1. What is the piece of code, you've written in Objective-C or Swift, of which you are most proud? Please post a link (or attach the code).

Air France Music, an iOS app developed by me several years ago (2012) which required the use of augmented reality, displaying markers on the view based on the user's location. I'm most proud of this app (and code in it's entirety) because it was such a challenge, especially being one of my first apps. While the app was extremely challenging to develop (even with the use of a base AR library), a lot of the code had to be improved to work on devices at that time, being iPhone 3G and iPhone 4. Because of the limited processing power on these devices compared to modern iPhones, a lot of effort went into memory management and performance enhancements. While there were many late nights developing the app, the overwhelming sense of achievement at the end made everything worthwhile. The app went on to win several awards during 2012 and 2013.

The code can be found on my Github page (private repo, but will grant access upon request).

2. What is your favourite language feature of Obj-C or Swift?

Obj-C: *Perhaps Categories within Obj-C has been a big plus point for me, easily allowing me to extend different items within the apps, such as UIColor and UIFont, and just reference these items by calling [UIColor lightRedColor] for example, hiding the hex codes in the Category class.*

Additionally, not so much a language feature, but the naming convention within Obj-C has always been a plus point. Better to have extremely long method or variable names that gives the developer clarity, rather than other programming languages that seem to advocate shorter names that create ambiguity.

Swift: *Generally, I like Swift because it's a more modern language compared to Obj-C. It's a lot more flexible in terms of declaring variables with the optionals feature. While I still spend some time programming in Obj-C and will continue to do so, moving towards Swift has been a good thing, and perhaps a long time coming for Apple.*

Switch statements with String values also a big big positive.

3. What frustrates you the most about Obj-C or Swift?

Obj-C: *The language is dated and it feels dated. Declaring variables, objects just seems long-winded compared to more modern programming languages. Aside from this, it's actually a very robust language, and for good reason considering it's age.*

Swift: *While the language is now almost three years old, it still doesn't seem ready. The number of version updates that are coming out each year, with code breaking changes does become quite frustrating, especially when Migration Assistant within Xcode only works some of the time. With the release of Swift 3, and now 3.1, it seems that the language is moving towards completion, so a lot more developers are taking the language seriously compared to how things were in the language's infancy.*

4. What was the toughest bug you encountered while developing for the iOS/macOS platform,
and how did you track it down?

Again going back to the Air France Music project, there was an issue with memory performance on older devices (iPhone 3G) and improving this performance was of the utmost importance. In order to do this, the app had to be profiled using the Xcode Instruments to find out what was causing the

most time delay. While it wasn't a bug, I found that the issue was caused by allocating a new NSDateFormatter into memory whenever a method was being called. By declaring this in the class, and simply referencing the sole object in the method when required, the performance improved dramatically.

A lot of other bugs I've experienced have actually occurred within xib or Storyboards, which can be extremely tough to fix considering it involves either digging through the XML source code, or spending time checking all of the configurations on each UI object within the xib/Storyboard.

5. Describe one of your favourite features of Xcode?

Instruments. Having a series of tools which allows the developer to profile the app and test for memory leaks etc makes a world of difference. During my years of iOS app development, I've been extremely grateful for these tools, especially in the earlier years without the existence of ARC, ensuring that the memory management was rock-solid.