Structs, Classes, Enums, and Protocols

Jon Flanders
MOBILE ARCHITECT
@jonflanders www.jonflanders.com

Swift / Objective-C Interop Basics

To bring Objective-C code into Swift – create and configure a “bridging” header. #import Objective-C header files.

To bring Swift code into Objective-C – add #import for “<ProjectName>-Swift.h”

@objc can add to:

class

class methods

protocols

Objective-C Attribute in Swift

Structs vs Classes

Structs

Memory allocated on stack (no reference counting needed)

Can have initializers, properties, methods, and can implement protocols

Passed by value (efficient copy on write semantics) – each actor gets its own copy

Can be extended

Immutable (methods marked with mutable excepted)

No inheritance

Classes

Memory allocated on heap (must be reference-counted)

Can have initializers, properties, methods, and can implement protocols

Multiple references allow shared state

Can inherit functionality from a base class

Can be extended

Mutable

Can be type-checked at runtime

Can have a deinitializer to clean up resources

Apple’s advice:
Use structs whenever possible.

Reality: Structs only work "inside" of Swift code, can't pass to Objective-C code.

enums

Named constant values

Not limited to int values

First-class type

Computed properties

Methods

Can implement protocols

Only int-based enums can be Objective-C compatible!

Extensions

Like Obj-C categories

Can add computed props

New initializers

Define subscripts

Implement a protocol

A Swift extension can extend an Objective-C class, including Cocoa Touch classes!

Protocols

Similar to protocols in Objective-C

But much more powerful in Swift

All Swift types (classes, structs, and enums) can implement protocols!

Summary

Swift adds powerful features to our toolbox

- Protocols and Structs are awesome!
- Always remember interop