Technical Opinions

Yann Esposito

[2020-11-01 Sun]

About how I forged some of my technical opinions. How I judge some technologies.

During the early age of AI, a very promising field was Expert Systems creation. The goal of those systems were to put the knowledge of experts into a computer. As an example they tried to put the knowledge of doctors to improve and perhaps even automate diagnosis. And guess what? It was a complete failure.

One advanced reason was due to doctors (and experts in general) not playing it fair by fear of being replaced. But I am more convinced by another explanation:

Doctor were unable to describe correctly their knowledge.

I believe part of their knowledge was unknown from them. Or said otherwise; *they didn't know they knew*. Part of that knowledge was simply an evidence to them that need no explanation.

Why? I think it ends to internalisation. The knowledge hardly learned through years of experience became a second nature to those experts. And as such, that knowledge transformed itself into an *instinct*.

The important point being that the doctors themselve didn't really have a full consience about this knowledge.

Now what does that make us learn about programming? If like me you are considered an expert, you have a lot of years of knowledge. You have internalized a lot of knowledge.

Debates

If you read about those questions you often endup with people with strong positive or negative opinions. But the problem is when you read the reasons behind their opinion. Most of these reasons are dressed as rational but under the hood, most of them are about passions.

The typical debates:

- Editors: vim vs emacs vs ...

- OS: linux vs macOS vs windows vs ...
- imperative vs OOP vs Functional Programming
- programming languages choices
- framework choices
- language syntaxes: C-like vs S-expressions vs Python-like, etc...
- space vs tabs vs smart-mix
- colorschemes: light vs dark, high vs low contrast, yellow vs blue, etc...

And the list goes on and on. I've read **a lot** of articles about all those subjects. And most of the time, my opinion was mostly forged on something. But as a geek, I tend to try a lot of new things (Editors, OS, programming languages, frameworks, etc...)

When I look back, I think I've got quite an experience with a lot of those choices now. And I finally ended up with some choices that I still consider the best. Most of time, I read a new article about "use X it's better than Y, Z, T, etc...". As a user of X if I feel the expressed plus-values are worth it. I generally give a try. And I am the kind of guy that switch easily to a new tool, new programming language, new environment if I get sufficiently plus-value. I am what is called an *early-adopter* in the Marketing space for most products I use[1].

Generally I'm passionate about something, and a few years after, i ever gave up, or I switch to a new tool.

If you take a look at my About me page you'll see the list of my current tech environment. So here are a few of my choices, and a short exerpt to why I use these. I also try to mention all things I tried before.

OS

I use macOS.

Why?

Mainly I consider that this is *unfortunately* the best OS today. I say unfortunately because I would prefer a lot that Linux would be a better fit for me. But I don't think this is the case yet.

Mainly why the best?

1. The hardware/OS is quite well integrated. So font quality, colors (with a very specific gamma), integrated well working retina display.
2. Focused environment. I pass very few time tweaking the OS. I open my computer and I can focus on my work.
3. A tremendous amount of very little design sugar everywhere makes your life a lot easier.

[1]Here is a pretty interresting talk by Gabriel Gonzales about marketing for progamming languages (here for Haskell). Because as developer we tend to ignore and even despise "Marketing". If this is your case this is probaby worth a watch: https://www.youtube.com/watch?v=fNpsgTIpODA

4. Integration with my phone is almost flawless. Sync my passwords, my photos, I can take call from my phone on my computer, etc...

As a developper I consider every update since Snow Leopard a bit of a regression. For every new feature, Apple give me a lot of anti-features. The most annoying ones being: forced to use App Store, impossible to remove OS Notifications, feel like an iPad/iPhone instead of like a real computer, security to only run signed software. I understand why they do this, but I'm a power user and I do not have enough control of my environement as I would like. This is why, at the second I feel I can make the switch to Linux, I will. But I don't think this is the case. In fact, I plan to give a try in a few days. Probably using a VM and see what it looks like.

Previous experiences (warning I'm old):

- window (up to 98)
- Linux Debian (up to 2002)

Programming Languages

I learnt a lot of programming languages. And really a lot more than probably necessary[2]:

- BASIC
- Logo
- Pascal
- C
- ADA
- C++
- Eiffel
- Java
- Objective-C
- PHP
- Python
- Ruby
- Awk
- Perl
- Javascript
- CamL
- Haskell
- Scheme
- Clojure
- Purescript
- Metapost
- zsh/bash/fish
- Prolog

[2]If you are curious I wrote about my biased opinion about all those languages in 2011.

I certainly forgot a few, I just listed the one that I either used a lot or had an impact on me.

After a few languages it is easier and easier to learn new ones. Mainly the concepts are always the same. Your brain start to see the *semantic* and slowly forget about the *syntaxic*.

And this is probably my biggest gripe against people judging programming languages both online and in real life.

> PhilipWadler's Law of Language Design:
>
> In any language design, the total time spent discussing a feature in this list is proportional to two raised to the power of its position.
>
> 1. Semantics
> 2. Syntax
> 3. Lexical syntax
> 4. Lexical syntax of comments
>
> In other words, twice as much time is spent discussing syntax than semantics, twice as much time is spent discussing lexical syntax than syntax, and twice as much time is spent discussing syntax of comments than lexical syntax.