Issue 044: Mobile

Graham Lee

May 2nd, 2022

Welcome to the forty-fourth issue of *De Programmatica Ipsum*, dedicated to the subject of *Mobile*. In this edition:

- Adrian argues that using cross-platform UI frameworks is a bad business decision[1].
- Graham discusses the current duopoly in the smartphone market[2].
- In the Library section[3], Adrian reviews the work of Erica Sadun[4].

This magazine denounces and abhors Russia's invasion of Ukraine. We urge Russia to withdraw its troops, and NATO to stop its expansion immediately. There must be an immediate reduction of nuclear weaponry worldwide, and we call on all sides to de-escalate and seek peaceful solutions. We express our deepest solidarity to all those campaigning for an end to the war, often under very difficult conditions, in Russia and Ukraine.

Enjoy this issue! Please subscribe to our free newsletter[5] to stay updated about new releases, share the articles on social media, or contribute[6] if you would like to support our work.

[1] https://deprogrammaticaipsum.com/the-law-of-diminishing-returns/
[2] https://deprogrammaticaipsum.com/on-the-duopoly-of-mobile/
[3] https://deprogrammaticaipsum.com/category/library/
[4] https://deprogrammaticaipsum.com/erica-sadun/
[5] https://deprogrammaticaipsum.com/newsletter/
[6] https://deprogrammaticaipsum.com/contribute/

Cover photo by Hardik Sharma[7] on Unsplash[8].

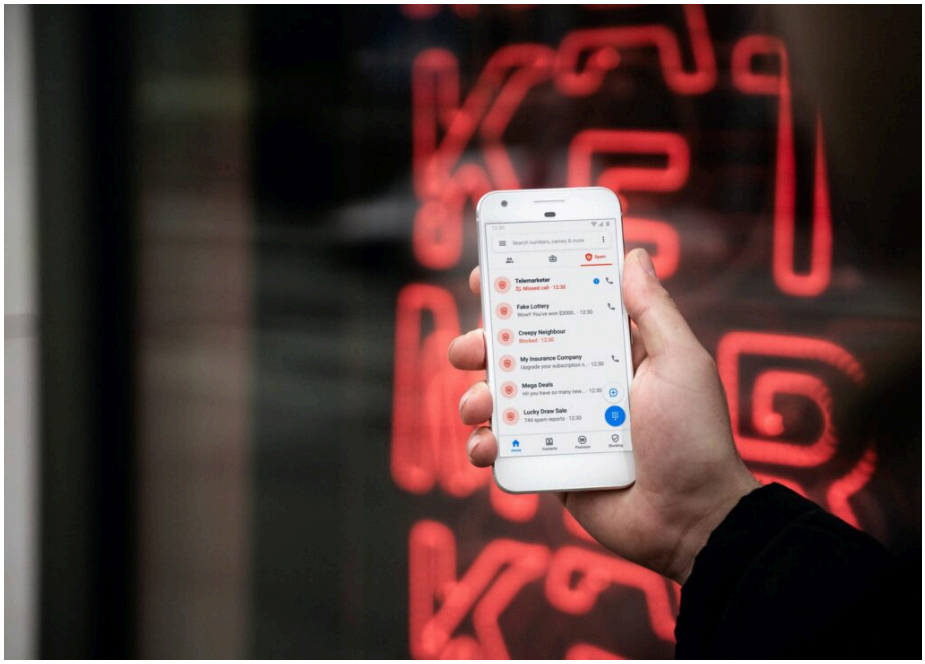
[7]https://unsplash.com/@v4ssu?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[8]https://unsplash.com/s/photos/ios-android?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

The Law Of Diminishing Returns

Adrian Kosmaczewski

May 2nd, 2022



The myth of the cross-platform[1] mobile application is as old as the iOS + Android duopoly. Back in 2008, a small Canadian company called Nitobi Software[2] released a JavaScript-based framework for cross-platform mobile applications called PhoneGap. Adobe snatched Nitobi in 2011, right after realizing that Steve Jobs was right[3], and that mobile apps based on Flash were nothing else than a chimera.

Adobe then took the commendable decision of donating PhoneGap to the Apache Foundation, becoming Apache Cordova[4] (and featuring a decidedly wrong spelling of both a Spanish and an Argentine city name), and Adobe PhoneGap became a distribution thereof (think Linux or Kubernetes). Adobe finally announced its demise[5] and shut it down[6] in 2020. *Dominus providebit, Dominus abstulit, sit nomen Domini benedictum.*

And PhoneGap was far from the only option available; this author enumerated many others[7] in October 2009. Interestingly, Java[8] is not a prominent item in the list.

At the time of this writing, the crapps[9] (cross-platform mobile apps) market is dominated by

[1] https://deprogrammaticaipsum.com/dr-dobbs-and-the-deathly-cross-platform-app/
[2] https://www.crunchbase.com/organization/nitobi-software
[3] https://en.wikipedia.org/wiki/Thoughts_on_Flash
[4] https://cordova.apache.org/
[5] https://helpx.adobe.com/experience-manager/kb/adobe-phonegap-end-of-service.html
[6] https://cordova.apache.org/announcements/2020/08/14/goodbye-phonegap.html
[7] https://akos.ma/blog/iphone-apps-without-objective-c/
[8] https://deprogrammaticaipsum.com/write-anywhere-run-once/
[9] https://www.urbandictionary.com/define.php?term=crapp

Xamarin[10] (C#), Ionic[11] (TypeScript), Dart[12] (Flutter), and by the biggest software development scam of all time, React Native[13].

There are two major problems with crapps developed with the tools enumerated above. The first is the most obvious one: the abysmal level of user experience provided; we are not going to dive into this issue right now.

The second problem of crapps, however, is a bit more subtle and of economic nature; and is as important as it is ignored. It has to do with the law of diminishing returns[14]. Said law

> ...defines a point on a production curve whereby producing an additional unit of output will result in a loss and is known as negative returns. Under diminishing returns, output remains positive, however productivity and efficiency decrease.

The hypothesis of this author is that, in the current duopoly situation[15], the cost of developing a crapp is actually higher than that of developing two apps using the officially sanctioned toolkits for iOS and Android apps.

You heard it right: *by using any of these fancy cross-platform mobile app frameworks you are spending more money, and achieving a less-than-optimal result.*

In other words, there is absolutely no economy of scale (or of any other kind) provided by using a cross-platform mobile app framework. Not even small economies. There is absolutely no advantage at all, whatsoever.

This bold claim is based on the personal experience of this author, who made a living by exclusively building mobile apps for those platforms from 2008 to 2019, and who also wrote two books about the subject of mobile apps based on web technologies.

Building a complicated[16], reasonable, interesting, stable, maintainable mobile app for iOS and Android is not just writing the code; there are many more factors that come into play. Let us mention just a few, those that are negatively impacted when working with cross-platform toolkits:

- Tightly monitoring battery consumption and memory usage;
- Integrating with hardware features, including wearable devices;
- Debugging, maintaining, and troubleshooting;
- Keeping up with new operating system features every year;
- Enabling widgets, and taking advantage of any kind of extensions;
- Offering non-cross platform compatible "native" features;
- ... and many more.

Cross-platform mobile app frameworks *kind of* made sense during the short time when Windows Phone existed. In those times (and with careful design), adding a third build product for Windows Phone incurred a relatively low overhead compared with the rest of the development cost, *but only* for small, simple apps.

[10] https://dotnet.microsoft.com/en-us/apps/xamarin
[11] https://ionicframework.com/
[12] https://dart.dev/
[13] https://reactnative.dev/
[14] https://en.wikipedia.org/wiki/Diminishing_returns
[15] https://deprogrammaticaipsum.com/on-the-duopoly-of-mobile/
[16] https://deprogrammaticaipsum.com/complex-vs-complicated/

The reasons why teams choose cross-platform toolkits for their projects stems less from a technical standpoint than with the lack of unionization[17] in the software industry. It is a simple way to business owners to reuse manpower from one project to the other, making a false economy and leading software developers to burnout[18] and abuse[19].

Of course, all of the platforms enumerated above include some kind of "plugin system" enabling access to some iOS or Android specific features. The biggest problem with these plugins goes beyond maintainability (since most of them are not provided by the original framework developers); it is rather a conceptual and (again) economic issue.

This is the gist: the mere existence of those plugins highlights the fact that iOS and Android offer *quite different* user experiences and features, effectively killing the promise of those frameworks. This is a subtle issue, not considered by developers blinded by the quick "Hello World" experience shown in the tutorial.

As said above, this is a subtle issue; it is also, sadly, a massive one. The level of technical debt introduced by these frameworks render them useless for large projects, and can effectively sink whole companies.

Bluntly speaking, using a cross-platform mobile app framework is a great way to solve bugs in one platform, while inadvertently adding some more in the other.

In other words, they are another, spectacular example of a leaky abstraction[20].

> All non-trivial abstractions, to some degree, are leaky.

As a software developer, I can only recommend using tested, boring, stable technologies for your mobile apps (or really, for any kind of app). Use Kotlin with Android Studio for your Android app. Use Xcode and Swift for your iOS application (but not SwiftUI[21], at least not yet at the time of this writing). Build two different apps, talking to the same REST backend; and push logic to that layer, so that you can reuse it where it makes sense. Use native components following the user experience guidelines set forth by Google and Apple. Make both apps eat as little battery as possible, and make them use as little memory as possible. Make each of them fast, using the tools provided by each vendor.

Apply the best ideas and the best patterns for each platform, separately.

And if you ever need to share code between your iOS and Android apps, you can always use C++; which was the solution adopted by Dropbox[22] for almost a decade. You can encapsulate algorithms and data structures in a native library built with C++, and then make your Android and iOS app use that code. Complex? Yes. Based on boring and mature tools? You bet. Yielding really fast code? You have no idea.

Building separate apps for each platform means lower maintenance costs in the long run, and an optimal user experience everywhere.

Because, let us be frank; it does not seem like the duopoly will be over anytime soon. There will be Android and iOS for quite a while, and there are no viable competitors in the horizon from an economic point of view.

---

[17] https://deprogrammaticaipsum.com/divide-et-impera/
[18] https://deprogrammaticaipsum.com/business-as-unusual/
[19] https://deprogrammaticaipsum.com/learnings-from-toxic-abuse/
[20] https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/
[21] https://developer.apple.com/xcode/swiftui/
[22] https://oleb.net/blog/2014/05/how-dropbox-uses-cplusplus-cross-platform-development/

Cover photo by Lindsey LaMont[23] on Unsplash[24].

[23]https://unsplash.com/@travelpen?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[24]https://unsplash.com/s/photos/scam?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

On The Duopoly Of Mobile

Graham Lee

May 2nd, 2022

It is surprising how quickly the duopoly of iOS and Android, Apple and Google, became entrenched. At the 2007 iPhone launch event[1], Steve Jobs compared the touchscreen-centric iPhone UI with four other competitors: the Moto Q, BlackBerry, Palm Treo, and Nokia E62. The Moto Q ran on Windows Mobile, the last release of which was in 2010. BlackBerry had replatformed onto Android in 2015, joining the duopoly. Palm effectively stopped making anything (even at its new home in HP) in 2011, then popped up again in 2018 making Android devices. The holdout was Nokia, who partnered with Microsoft in 2011, sold to them in 2014, and (as Microsoft Mobile) closed in 2018.

Within little more than a decade, all competition had disappeared. Apple targeted 1% of the global mobile phone market at the 2007 launch. Today, they control 25% of the global market, with Android at 70%. The remaining 5%...well, who are they?

It depends on how you count. True alternatives include Tizen, Sailfish OS, Firefox OS/boot 2 gecko...they appear and disappear without leaving a mark. I worked at one of the multinational telcos back in 2012, and already (five years in to this journey) they were willing to throw their money at anyone who promised to be a third choice in this two-horse race, putting a massive marketing budget behind Windows Phone and even seconding members of my engineering team to Mozilla to work on Firefox OS. It did not stick. Then there are the budget feature phones sold in developing markets and to Dutch hipsters[2]. They are a doubly-small

[1] https://www.youtube.com/watch?v=MnrJzXM7a6o
[2] https://johnsphones.wordpress.com

contribution: low numbers, low value.

We are technologists, so we might count among the alternatives those that are based on the same technology as the others but that represent different markets. "Ubuntu for Android" is Ubuntu, but with the Android kernel and device drivers. It is so dead that Canonical have deleted the website. We can say the same for the Amazon Fire Phone, which (like the Kindle Fire tablet) is Android but with Amazon's stuff instead of Google's.

One that still exists is Replicant[3]: a fully free build of the Android Open Source Project plus other components. Replicant is not Android in that it is not Google and does not have the Google Play Services, and Google do not make any money when you install Replicant. But does it truly compete? Does a five-year old device with a free download of a fork of Android 6 and a copy of the F-Droid store "count" as a part of the mobile phone market? What about a rooted iPhone with Cydia, does that compete with Apple? Or does it actually *complement* Apple, helping them to sell iPhones to people who are averse to walled-garden, console like experiences?

What about all of the things that are truly mobile computers, but that do not apparently count as mobile devices in the market share? What would it look like if we included portable Windows 11 computers that have a tablet mode alongside the other mobile platforms? Macs that can run iPad mobile applications, or Chromebooks that can run Android applications? Why are not these "mobile devices"?

OK, so the marketplace is messy, and things are counted in weird ways, but by and large there genuinely are two clear leaders: Android is far and away the majority choice for customers, iOS comes a respectable second, and everything else is so small that it is hard to know whether they are even being counted. What would it take to break that?

Evidently some people think it is possible, otherwise these wannabe third-place vendors that come and go would stop coming. But come they do: one of the latest is PureOS[4]. What do they need to do? We can answer that by looking at the headline of the PureOS website: "A fully-convergent, user friendly, secure and freedom respecting OS for your daily usage." Notice that the first thing they mention, before freedom-respecting, secure, or even user-friendly, is *fully convergent*.

Convergence[5] was, if you go back and re-watch that video from the iPhone launch linked above, the big selling point of the iPhone. It is a media player, an internet browser, and a mobile phone all in one. And these days, it is a camera, a satellite navigation system, and a payment card too. So it is not enough to make a good phone, you have to make a good all-of-those-things.

But actually in 2022 it is much more than that. Convergence has flipped inside out: it does not just mean "I only need one box of electronics in my pocket", it means "my thing works with all of my other things". Google, Apple, even Microsoft: none of these are in the gadget-selling business or even the software-for-gadget-selling business any more; they are all in the "making your electronic stuff work wherever you are and whatever you are up to" business. So your new competitor needs to not only replace my phone, but that phone needs to work well with my smart watch, my smart TV, my laptop. And there is a problem, because all of those things are within the current ecosystems, so your new thing becomes an accessory, not a replacement.

[3] https://www.replicant.us/
[4] https://pureos.net
[5] https://deprogrammaticaipsum.com/plus-ca-change/

That is why all of the supposed new platforms that will disrupt this space are not actually disrupting this space. Let us say for a second I believed that dApps—so called "Web 3.0" applications based on blockchain technology—were clearly going to be the future of the internet. Does that mean that I believe that any one or more of the companies operating in that market will eclipse Apple or Google as the go-to supplier for electronic gizmos? No. Because to take over the planet, they have to address the current planet: and the current planet is on Android and iOS. They just drive new applications of the existing, entrenched platforms. Same goes for VR goggles: Meta is not going to supplant Apple or Google by being the goofy headgear vendor of choice, even if they do manage to corner the goofy headgear market.

The people who understand this dynamic and want to try to beat it anyway are doing the trick Indiana Jones did with the idol and the bag of sand: they are giving you a new way to use your existing thing that you get familiar with until one day, "boom", you can do away with the existing thing and just go straight to them. See, for example, FriendOS[6].

There is in fact a pretty sweet solution to overturning this duopoly, though its advance seems to have stagnated. Plenty of the applications you actually use on your mobile phone are in fact web apps, whether you launch your browser to interact with them, install them as a Progressive Web App[7], or even download a native bundle of Kotlin or Swift from an App Store. They are just little blobs of code that do some authentication to a web service, then send and receive JSON documents to render on the screen. This puts iOS and Android in danger of being the "dumb pipes"[8] that the telcos have discovered they are supplying: all you get when you unwrap your iGalaxy Zperia S220 is a SIM card holder with a big button marked "browser" and somewhere to enter your wi-fi password.

Then the next replacement is simply the vendor who supplies the "pocket browser plus"—any old web app works, any old React Native or Ionic app works, *and* you can do whatever it is that is unique to this platform and highly desirable. Google and Apple have to catch up—they have bottomless pockets so they will do so, but they will have to license the new IP to be allowed to do so, meaning the new vendor gets rich anyway. Wherever that new thing comes from, I expect it will be what Clayton Christensen called a disruptive innovation in his Innovator's Dilemma[9]: nobody will see it as world-changing, *including* the vendors, until it has already changed the world.

Cover photo by BP Miller[10] on Unsplash[11].

[6] https://friendos.com

[7] https://docs.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/

[8] https://en.wikipedia.org/wiki/Dumb_pipe

[9] https://www.stratrix.com/innovators-dilemma/

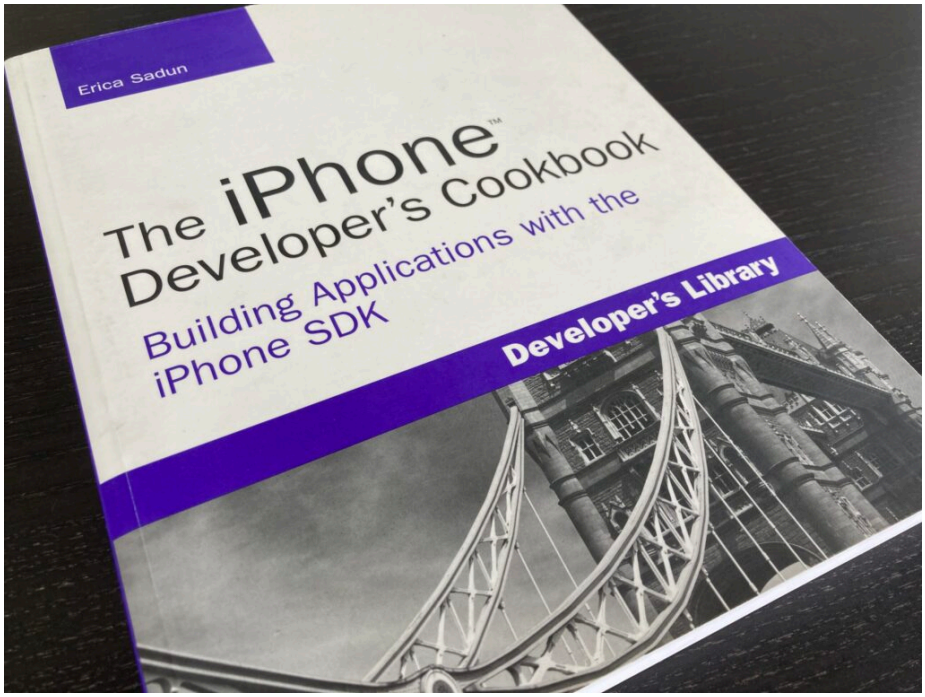
[10] https://unsplash.com/es/@bp_miller?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[11] https://unsplash.com/s/photos/monopoly?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Erica Sadun

Adrian Kosmaczewski

May 2nd, 2022



iOS developers new to the platform are completely (and thankfully) unaware of its rocky start during its initial years. The first iPhone was announced on January 9th, 2007, and was released in the United States on June 29th that year. The iPhone SDK was announced[1] by Steve Jobs in October 2007, and released in March 2008. But even before the official SDK was first announced, people were already "jailbreaking" the device, and thereby making applications for the iPhone. First-generation iPhone and iPad developers will surely chuckle when reading the words "PwnageTool," "JailbreakMe," and the name of the first App Store, also known as "Cydia."

I will wait until you stop laughing (or crying).

These were the times of Justine Ezarik opening a box[2] containing the 300 pages of her first AT&T bill. Of Joe Hewitt[3] releasing the first iteration of the iUI framework. Of Steve Ballmer just laughing[4] at the iPhone. Of James Duncan Davidson[5]'s iconic photo of the iPhone[6] inside a glass screen.

The iPhone SDK and the App Store did not come alone, though. They were accompanied by

[1] https://arstechnica.com/gadgets/2007/10/apple-to-open-iphone-ipod-touch-to-third-party-developers-in-early-2008/
[2] https://www.youtube.com/watch?v=UdULhkh6yeA
[3] https://en.wikipedia.org/wiki/Joe_Hewitt_(programmer)
[4] https://www.youtube.com/watch?v=eywi0h_Y5_U
[5] https://twitter.com/duncan
[6] https://www.yatzer.com/duncan-davidson-beauty-detail/slideshow/6

an infamous NDA[7] that lasted until October 2008. Said NDA prevented developers from discussing in public about the iPhone SDK features; among other things, this meant that nobody could ask questions about it, not even in this new website called "Stack Overflow" that started operating in September 15th, 2008.

On November 21st, the first edition of "Beginning iPhone Development"[8] by Jeff LaMarche and Dave Mark went on sale and became an instant hit; this one is often mentioned as the first published book explaining the iPhone SDK. Well, not really. A few weeks before that date, on October 13th, 2008, probably the same day the NDA was dropped, Erica Sadun's first magnum opus hit the shelves: The iPhone's Developer's Cookbook[9].

Erica Sadun was already well known in the jailbreak online community, patiently and painstakingly dissecting every new release of the iPhone OS, and dumping class headers for all developers to use in their own applications. She was referred to by Engadget[10] as "one of the soldiers heading up the fight to break Apple's stranglehold." Such a title carries the sentiment the developer community regarding the close nature of the iPhone; and which would play a tremendous role in boosting Android as a developer-friendly alternative, in the period from 2009 to 2012.

Erica has stated in an interview[11] that in this book she tried to write "the kind of reference I wish I had when starting out." One of the most interesting aspects of the book is that it navigates, not without controversy, the rough, uncharted waters of the private APIs; those that no developer is allowed to touch for their code to be allowed to hit the virtual shelves of the App Store.

And in this book, well, we get a glimpse of various of those. Let us mention some honorable examples: the hidden HTML editing support in the `UITextView` class in page 253; the `UICalloutView` API explained in page 258; the instructions to add text input fields to an `UIAlertView` in page 113; all of the undocumented `CATransition` animation types in page 67; a complete chapter about `UICoverFlowLayer` at the end of the book; and, yes, a warning against using all of this knowledge in applications targeting the App Store, very early on page 34.

Apple longtime loyalist John Gruber was understandably aghast and vocal[12], going ballistic against this book and its contents. Thankfully, Erica Sadun has kept working on the platform[13], to the delight of the rest of the community, in spite of such attacks; even becoming an important voice of the Swift community, after the language was released in June 2014. Fifteen years later, the App Store policies are still a matter of major debate and controversy, and this, even at the highest levels of the European Commission, through the Digital Markets Act[14].

As a personal note, unusual in this magazine, I would like to send a warm salutation to Erica, who helped boost my activity as an independent iPhone developer, by promoting my

[7] https://www.wired.com/2008/08/iphone-coders-feel-miffed-muzzled-by-apple-s-nda/

[8] https://www.amazon.com/Beginning-iPhone-Development-Exploring-SDK/dp/1430216263/

[9] https://www.informit.com/store/iphone-developers-cookbook-building-applications-with-9780321555458

[10] https://www.engadget.com/2007-10-08-iphone-v1-1-1-firmware-gets-the-jailbreak-treatment.html

[11] https://www.informit.com/articles/article.aspx?p=1184557

[12] https://daringfireball.net/2008/12/private

[13] https://ericasadun.com/

[14] https://en.wikipedia.org/wiki/Digital_Markets_Act

obscure nib2objc[15] tool through an article she wrote in ArsTechnica in April 2009[16], and even mentioning it in later editions of her book. The surprise and reach of this article is still one of the highlights and major prides in my career.

Cover photo by the author.

[15] https://github.com/akosma/nib2objc
[16] https://arstechnica.com/gadgets/2009/04/iphone-dev-convert-xib-files-to-objective-c/