Killer Apps

Adrian Kosmaczewski

2022-10-21

The D programming language lacked a "killer app" to break through[1]. Another brilliant language suffered from this situation, objectively deserving a much better fate than the one it had; Smalltalk[2].

Other programming languages have had more luck, and did have their "Killer Apps." These "Killer Apps" have taken various shapes across the ages. Some took the shape of other programming languages, some were operating systems, some consumer products, some were specific people, IDEs, open source projects, or other.

Let's look at some programming languages and their "Killer Apps." In this article we're going to concentrate on programming languages, leaving aside killer apps in other contexts, like for example VisiCalc[3] in the case of the rise of the Personal Computer, horizontal killer apps as described[4] by Joel Spolsky, and other cases. The list below is ordered *roughly* in chronological order of killer app introduction.

- COBOL: IBM System/360[5] and Mainframes.
- C: Unix.
- BASIC: microcomputers[6].
- Pascal: Turbo Pascal[7].
- ABAP: SAP.
- C++: Graphical User Interfaces (`Button` *is a* `Widget` and so on.)
- Visual Basic: Microsoft Windows and Microsoft Office.
- Perl: the World Wide Web and CGI scripts.
- Java: Netscape.
- VBScript: Active Server Pages[8].
- Python: NumPy[9] and, thanks to it, the Machine Learning craze of the 2010s.
- JavaScript: first Douglas Crockford[10], then XMLHttpRequest and AJAX,

[1] /blog/d-or-what-go-may-have-been/
[2] https://deprogrammaticaipsum.com/the-absolute-no-frills-quite-ignorant-very-incomplete-and-certainly-flawed-beginners-guide-to-smalltalk/
[3] https://en.wikipedia.org/wiki/VisiCalc
[4] https://www.joelonsoftware.com/2012/01/06/how-trello-is-different/
[5] https://en.wikipedia.org/wiki/IBM_System/360
[6] https://en.wikipedia.org/wiki/Microcomputer
[7] https://en.wikipedia.org/wiki/Turbo_Pascal
[8] https://en.wikipedia.org/wiki/Active_Server_Pages
[9] https://numpy.org/
[10] /blog/douglas-crockford/

then the V8 engine[11] that begat Node.js, and the rest is history.
- Lisp (well, functional programming in general): JavaScript.
- PHP: LAMP[12] and the Dot-com bubble[13] of 2000.
- C#: .NET.
- Lua: game programming.
- Ruby: Rails[14].
- Scala: Twitter[15] and its Failwhale[16].
- Objective-C: iPhone and App Store.
- Go: Cloud Native tools such as Docker[17], Prometheus[18], Kubernetes[19], and pretty much every project sponsored by the Cloud Native Computing Foundation[20].
- Erlang: WhatsApp[21].
- Elixir: Phoenix Framework[22].
- Haskell: Swift[23].
- TypeScript: Visual Studio Code and lately Deno[24].
- Kotlin: Android Studio[25].
- Dart[26]: Flutter.
- Rust: Linux Kernel (maybe?)

In each of the cases above, the existence of the object or event on the right propelled and boosted the use of the language on the left, and vice-versa. Without its "Killer App," the language would have not thrived and achieved the massive popularity and spread it enjoyed; and without the language, the "Killer App" on the right might not have existed at all.

Languages that lacked a killer app even though they deserved one: D, Smalltalk, Delphi, Ada (?), PL/I, and maybe Crystal[27] nowaways, too, although it's not too late yet.

---

[11] https://en.wikipedia.org/wiki/V8_(JavaScript_engine)
[12] https://en.wikipedia.org/wiki/LAMP_(software_bundle)
[13] https://en.wikipedia.org/wiki/Dot-com_bubble
[14] https://rubyonrails.org/
[15] https://www.artima.com/articles/twitter-on-scala
[16] https://www.theatlantic.com/technology/archive/2015/01/the-story-behind-twitters-fail-whale/384313/
[17] https://www.docker.com/
[18] https://prometheus.io/
[19] https://kubernetes.io/
[20] https://www.cncf.io/
[21] https://www.erlang-solutions.com/blog/20-years-of-open-source-erlang-openerlang-interview-with-anton-lavrik-from-whatsapp/
[22] /blog/elixir-and-phoenix-framework/
[23] https://academy.realm.io/posts/swift-summit-abizer-nasir-lessons-from-haskell/
[24] https://deno.land/
[25] Google decided to get rid of the Eclipse-based Android development toolkit (one of the wisest ideas in the history of software development) and asked JetBrains to provide Android Studio based on their IDE toolkit in 2014. This migration paved the way to JetBrains' own new language Kotlin to thrive, but there is another possible reason: given the tortuous lawsuit between Oracle and Google about the use of Java in Android, it is possible that allowing Kotlin to be the primary language for Android may have saved millions of dollars to Google. Who knows. On the other side of the fence, Google hiring JetBrains also left many iOS developers wishing Apple did the same, ditching Xcode for AppCode. *Le sigh.*
[26] /blog/dart-is-boring/
[27] /blog/crystal-is-a-surprise/

Interestingly, Dart and Flutter were conceived together; in a way, they mutually reinforced each other, one being the killer app of the other. This is exactly the same case as C with Unix, C# with .NET, Java with the JRE, or ABAP with SAP. The language and its framework mutually reinforcing one another. There's an interesting corollaire of this situation: *if you're creating a programming language, you might want to create with it its "Killer App" and release both at the same time.* Stephen O'Grady from Redmonk observed this in 2011[28], and it's spot on: Frameworks Lead Adoption[29].

This reinforcing mechanism once again shows how platforms[30] work, how they trigger and sustain their own growth past a critical threshold, and why the competition finds it so hard to fight against them.

It's also worth noting the explosive nature of a "Killer App" upon appearance; it can upset the dynamics of an established market almost overnight, catching everyone (including critics) completely off-guard, changing the course of (predicted) history in a snap.

Objective-C existed since the mid 1980s; the iPhone App Store appeared in 2008 and by 2013 Objective-C was the top language in the TIOBE rankings. Ruby existed for 15 years in relative obscurity until David Heinemeier Hansson used it to create Rails[31] in 2004. JavaScript had existed in browsers for 7 years until Crockford taught us[32] its true nature.

Programming languages can lay dormant for years or decades, and all of a sudden somebody figures out exactly what to do with it, and the world changes completely.

[28] https://redmonk.com/sograady/2011/04/27/frameworks-lead-adoption/

[29] The exception to the rule is probably Objective-C, who in spite of coming hand in hand with those wonders of NeXTSTEP and AppKit and Cocoa, had to wait for the iPhone to came along to get some popularity. And then Swift swept all of that away. *Le sigh, again.*

[30] https://deprogrammaticaipsum.com/geoffrey-g-parker-marshall-w-van-alstyne-sangeet-paul-choudary/

[31] https://www.youtube.com/watch?v=Gzj723LkRJY

[32] /blog/douglas-crockford/