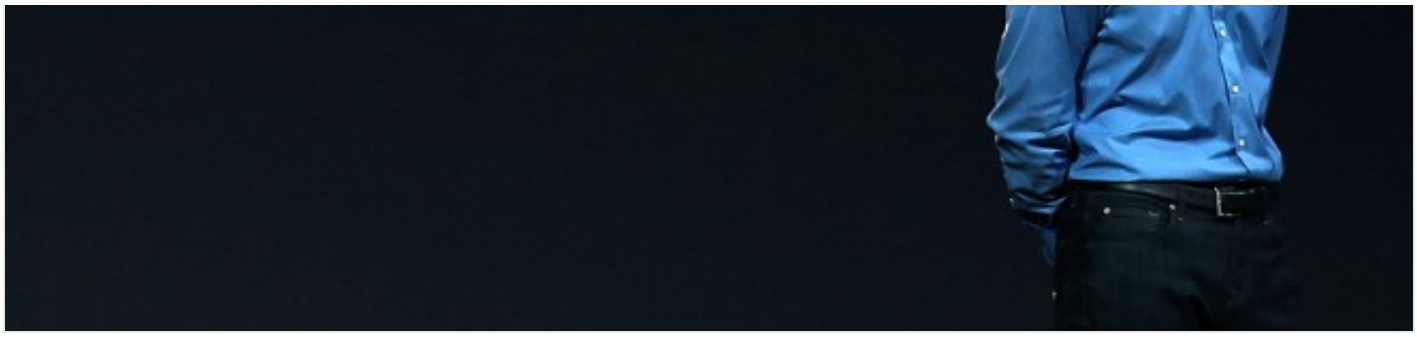
Why Coders Are Going Nuts Over Apple's New Programming Language



Craig Federighi introduces Swift during the Apple Worldwide Developers Conference. *Photo: Justin Sullivan/Getty Images*

When Apple unveiled a new programming language at its World Wide Developers Conference on Monday, the place went "nuts," erupting with raucous cheers and applause. It was the coding-world equivalent of Oprah giving away all those free cars.

WWDC is a gathering of people who build software applications for Apple hardware devices—from the iPhone and the iPad to the Mac—and with its new language, dubbed Swift, Apple is apparently providing a much faster and more effective means of doing so, significantly improving on its current language of choice, Objective-C. With something that Apple calls an "interactive playground," Swift is even exploring a highly visual kind of programming that may go beyond other mainstream languages. All those developers went nuts not only because they love Apple, but because the new language could make their lives that much easier.

If it lives up to Apple's billing, Swift may also allow a whole new type of coder to build applications for devices running the iOS and Mac OS X operating systems. "It could lower the barrier to entry for Apple developers," says Caylan Larson, an iOS and Mac OS developer based Winona, Minnesota who watched the WWDC keynote online and is already poring over the new guide that details the Swift language. "It could open a lot of new doors for a lot of people."

'Swift could lower the barrier

for entry for Apple developers. It could open a lot of new doors for a lot of people.'

But there's a flip side. Even though Swift could ultimately ease the process of building apps for Apple hardware, existing developers like Larson must first take the extra time needed to learn the new paradigm. "Do I put all my projects on hold while I pick this thing up?" he says. "It's a balancing act." Swift may look like the future, but the world is littered with programming languages that promised to make life easier for developers before ultimately fading into obscurity because people just didn't want to deal with something new.

What's more, Swift seems to extend Apple's split with the rest of the software development universe. Many coders would prefer that Apple shift toward tools that would also let them build software for other machines from other vendors. But Tim Cook and company are traveling in the opposite direction. "Swift has all the right check boxes, but do we really need something that's proprietary to Apple's platform?" asks programming guru David Pollak. "Yes, it solves a lot problems, but it's yet another way to drive a wedge between iOS development and everything else."

This is unlikely to harm Apple any time soon. In fact, the company prefers things this way. It insists on defining its own rules, and its devices are so widely used, it knows that large numbers of developers will happily build apps for them no matter what language this requires, driven by the enormous dollars signs they see in names like the iPhone and the Mac. But life would be even easier for these developers if they could build applications that instantly ran on all devices, from Android phones to iPads. We were already a long way from that, and now Apple has taken us even further away.

Beyond Objective-C

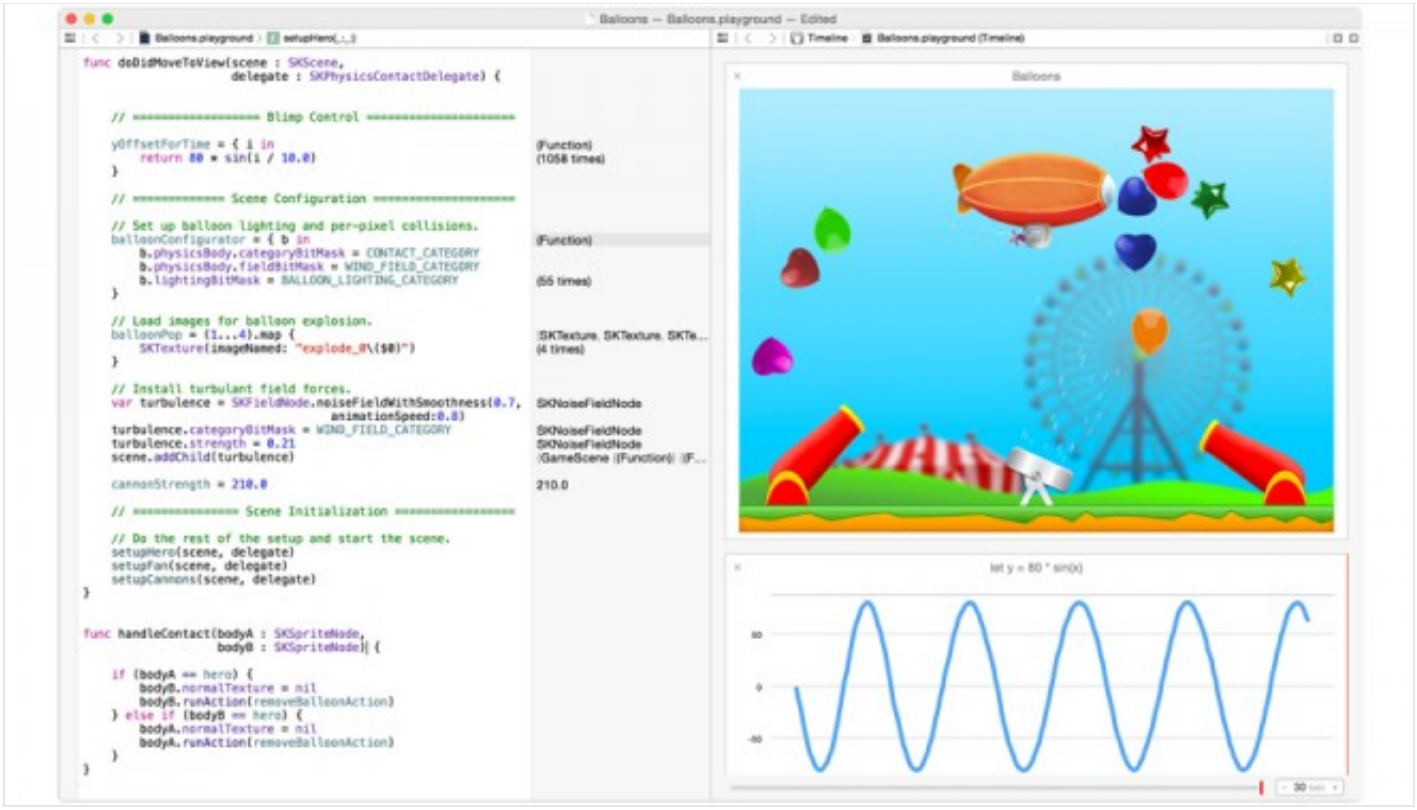
Today, most all Apple software is built with Objective-C. Originally designed in the

1980s and soon adopted by Next Computer–the Steve Jobs-led company whose technologies eventually morphed into the modern Mac and the iPhone–the language has ridden these devices to enormous popularity.[1] Based on the venerable C language–still the world's most widely used–it feels familiar to many coders, and it lets you build code that runs rather fast.

But in many ways, Objective-C isn't nearly as easy to use as more modern languages such as Ruby and Python. Generally, such "interpreted languages" let you develop software much quicker, handling many tasks in the background that you would otherwise have to handle on your own. Swift appears to change this.

For both Larson and Pollak, one of Swift's biggest advantages over Objective-C is that it does what's called "automatic garbage collection." Basically, this means that it will automatically dispose of unneeded information that's sitting in a machine's memory, and the result is that developers won't have to spend a lot of time and energy trying to deal with memory management on their own. This is the primary reason that Larson believes Swift can reach a much larger number of developers.

Swift also offers "inferred typing," which means that developers don't have to spend as much time defining what types of variables they're using. And the new language allows for concise "closures," which basically means that it's easier to build a small piece of code that can repeatedly collect information on behalf of your program. As with automatic garbage collection, both of these additions mean that Swift will "feel more modern," that it will let developers build reliable software more quickly and with less hassle.



The Swift "interactive playground." *Image: Apple*

The Interactive Playground

On top of this, Apple has added the "interactive playground," a way of viewing the results of a piece of code as you type it. Pollak says it reminds him of the work that ex-Apple designer Bret Victor has done with the Javascript language, trying to make it a more visual endeavor. And Larson compares it to the new Wolfram programming language designed by respected scientist and technologist Stephen Wolfram. Andrew Stone, an independent Apple developer living in New Mexico, calls it "absolutely amazing."

This is the wild card. Depending how well it works, Pollak says, the playground could push mainstream programming in a new direction, providing a more intuitive means of building code. "This isn't necessarily an Apple invention, but it is something that has been bubbling up over the last few years," he explains.

But if you put the playground to one side, Swift isn't really that different from many other languages. It's mainly a way of bringing Apple's platform up to par with languages like Ruby and Python–all without sacrificing the speed of Objective-C.

The Apple Way

For this reason, Pollak believes that, rather than building Swift, Apple should have embraced a language that already provides many of the same benefits, merging the Apple world world with modern tools that are already used by an enormous community of developers. "They'd be much better off moving Ruby to their platform—or Python," he says.

But Apple is pretty much behaving as you'd expect. It appears that Swift is an extension of Objective-C–a language the company has spent years refining–and it builds atop the LLVM language compiler, an extremely powerful tool that Apple has used to shape its C-based tools in more recent times. The company is working with what is already in place.

But beyond that, as Andrew Stone points out, Apple is a company that very much believes in building things on its own–in what it sees as the right way–rather than waiting for the outside world to create some kind of standard platform that everyone can use. The company once embraced the popular Java programming language–now use to build apps on Android devices–but it ditched Java long ago in favor of its home grown platform. Swift is the most natural expression of the Apple Way–and that's another reason developers went nuts at the keynote.

The question is how many people will not only cheer for it, but actually use the thing. Apple has tried to make this as easy as possible. The company's new developer tools let you use Swift alongside Objective-C. But a keynote presentation does not a language make, and coders like Larson and Stone aren't yet sure if it's the way to go. "I've been using Objective-C for 25 years, so we'll see," Stone says. "I love

playing with new things. But is it right for me?"

Additional reporting by Christina Bonnington

[1]*Correction 13:57EST 06/03/13: An earlier version of this story said that Objective-C was developed at Next Computer. It was developed elsewhere and then adopted by Next.*