202: Swift (Conclusion)

All right, that concludes this talk on Swift generics, closures, and enums. I hope it gave you some insight in how to use these new features from Swift.

They let you do a lot of cool stuff that is much harder to pull off in Objective-C.

What Did You Learn

Closures. You've seen that, apart from a few minor syntax differences, closures are really no different from functions and methods in Swift. They have the same type signature and you invoke them the same way that you call a function.

In fact, you can use closures instead of functions and methods, and you can use functions and methods instead of closures. They are completely interchangeable. Just think of a closure as a function without a name.

Closures and functions are so-called *first-class citizens* in Swift. They are really an integral part of the language, not something that is bolted on afterwards, like blocks are in Objective-C.

The one thing that does set closures apart from functions and methods is that they capture their environment. Whenever you use a variable from an outer scope inside the closure, it will be captured. This is especially important if the closure is inside a class, in which case it will capture `self` if it uses an instance variable or method.

You've also seen how you can use a capture list to change how a variable is captured, by making it `weak` or `unowned` instead of `strong`. This can be important to prevent ownership cycles and memory leaks. Unfortunately, you still have to worry about those with Swift.

Generics. You also learned about generics. You've seen how you can make container types generic – like the `Grid` we used in Tic-Tac-Toe. For this game we made it a grid of `Player` objects, but you can take the same source code and use it in a chess game and give the grid `ChessPiece` objects instead.

Because `Grid` is generic, it doesn't really care what sort of objects it contains. You can pass in any type that you want. But once you've given it a type, that `Grid` instance can only contain objects of that particular type. That's what makes it type-safe.

You’ve also seen how you can make functions and methods generic, so you can use the same function with different data types.

When applied well, generics can really save a lot of duplicate code. They also allow you to write algorithms that are independent of the actual data you’re processing.

Enums. And finally, enums, in particular enums with associated values. When I first came across this concept it didn’t make any sense to me, but now I think it’s one of the best features of Swift.

These associated values allow you to give enums one or more pieces of data at runtime. In the playground we used this to associate a username with the `Player`. And if you got around to it in the challenge, you also associated a `UIImage` with the `Player`.

Enums with associated values are really cool – after all, it is what makes optionals possible.

You’ve also seen how to use `switch` to read these enum values, and how to do basic pattern matching. Switch and enums really go hand-in-hand.

Where To Go From Here?

As you’re no doubt aware, we have a ton of free tutorials on our website, and we’re updating our old tutorials to Swift. Not many of those tutorials use generics yet, but I’m sure that new tutorials will start to make use of these cool Swift features.

If you like games, you may be interested in my tutorial on how to make a game like Candy Crush. This shows how to use generics, closures, and enums, so you may want to check that out.

Of course, we also have a book, *Swift by Tutorials*. It covers all of this stuff in depth, with real-world examples. It’s definitely worth a read.

Apple also has a book. This is a must-read. If you’ve read it and it didn’t make much sense before, then I suggest you go read it again after this conference and – hopefully! – things should be a lot clearer the second time around.

If you want to talk to more about this stuff, you can also find me on Twitter and in the Ray Wenderlich forums. And of course I’ll be available here at the conference, so please feel free to come up to me any time.

By the way, earlier I mentioned that closures are great for functional programming. If you’re interested in that, check out the talk at 3 PM today in this room, because that goes into depth on how to use functional programming principles with Swift.

Thank you! And enjoy your lunch!

[4:30]