How to choose your tools

Yann Esposito

[2020-05-09 Sat]

+STARTUP: showeverything

This week I didn't take a look at HN to grab some news. And this week-end, in the morning I read those:

- Zoom acquires keybase
- Making Emacs popular again
- Github Codespace

Figure 1: Welcome to Halsingland

Similar articles have existed for years on different products. What is their common point? *Software tooling and their potential change and disparition.*

Accross the years, too many times I saw tools disapear. By tools I mean applications, web applications, web sites. I think we can also include programming languages, control versionning tools, building tools, package manager, etc...

The story can be quite different. Sometimes the disparition of a tool is positive, because I found a better one (from cvs to svn to git). But, too often, the tool

simply disapears or worse downgrade its quality. I think we can find different names for those softwares:

- *bloatware*: remember digg, stumbleupon, windows?
- *downgradeware*: Swagger-UI v3 (v2 is neat), reddit new redesign (looks better, but slow)
- *payware*: Useful free service ask for money now. Or cost a lot more.
- *crapware*: Stop to works, quality degrate unless you pay: Twitter streaming API?
- *dieware*: Remember Friendfeed, Google Reader™, etc...
- etc...

This is often quite frustrating because you lose a lot of your investment with that tool.

Regarding Github Codespace; the integration of VSCode™ inside GitHub™ can be even worse. This is what I would call a *trapware*.

trapware: A software that is intented to put you inside a closed ecosystem. By slowly but surely add features that while looking great for the user at first sight will ensure to entrave other tools to interoperate.

Furthermore, the fact that Microsoft is involved give this story a taste of Embrace, Extend and Extinguish.

My real concern is that it could become a *work framework*. This could impose the full tooling on a lot of developers without giving them the freedom of choice.

For a startup CTO/CEO this GitHub™ Codespace™ could offer the following advantages:

- *security*: impossible or very hard to steal the source code by a single dev.
- *homogeneity*: all dev must use the same development environment. Thus the integration of new dev is faster.
- *cheaper*: don't need to pay for a full featured, fast machine to each new developer. A less performant machine able to display an electron app will do the trick.
- *stats*: you can observe the throughput of your developers. How many commits a day, how many lines of code, etc... How much bugs involved which part of the code and thus which dev to blame? How much time the dev is typing, moving its mouse, how much copy/paste is involved, etc...

For the single developers and open source developers this offer:

- *homogeneity*: if I learn how to work in this environment, I'll be easier to recruit and I'll know how to work fast.
- *lower barrier to entry*: for an opensource repository, it will become much easier for anyone to propose a PR to fix some issue. No need to local clone the project, no need to download all the dependencies to test it locally, etc...

But the price to pay is hidden.

Figure 2: Midsommar Sorrow

1. First, you are now, not able to choose your local working environment on your machine.
2. GitHub™ can still change so much to become one of the previously mentionned `/.*ware/` you don't want to be involved with. They could forces you to pay a lot more, remove features, redesign to a bloatware, make it harder to interop with other platforms (prefer Azure to AWS etc...).
3. If everything involve machines in the cloud via the browser and via authorized plugins only. A lot of tools, features will never be allowed in this new ecosystem.
4. Surveillance on meaningless or wrong metrics about your work. Instead of being evaluated on the feature you shipped or on other higher level metrics. It will be very tempting for your bosses to find flaws in your working habits. We are already living in a world were surveillance, metrics and stats are too easy to grab about a person. And anyone involved know this is all bullshit. Human are very good to play those kind of games. So people really working hard for the best will certainly perform badly compared to other people that simply trick the system.

So as good as Codespace can be, I think it is good to keep that in mind. Don't put yourself in a trap.

The Zoom acquires keybase is just another story of a dying product. Apparently the keybase team will probably stop maintaining keybase. The idea behind keybase was pretty nice. And they filled a gap in the current open source world.

The last article I mentionned was Making Emacs popular again. The first comment in HN was about how VSCode is easy to start with as compared to Emacs that need a lot more time to configure correctly for your needs. Yes, VSCode

certainly just work and is easy to use. But Emacs is another beast. VSCode can become bad very fast, you don't control how it will evolve. The fact that this is a succesful Microsoft product does not garanty it will keep its currently quality. Emacs on the other hand is 44 year old and was designed so that it adapts to you. You are the one using libs and customizing it.

The argument to chose VSCode instead of Emacs look similar to me to the debate "Frameworks vs Libraries". Frameworks are easier to start with, but soon you find corner cases were you start to fight against them.

A Library on the other hand, is just a bunch of helpers you can use. And if you need another functionality, just make it using the libraries. But you have a lot more work to do yourself.

The common pattern I see during choice decision is often reducible to:

1. Easy now, but less extensible and harder in the long run.
2. Harder now, but more extensible and easier in the long run.

As a conclustion I would state that when you need to choose between different tools. Take the time to think about the investment costs. Sometime, the bit of pain in the begining is worth it. In particular if you are going to use this tool every days for many hours during the following years. If on the other hand you don't plan to use that tool much. Going with the easy option is certainly the best choice.

I consider Emacs to be of the 2nd option when compared to VSCode. Harder to start, but with a lot more control and potential power that you will probably never be able to get with most modern IDE/Editor. Also choosing a Free Software[1] gives you a lot more control about its future.

Post-conclusion – Emacs is awesome

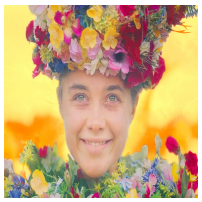

Figure 3: Midsommar Joy

To go beyond my opinion, I'd like to share my experience with editors and emacs.

When I started to be serious about coding, I was taught to use vi, not vim, vi. I only knew survival vi commands: `i`, `a`, `dd` and `cw`. A few years later I started to

[1] note I said *free software* and not *open source*; c.f Why Open Source misses the point of Free Software

use IDEs and I was thrilled. Again a few years forward I started to work for a company that forced me to use their shitty computers. Quite soon, I started to have wrist issues. Thus I decided to use vim again but be serious about it this time. And I saw the benefits only after a few weeks. They were tremendous. No more wrist pain. And an incredible edition power at the tip of my fingers[2].

Then, I started a new job where we decided to code in Clojure. Of course Clojure being a LISP and emacs using also a LISP as script language, it sound natural to try Emacs even though I loved Vim. I started by installing spacemacs. At that time I didn't want to invest much time in learning Emacs. I just wanted to learn the tricks that will make Emacs more valuable to my work. It did after just a few days or maybe weeks. I used Emacs superficially for years. This was already quite efficient, at least as much as vim.

Recently I dig deeper. I heard much praise about org-mode and I became curious. I discovered why it is so great. Basic org-mode is already quite valuable. But if you dig, it starts to be awesome. Unfortunately this is a bit hard to describe how org-mode is great without really digging a bit.

You can think of org-mode as an extremely versatile todo-list and note taker with agenda and time tracking integration. You are in deep control of your workflow. But mainly here are a few example of usages that are really worth it:

- note taker
- documentation; this is a far better than markdown
- interactive document; run code inside the doc, keep track of the results
- export to HMTL/PDF
- time tracking
- reminders

Recently there is also org-roam that is a step further to make orgmode a nice place to keep track of all your knowledge in one place. You can take a look at this great video by Matt Williams.

Emacs changed my workflow by making me more productive. It improved not only my coding workflow, but my full work environment. I started with the editor, a few plugins, and slowly, I integrated more aspect of my day to day tasks in emacs. Emacs is designed to adapt to your own needs. As such it is a lot easier to automate a lot of small tasks.

I really love Emacs and if you want to joyfully join the Emacs users here are my advices:

Start by using either spacemacs or doom-emacs. It will take a few weeks to absorb vim keybindings. Slowly you'll start to learn how to configure it for your needs.

I really advise you to take a look at org-mode. Mastering it could change your carrier. Im my opinion org-mode alone is a good reason enough to use emacs.

[2]Lear Vim Progressively is an old "popular" blog post of mine.

But there are a lot more to discover.

However, if you are used to tools from startups, with nice UI/UX. Almost no configuration cost. Be aware that digging in Free Softwares is a lot diffierent. Instead of having a big bundle with everything prepared to work you you will need to take the time to configure each part of a big system separately.

Howevery I'm deeply convinced the investment is really worth it.