A new era for native mobile development?

By Matthew Longden

13 May 2017

Over the last few years, new languages have been introduced for both iOS and Android that could see the start of a new era in native mobile development - one where teams can collaborate more and may even be able to work on both systems.

Until very recently, developing native mobile applications meant having an iOS team who coded in Objective C and an Android team who developed in Java. Stark differences between the two ecosystems and the languages themselves meant there were few developers who were experts on both platforms. As a result, development for each system often occurred in relative isolation.

Swift

Apple introduced Swift, initially as a proprietary language and a predecessor to Objective C, for developing iOS and OS X apps in 2014. Despite the fact that it was a radical improvement to the rather old Objective C, uptake for Swift was initially slow. Lack of backwards compatibility between each release did not help developers migrate their projects with confidence.

With the release of Swift 2, and taking Swift open source, the floodgates for adoption have been opened – in January 2016, Swift overtook Objective C in the TIOBE index. Unfortunately, the release of Swift 3 did not include a stable Application Binary Interface (ABI) and so we will probably need to wait for Swift 4 before developers consider this a safe language to invest in.

Kotlin

Kotlin has been developed by the JetBrains team who produce Android Studio - the official IDE for Android development. It has been around since 2011 but the first stable release was made in February 2016. It is a JVM language, which means it compiles down to Java byte code.

It represents a step forwards from Java but has been designed to be fully interoperable with Java, even to the point where a lot of the syntax should be easily recognisable to Java developers.

While Kotlin has not been announced as Google's chosen replacement for Java on the Android platform just yet, the support for the language via Android Studio has helped drive adoption.

Both Swift and Kotlin are seen as modern programming languages. When compared to their respective predecessors, they bring a host of features and structures that the old languages do not support.

At a high level, it can be argued that both of the new languages are:

- Easier to read
- Easier to maintain
- Less verbose
- Have improved safety - thanks to strict strongly typed systems, which include nullability
- Have a more modern syntax

Both languages have powerful features like extensions and immutability and they support functional as well as procedural programming paradigms, allowing developers to use new patterns in their designs.

Similarities

An interesting aspect of these languages is their similarity. In fact, the syntax is so similar, it can be tricky to distinguish them. This provides opportunities for development teams. The ability to more easily share designs between the platforms have mixed peer reviews where Android and iOS devs review each other's code and potentially share common libraries would change the landscape for mobile developers.

While I don't see the system specialists disappearing anytime soon, there is the very real prospect of support teams working on both platforms and even building development teams with central mixed discipline developers supported by systems experts. At the very least, the syntactical similarity means that cross training from one platform to another should be a simpler job in the future – anyone who has tried to persuade a Java developer to take up Objective C will appreciate this!

Balance to tip more in favour of native development

These developments may also have a bearing on the 'native vs hybrid' conversation that occurs periodically in nearly every mobile team. With little overhead of switching between systems, shared code and easier support, I'd expect the balance to tip even more in favour of native development.

Systems like Xamerin, React Native and Cordova may in time feel the pinch from these new languages.

A further advantage of these new languages is the way they are being adopted as server side languages. IBM is using Swift for developers to write server side applications and Spring has announced that Spring Framework 5.0 will have dedicated Kotlin support. In one of our existing teams, where the mobile team is also responsible for writing service layer code, this possible alignment of languages across the systems is an exciting prospect.

While it’s unlikely that we’ll reach the point of just using one of these languages to code for both mobile platforms anytime soon, their similarity, and the fact that their adoption is moving outside of just the mobile arena, will impact our mobile teams.

More robust products built from closer developer collaboration, consolidated and streamlined support teams, and the ability to keep developers engaged by allowing them to more easily work as full stack developers, are some of the ways future mobile teams will be different from those of today.

ABOUT THE AUTHOR

Matthew Longden is the delivery manager at Entelect.