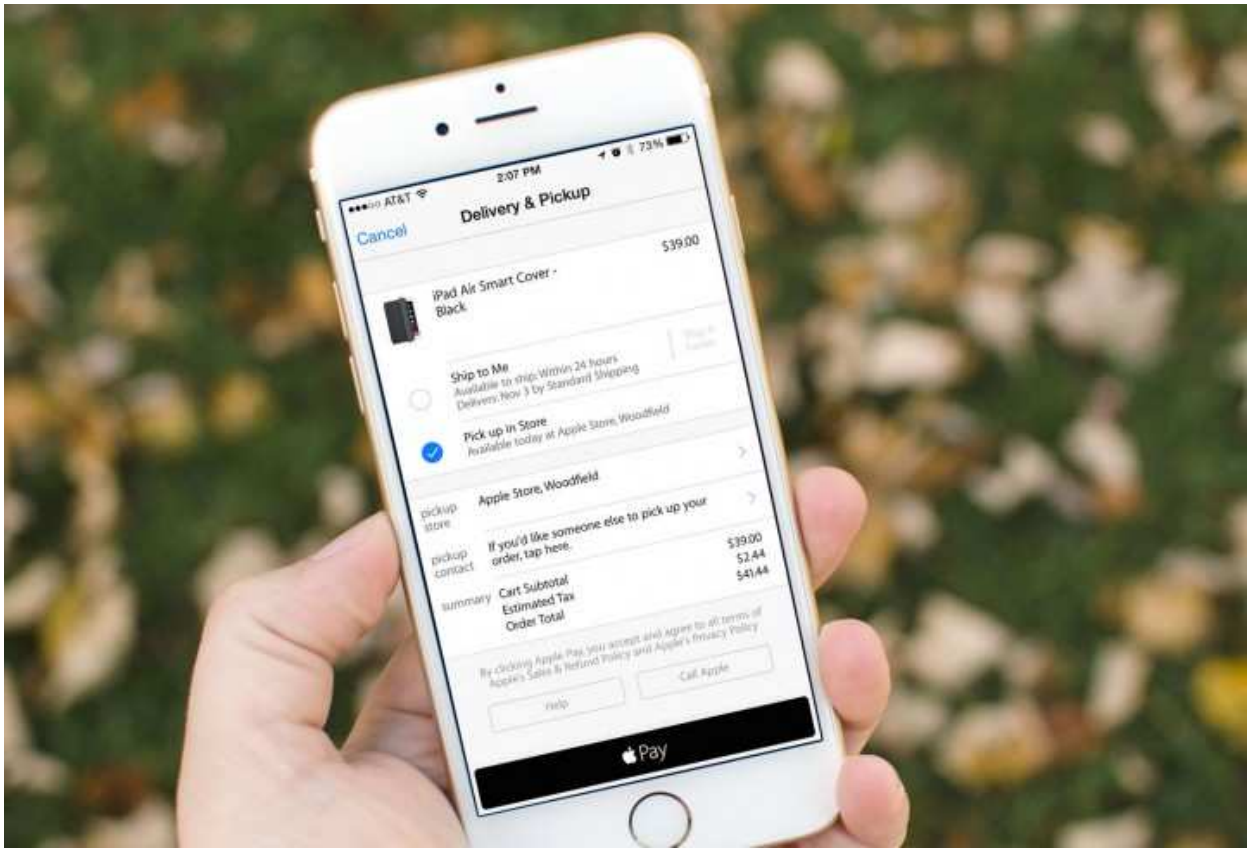
“Apple’s Science and Technique”

Apple Pay

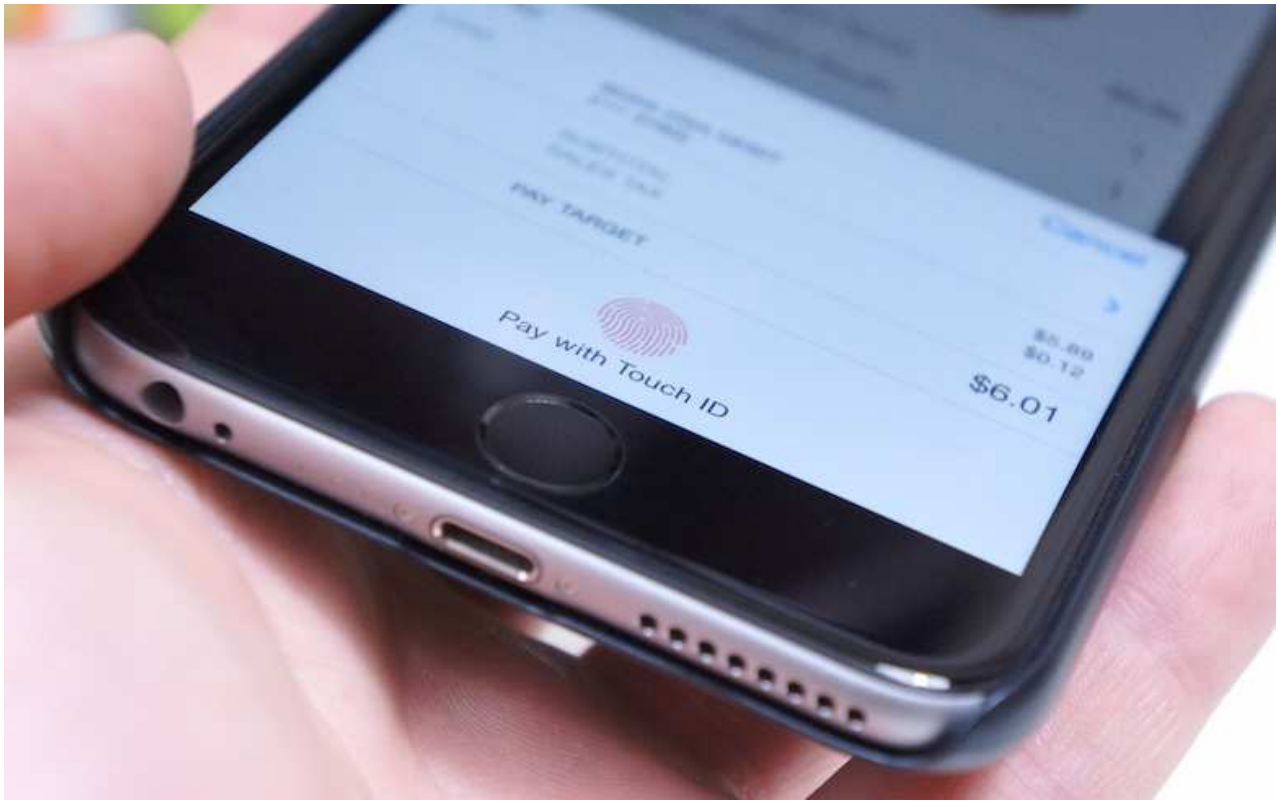


On iOS, mobile e-commerce development has long been augmented by services like PayPal, which provide mobile payment processing that spans mobile web and mobile native applications with mobile user authentication and payment processing. But with iPhone 6 Apple introduced a new approach for payments within iOS native applications: in-app Apple Pay.

Apple Pay in apps could reshape how e-commerce brands approach iOS native development.

Apple Pay isn’t only an implementation of the iPhone 6 NFC chip for retail payments. It’s also a broad-ranging payments system that leverages the enhances security features of Touch ID to process secure payments in iOS apps. Native developers can include Apple Pay as a payment option in their apps alongside traditional credit card entry or even PayPal sign-in, drawing from the saved payment card data stored in iPhone’s secure element.

But Apple Pay in apps is more than a shortcut to payment cards from within native applications, or even a novel way to make use of Touch ID.

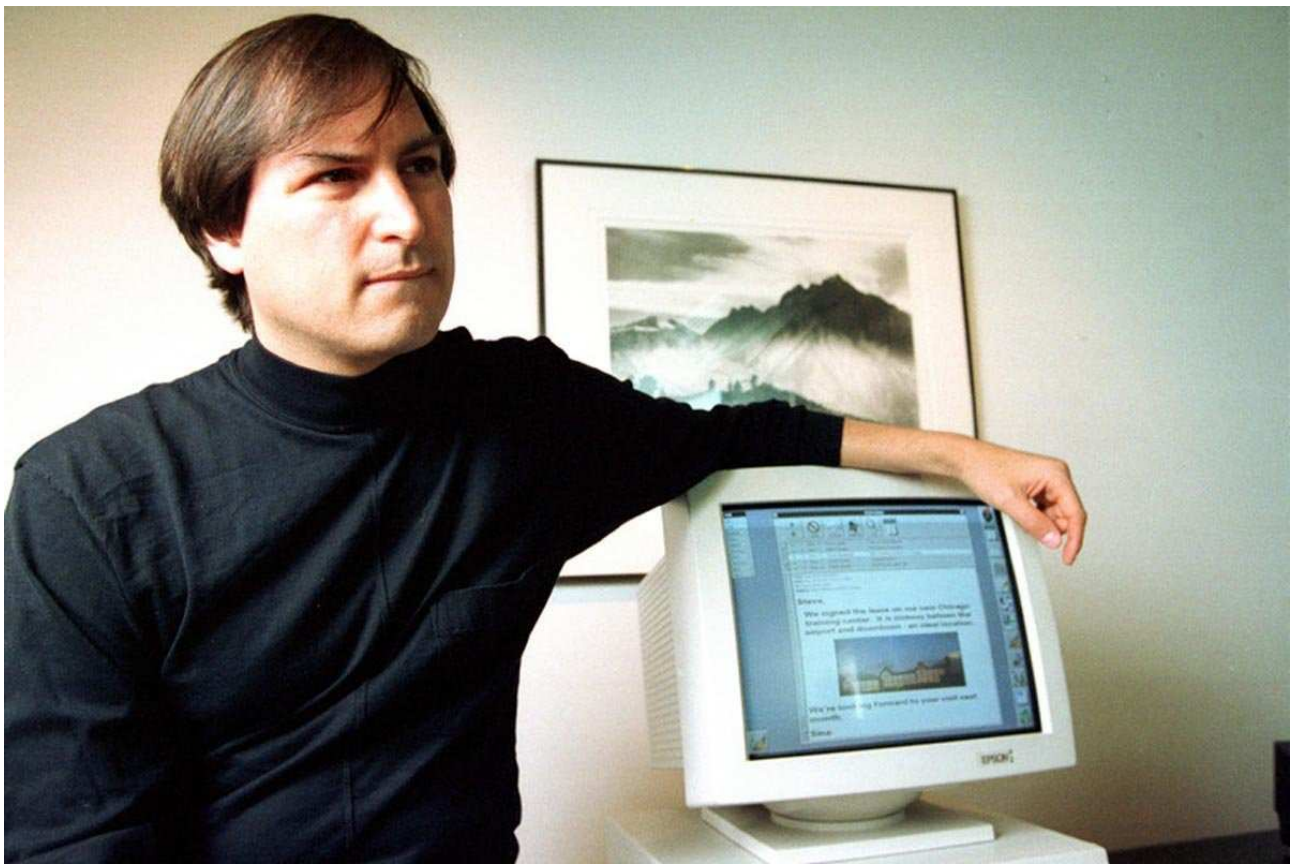


Apple Pay can simplify the registration workflow, getting users through sign-in screens and into transactions quicker.

Mobile app developers can use the data provided by Apple Pay to instantly register accounts and get users paying quicker. Uber, an in-app Apple Pay launch partner, uses the Apple Pay data to immediately authenticate users and help them begin hailing cabs in seconds. Rather than using the traditional email-and-password registration process, and the lengthy payment card entry screen, users simply scan their fingertip with Touch ID, and an Uber driver is en route.

Simplifying payment processes for e-commerce transactions is valuable for online retail brands, but this speed-to-checkout improvement is massively critical for mobile e-commerce developers.

SWIFT



The story of Swift begins with Steve Jobs‘s second company, NeXT. In 1983, Brad Cox and Tom Love introduced a modified version of the longstanding C programming language called Objective-C. The new language incorporated object-oriented programming, which handles conceptual “objects” that can have various attributes, and was designed to coexist with legacy C code. When Steve Jobs founded NeXT during his exile from Apple, the company designed its forward-thinking NeXTSTEP operating system to run apps built in Objective-C, which ignited developer interest in the language as they explored the exciting new platform.

When Apple acquired NeXT in 1996, this commitment to Objective-C programming infected the Mac OS platform, as well. As NeXTSTEP was transformed to replace the legacy Mac operating system with what would be known as Mac OS X, Objective-C became the foundation for Apple’s revolutionary new direction. The Project Builder developer tool became Xcode, and a new generation of Mac developers were exposed to Objective-C as they prepared apps for Steve Jobs’s Mac OS X. Over the years, Apple invested in their operating system with Objective-C, introducing a number of APIs and frameworks built in the language. When the company announced the iPhone in 2007, its fledgling operating system—then called iPhone OS—was an evolution of OS X with the same Objective-C capabilities.

Because of the massive growth in popularity for iOS products, Objective-C has enjoyed widespread adoption and use among the developer community in the last few years. Any of the

millions of iPhone and iPad apps populating the App Store is built in Objective-C, and each has a team of hardworking developers who code in the language all day. But it has never been perfect—a programming language designed in 1983 couldn't possibly account for modern challenges, no matter how many helpful developer tools are built on top of it. That's why developers were so thrilled when Apple surprised them with the news that it had developed an entirely new programming language, billed as "Objective-C without the C." It's called Swift, and it's going to change the way millions of developers approach iOS and OS X.

A lot has changed in the 31 years since Objective-C was first designed, and Swift is a play to bring OS X and iOS up to date and into the future. Although Apple has added substantial new changes to Objective-C over the years, like blocks and synthesized properties, there were only so many updates it could before beginning to move on completely. Apple created Swift with an awareness of modern programming languages, offering clean, easy-to-read syntax and simplified organization features to eliminate the need for .h header files.

Apple designed Swift to coexist peacefully with existing C or Objective-C code. Swift shares the same runtime with these older languages, so both can run in tandem—even within the same app. This means that developers with an existing Objective-C codebase can begin using Swift in their iOS 8 or OS X 10.10 applications right away, easing the transition between languages.

More important than elegance is ease of use—Swift is designed to make many longstanding Objective-C programming hang-ups nightmares of the past. Swift checks developers' work momently, initializing variables before use and checking arrays and integers for overflow.

Another example is the inclusion of automatic reference counting in Swift versus a tedious developer process in Objective-C. In OS X 10.5 Leopard and iOS 4, Apple introduced automatic reference counting, or ARC, to help developers keep track of complex "retain" and "release" calls throughout their code. The addition made programming in iOS and OS X easier, but Swift takes ARC even further.

Because of Apple's extensive experience writing for their own platforms, its developers intimately understand the more frustrating issues they'd run into.

The biggest obstacle facing any new language, spoken or coded, is attaining fluent speakers. With their new language, Apple emphasized easy learnability to ensure a low barrier to entry and encourage swift adoption. The syntax was designed with legibility in mind, similar to human-friendly languages Python and Ruby, to help newcomers and veterans alike better understand how existing codebases operate with as shallow a learning curve as possible.

The new version of Xcode was specifically designed to aid developers as they began exploring Swift's capabilities. Built-in tools called Playgrounds allow developers to view their code's impact in real time, so they can better understand their creations as they physically type. A new overlay

shows a live preview of an app view that transforms in real time as new lines of code are added, introducing a degree of interactivity and play to the Swift programming experience.

A lot of senior developers have said that Swift is similar to Scala, which puts it closer to Python and Ruby in terms of readability. It's not as explicit or verbose as Objective-C, but the readability means that more people are going to feel comfortable using the language. This instant accessibility, combined with Apple's uniquely broad and engaged developer audience, could drive Swift adoption among newcomers and longstanding iOS developers alike. But the switch to Swift won't happen overnight. Because it's a complete overhaul of the language. Especially because so many iOS apps out there have such a huge codebase, they can't be rewritten overnight." One factor aiding a gradual adoption is Swift's ability to coexist with C and Objective-C within the same app, but writing new Swift apps from scratch are the next logical step of iOS developers. Just as Apple phased out the floppy disk and iteratively eliminated CD and DVD drives from modern Macs, developers anticipate that Apple might have similar plans for the longstanding Objective-C language.

Because the Swift programming language will not be cross-compatible with non-Apple platforms like Android and Windows Phone, software engineers who specialize in Swift will be locked into iOS development. (The same goes for iOS applications written in Swift, but apps written for iOS and Android today won't be cross-compatible, either.) In future versions of OS X and iOS, Apple might invest in Swift-specific versions of Cocoa and Cocoa Touch, collections of frameworks like Core Animation and UIKit that allow apps to leverage the operating systems' various capabilities. In addition, future frameworks for theoretical capabilities might only be added to the Swift-oriented frameworks, meaning developers would need to reform their apps in Swift to take advantage of exciting new features. This could lock out third-party applications dependent on today's frameworks, which are all built on Objective-C.

But Swift could have the larger impact of allowing developers to release better applications on iOS than those available on Android with no additional work. With Swift's failsafes to prevent common programming oversights, applications for iOS written in Swift can be on average more stable, more reliable, and offer a better overall customer experience. Beyond reliability, Swift could usher in a new generation of apps written uniquely for the unique language. The matter of iOS app development has more pressing implications for consumers than the complex subject matter might imply. The adoption and evolution of Swift could have a serious impact on the story of iOS's future, and could inspire new ideas among the thousands of iOS developers who start learning it and deploying it. Now that Apple has a programming language and two computing platforms it completely controls, the company can begin iterating with new features and unprecedented capabilities with greater acceleration. Years from now, when Apple introduces new whiz-bang features that delight its customers on future iPhones, iPads, and Macs, many of those capabilities will be built on the foundation set by Swift. And the new crop of iOS programmers who get their start on Swift might grow to become the next generation of Zuckerbergs and Wozniaks.

Reference

1) https://developer.apple.com

2) https://www.punchkickinteractive.com/blog/2014/09/04/ios-8-in-context-healthkit
3) https://www.punchkickinteractive.com/blog/2014/09/30/ios-8-in-context-swift-is-the-language-of-apples-future
4) https://www.punchkickinteractive.com/blog/2015/02/04/apple-pay-reinvents-mobile-e-commerce-for-ios-apps-why-brands-should-jump-in