The Languages That Bend Your Mind

Adrian Kosmaczewski

2022-09-16

Many programming languages have been sold to unsuspecting software developers with enticing descriptions, promising “transformative experiences” that “irrevocably alter their way of thinking” and other ethereal descriptions, seemingly belonging to other categories of products, such as yoga classes, religions, drugs, progressive rock albums, or role playing games.

Most of the languages advertised as such belong to the “functional” category; a moniker usually justified by their lack of “side effects,” which for most purists is a bad, bad thing.

The first one in the list of mind-bending languages is of course Lisp[1], carried on by its quintessential idea; that the language looks exactly like its primary data structure, and thus, as such, naturally lends itself to metaprogramming. The name of this wonderful property is “Homoiconic[2].” It is also the first language implementing Alonzo Church’s lambda calculus. Not a small feat. Richard Stallman, Robert Morris, and Paul Graham are the great apostles of this religion, spreading the word of the church, because it’s not just lisp service. Sorry, I had to write that one.

APL[3] is the next in the list. A language that consists of a mathematical notation able to parse and process vast amounts of data with the shortest of syntaxes, for which you need a special keyboard. A small but important community keeps this language alive, touting its capacities and eager to build the most complex programs with the least number of lines of code (usually just one,) maintainability be damned. It awkwardly[4] reminds me of another language.

Then followed Smalltalk[5]. The actual runtime where everything is an object, where everything is inspectable. Its most popular spin-off, called Ruby[6], took the crown thanks to Matz[7], _why[8], and DHH[9] in the 2000s, spearheading the birth of a large number of Web 2.0 successes: Twitter, GitHub, GitLab, Airbnb, Dribbble, Shopify, and many more. Alas, with a somewhat lackluster

[1]https://en.wikipedia.org/wiki/Lisp_(programming_language)
[2]https://www.expressionsofchange.org/dont-say-homoiconic/
[3]https://en.wikipedia.org/wiki/APL_(programming_language)
[4]https://en.wikipedia.org/wiki/AWK
[5]https://deprogrammaticaipsum.com/the-absolute-no-frills-quite-ignorant-very-incomplete-and-certainly-flawed-beginners-guide-to-smalltalk/
[6]https://en.wikipedia.org/wiki/Ruby_(programming_language)
[7]https://en.wikipedia.org/wiki/Yukihiro_Matsumoto
[8]https://whytheluckystiff.net/
[9]https://twitter.com/dhh

performance and fail whales[10], a problem that Crystal[11] and Elixir[12] are still trying to solve.

Another recent mind bending language (although it took 20 years to reach that status) is Haskell[13]. Miran Lipovača's book[14] clearly played a role in this case. The timing of this book (2009) couldn't have been better, given that many languages released during the 2010's claimed Haskell as one of their major inspirations. Another illustrious member of the functional programming family.

We could also add F#, who as the bread-winning member of the ML family gets a lot of (deserved) press for its usefulness and expression power. Apart from Don Syme, the creator of F#, arguably Scott Wlaschin[15] is the one stretching F# the most.

Using any of these languages, claim their pundits, is equivalent to a good dose of LSD, coupled with the vision of a thousand gods on top of a pyramid singing Dr. Alban's hit "Sing Hallelujah!"[16]

Some of these mind-bending languages have had notorious uses in finance, for example: most notably Smalltalk and F#. Lisp had its commercial hour of glory with Viaweb[17].

Of course, neither COBOL[18], nor Fortran[19], nor BASIC[20] and its `goto` statement[21], nor C[22] or C++ (although it sometimes does[23]), nor Java[24] or C#[25], nor Python[26], nor Go[27] or Rust, belong to the "mind-twisting" category. They are impure languages, as impure and as imperfect as the world they try to represent.

Those are the languages everyone complains about[28], as Stroustrup once said. The useful languages[29], as categorized by Perlis. The boring ones[30], as I call them. But maybe, just maybe, that's what explains their success in our industry and their simpler approachability; they embrace side effects and imperfection

[10]https://www.theatlantic.com/technology/archive/2015/01/the-story-behind-twitters-fail-whale/384313/
[11]/blog/crystal-is-a-surprise/
[12]/blog/elixir-and-phoenix-framework/
[13]https://en.wikipedia.org/wiki/Haskell
[14]http://www.learnyouahaskell.com/
[15]https://scottwlaschin.com/
[16]https://www.youtube.com/watch?v=YRqBcDwG8vs
[17]https://en.wikipedia.org/wiki/Viaweb
[18]/blog/hay-un-lenguaje-llamado-cobol/
[19]https://en.wikipedia.org/wiki/Fortran
[20]/blog/basic-standards/
[21]https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf
[22]/blog/pjsip-snippets/
[23]/blog/blow-your-mind/
[24]/blog/not-exactly-what-i-meant/
[25]/blog/quick-comparison-of-c-sharp-and-ruby/
[26]/blog/my-first-django-project/
[27]/blog/thoughts-about-googles-go-programming-language/
[28]https://www.goodreads.com/quotes/226225-there-are-only-two-kinds-of-languages-the-ones-people
[29]https://deprogrammaticaipsum.com/alan-perlis-and-the-evolution-of-programming-languages/
[30]/tags/boring-languages/

as natural features of our existence.

In any case, it'd be nice if at some point, as an industry, we could grow past this language fetish and move on.