesen.com

Background

Since 2015 (2015) I have mentored over 40 software developers; many that were in the process of changing careers into software development. I've also managed a couple of small software development teams.

Framing Approaches

Remember, we all don't know what we don't know.

Be curious while mentoring.

- Ask questions.
- Be visible.
- Pair to share.

What Are You Looking to Learn?

When I start mentoring folks, I ask some form of the following question:

What have you been wanting to learn more of, get better at, and improve on?

Then I ask that they *tell me more*.

When we later meet, I like to use coaching questions:

- What's going well?
- Where are you getting stuck?
- If you could change one thing, what would it be?

Make the Work Visible

Like many people, I shifted to remote work in 2020 (2020). I have noticed higher collaboration in remote work. . . when folks make their work visible.

I host office hours.

I try to attend other people's office hours.

I'll open up a Slack Huddle and code by myself; but let folks know they can hop in.

Hop In and Be Curios

I pay attention to other huddles that start; if they are still going after 45 minutes or so, I'll hop in to say hello. It's even odds that they are moving through the problem or are stuck.

By hopping into that Slack Huddle, I'm helping a common problem: **How to know you're stuck** . . . and when do you ask for help?

The earlier you ask, the quicker the course correction; but too early and perhaps you don't internalize the lessons of the struggle.

The later you ask, the more you'll hopefully understand of the sticky problem; however you may have been needlessly added stress and fatigue because others could've helped.

Pairing is for Sharing

I tend to **let others drive** in pairing sessions. They're typing and working to resolve the problem; and I'm there giving guidance. I'm also spending my time observing **how they interact with their text editor**. "In the moment," I **limit my advice to one concept**; saying something like: `control + a` will take you to the beginning of the line.

I'll gently mention that one thing once during our session.

And assuming we have a regular mentoring session, I'll make sure to ask how they're feeling about using their tools.

I'd love to get to the point where they ask me: "You saw me using my editor, what's some things you think I could/should practice/learn?" I'm working on that.

Editor Functions

While pairing, I like to pay attention to how folks handle the following:

- Where do they want to go?
- How do they get there?
- Here they are, now what?
- How do they summarize?

I know what I can do with Emacs; and I assume that VS Code can do something similar. It's a matter of helping the mentees find those packages/plugins.

Where Do They Want to Go?

Search within a project The subtle improvement of Orderless is undervalued

Search across projects Cross-repository search is simple in Emacs…and I've never seen someone do so in VS Code.

- Also show folks either Ripgrep: https://github.com/BurntSushi/ripgrep (rg) or The Silver Searcher: https://github.com/ggreer/the$_s$*ilver*$_s$*earcher* (ag)

Find file names Many folks use a directory tree to navigate; but I favor fuzzy text. I can get there quickly (see `consult-projectile`).

How Do They Get There?

Navigation through Language Server Protocol (LSP) I bind `M-.` to this; it hurts to watch folks use the mouse to do this simple jump.

File finders `consult-projectile` is an amazing multi-function finder.

Jump between definition and test I bind `s-.` a sibling of jump to definition; I want more folks to use the Ruby / RSpec plugin in VS Code.

Here They Am, Now What?

Word-completion dabbrev, templates, hippie expand, and completion-at-point; it hurts to watch folks type

Auto-formatting Tree Sitter; I haven't seen folks install VS Code packages for auto-format.

Multi-Cursor / IEdit It took a long-time to fully explore iedit; but the practice I did transformed my approach.

Inline searching My beloved Textmate introduced me to line filtering; to this day, I use that to understand the neighborhood of text around the text I'm searching.

How Do They Summarize?

The default interface for commits within VS Code is the equivalent of an Hyper Text Markup Language (HTML) `input[type=text]` field. Not a `textarea`, but a 30 character text field. That one foible is the primary functional reason I stopped considering VS Code for my text editor and settled on Emacs. Seriously.

I've seen a lot of boot camp graduates trained to **write commit messages by going to the command line** and typing `git commit -m`. They then use their command line to type out their message.

In my experience, commit messages written on the command line tend to be terse and missing useful context related to the commit. By shifting folks to use their text editor, we're encouraging an interface where they can write more expressive messages.

Teach them about `$GIT_EDITOR` and `$EDITOR`; they're making commits from the command line. At least help them learn to use their editor.

General Strategies

My goal is to encourage folks to use their editor for writing. To think about *owning* that tool.

Commit to One Item of Learning Each Week

There's an absurd abundance of short-cuts, automation, and approaches to solving problems. Don't overwhelm folks.

As folks are solving problems that "earn" their income, nudge them to practice one thing. That one thing is easier to slot into their everyday work.

Practice within Your Knowledge Domain

> Any sufficiently advanced hobby is indistinguishable from work.

I also encourage folks to practice working on problems within a comfortable-to-them domain of knowledge.

I play table top **RPGs!** (**RPGs!**); and use that domain as the material for practicing coding. In years past, I'd write Ruby code for dice rollers, note takers, random table lookups. Now I'm doing more of that in Emacs.

Note Taking

Pay attention to how folks create a fleeting note. Some will be in their text editor and anguish about where to create a new file. Others will have a notes application running and write something there.

Help folks think about their note writing habits. Ask about it. Listen to their story and needs.

- Help Them Navigate the Proprietary Software Tar Pits

  One person was a dedicated Evernote user who has watched their digital contents evaporate. They are approaching their needs very differently than the person that wants their mobile and laptop to have synchronized data so they can have on the go notes that update.

  My ideal state is for folks to use their text editor for note taking. Similar to commit messages. Through the language and driver of "getting better at using their text editor." No need to learn different short-cuts or lament having one tool available in one ecosystem and not the other.

- Help Show the Joy of Holistic Computering

  Put another way, many folks have ceded different aspects of their digital life to specific and isolated applications. Less prevalent is the holistic thinking of a generalized "computering" environment. Folks need help learning what they don't know, so they can choose how best to proceed for their lived experiences.

Playing is for Staying

I think one of the reasons I've remained a software developer is because I approach all of this as play and story-telling.

Some by-products of that play happens to be **shipped features** and **documentation**.

Yet I don't *tell* folks to use Emacs; instead I'm doing my best to *show* a myriad of reasons for why folks should consider Emacs.

Conclusion

Ask questions of those you're mentoring; namely how they are looking to improve. Know and show what your editor can do, so that you can make visible learning opportunities for those you mentor.

There's a mid-level engineer on my team and we both play the "Hey, I learned something new, can I show you?" game. Most important…we play with our editors.

A secondary goal is showing the malleablity of Emacs; how easy it is to extend.

And obviously there's so much more than what I've highlighted. But…that's Emacs.

Abbreviations

pm Post Meridian

EDT Eastern Daylight Savings Time

VS Code Visual Studio Code: Microsoft Sponsored Text Editor

rg Ripgrep: https://github.com/BurntSushi/ripgrep

ag The Silver Searcher: https://github.com/ggreer/the$_s$*ilver*$_s$*earcher*

LanguageServerProtocol

HSTML Hyper Text Markup Language