Swift <-> ObjC Interoperability

Nikita Lutsenko

@nlutsenko

Parse, Facebook, Banana!

Swift <-> ObjC Interoperability

Swift <-> ObjC

l14y

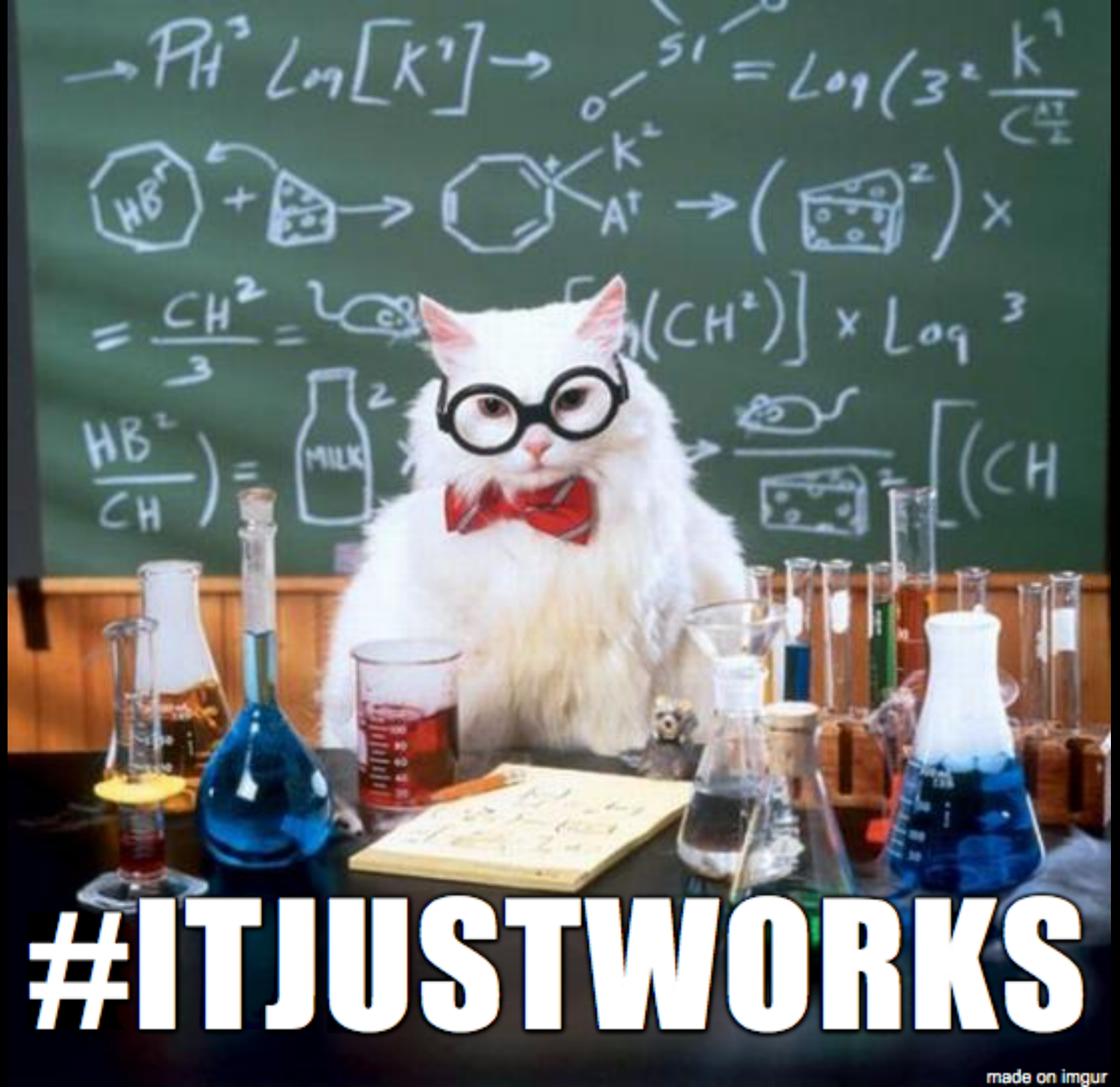
Agenda

→ ObjC -> Swift

→ Swift <- ObjC

→ Swift <-> ObjC

ObjC -> Swift


#ITJUSTWORKS
made on imgur

ObjC -> Swift

→ #ItJustWorks

→ 99% of API is `@available`

→ Swifty Names

→ Type Safety

→ Much better with Swift 3.0

NSInvocation and Friends

```
NSInvocation()
NSMethodSignature()

NSOperation(invocation: )
NSProxy.methodSignatureForSelector()

...

NS_SWIFT_UNAVAILABLE("NSInvocation and related APIs
                      not available")
```

ObjC -> Swift 2.*

→ Nullability Annotations

→ Error Handling

→ `NS_SWIFT_NAME`

→ `NS_SWIFT_UNAVAILABLE`

→ `NS_REFINED_FOR_SWIFT`

→ `NS_SWIFT_NOTHROW`

ObjC

```
- (BOOL)performRequest:(NSURLRequest *)request;
- (BOOL)performRequest:(NSURLRequest *)request error:(NSError **)error;
```

Swift 2.*

```
public func performRequest(request: NSURLRequest!) -> Bool
public func performRequest(request: NSURLRequest!, error: ()) throws
```

Nullability Annotations

```
- (BOOL)performRequest:(nullable NSURLRequest *)request;
- (BOOL)performRequest:(nullable NSURLRequest *)request error:(NSError **)error;
```

```
public func performRequest(request: NSURLRequest?) -> Bool
public func performRequest(request: NSURLRequest?, error: ()) throws
```

Nullability Annotations via Regions

```objc
NS_ASSUME_NONNULL_BEGIN

- (BOOL)performRequest:(NSURLRequest *)request;
- (BOOL)performRequest:(NSURLRequest *)request error:(NSError **)error;

NS_ASSUME_NONNULL_END
```

```swift
public func performRequest(request: NSURLRequest) -> Bool
public func performRequest(request: NSURLRequest, error: ()) throws
```

NS_SWIFT_NAME

```
- (BOOL)performRequest:(nullable NSURLRequest *)request
                    NS_SWIFT_NAME(perform(request:));

- (BOOL)performRequest:(nullable NSURLRequest *)request
                 error:(NSError **)error NS_SWIFT_NAME(perform(request:));
```

```
public func perform(request request: NSURLRequest?) -> Bool
public func perform(request request: NSURLRequest?) throws
```

NS_SWIFT_UNAVAILABLE

```
- (BOOL)performRequest:(nullable NSURLRequest *)request
                    NS_SWIFT_UNAVAILABLE("");

- (BOOL)performRequest:(nullable NSURLRequest *)request
                 error:(NSError **)error NS_SWIFT_NAME(perform(request:));
```

```
public func perform(request request: NSURLRequest?) throws
```

NS_SWIFT_UNAVAILABLE

```
- (BOOL)performRequest:(nullable NSURLRequest *)request;
- (BOOL)performRequest:(nullable NSURLRequest *)request
                 error:(NSError **)error;
```

```
public func performRequest(request: NSURLRequest) -> Bool
public func performRequest(request: NSURLRequest, error: ()) throws
```

NS_SWIFT_UNAVAILABLE

```
- (BOOL)performRequest:(nullable NSURLRequest *)request
                        NS_SWIFT_UNAVAILABLE("");
- (BOOL)performRequest:(nullable NSURLRequest *)request
                 error:(NSError **)error;
```

```
public func performRequest(request: NSURLRequest?) throws
```

Combine Everything

```
- (BOOL)performRequest:(nullable NSURLRequest *)request
                        NS_SWIFT_UNAVAILABLE("");

- (BOOL)performRequest:(nullable NSURLRequest *)request
                 error:(NSError **)error NS_SWIFT_NAME(perform(request:));
```

```
public func perform(request request: NSURLRequest?) throws
```

ObjC

```
- (void)getFirstName:(NSString **)firstName
            lastName:(NSString **)lastName;
```

Swift

```
func getFirstName(firstName: AutoreleasingUnsafeMutablePointer<NSString?>,
                  lastName: AutoreleasingUnsafeMutablePointer<NSString?>)
```

ObjC

```
- (void)getFirstName:(NSString **)firstName
            lastName:(NSString **)lastName
                        NS_SWIFT_NAME(get(firstName:lastName:));
```

Swift

```
func get(firstName firstName: AutoreleasingUnsafeMutablePointer<NSString?>,
         lastName: AutoreleasingUnsafeMutablePointer<NSString?>)
```

ObjC

```
- (void)getFirstName:(NSString **)firstName
            lastName:(NSString **)lastName NS_REFINED_FOR_SWIFT;
```

Swift

```
extension Person {
  var names: (firstName: String?, lastName: String?) {
      var firstName: NSString? = nil
      var lastName: NSString? = nil
      __getFirstName(&firstName, lastName: &lastName)
      return (firstName: firstName as String?, lastName: lastName as String?)
  }
}
```

ObjC -> Swift 3.0

→ "The Grand Rename"

→ Objective-C Lightweight Generics

→ `NS_NOESCAPE`/`CF_NOESCAPE`

→ `NS_EXTENSIBLE_STRING_ENUM`

ObjC Generics

```
@interface MyNetworkController<Request: NSURLRequest *> : NSObject

- (void)performRequest:(nullable Request)request;

@end
```

```
public class MyNetworkController : NSObject {
    public func performRequest(request: NSURLRequest?)
}
```

ObjC Generics

```objc
@interface MyNetworkController<Request: NSURLRequest *> : NSObject

- (void)performRequest:(nullable Request)request;

@end
```

```swift
public class MyNetworkController<Request : NSURLRequest> : NSObject {
    public func perform(_ request: Request?)
}
```

NS_NOESCAPE/CF_NOESCAPE

```
- (void)executeActions:(void (NS_NOESCAPE ^)(void))actions;
```

```
func executeActions(_ actions: @noescape () -> Void)
```

NS_NOESCAPE/CF_NOESCAPE

```
- (void)executeActions:(void (NS_NOESCAPE ^)(void))actions
                                NS_SWIFT_NAME(execute(actions:));
```

```
func execute(actions: (@noescape () -> Void))
```

NSEXTENSIBLESTRING_ENUM

ObjC

```
typedef NSString *NSRunLoopMode;
extern NSRunLoopMode const NSDefaultRunLoopMode;
```

Swift 2

```
public typealias NSRunLoopMode = NSString
public let NSDefaultRunLoopMode: String
```

NS_EXTENSIBLE_STRING_ENUM

ObjC

```
typedef NSString *NSRunLoopMode NS_EXTENSIBLE_STRING_ENUM;
extern NSRunLoopMode const NSDefaultRunLoopMode;
```

Swift 3

```
struct RunLoopMode : RawRepresentable {
  public static let defaultRunLoopMode: RunLoopMode
}
```

ObjC -> Swift

→ Use Nullability Annotations

→ Use Objective-C Generics

→ Rename (`NS_SWIFT_NAME`)

→ Refine (`NS_REFINED_FOR_SWIFT`)

→ Rinse (`NS_SWIFT_UNAVAILABLE`)

→ Repeat


I LOVE THIS JOB

Swift -> ObjC

Swift -> ObjC

→ No Tuples

→ No Generics

→ No Typealiases

→ No Swift Enums

→ No Swift Structs

→ No Global Variables

→ No Free-standing Functions

→ Only ObjC Runtime Compatible


NO FUN ALLOWED?
GOOD.
quickmeme.com

Swift -> ObjC

→ Subclasses of `NSObject`

→ Not `private` (`fileprivate`)

→ Not `@nonobjc`

→ `@objc` protocols

`@objc` & `@nonobjc`

→ Subclasses of `NSObject`

→ Explicit export to ObjC

→ Concrete extensions

→ `@objc` protocols

→ `Int` enums

`@objc` **protocols**

→ Special case of protocol

→ Weaker and not-Swifty

→ Supports `optional`

→ Extensions are inaccessible from ObjC

Value Types via `_ObjectiveCBridgeable`

→ Powers Foundation

→ Implementation Detail Protocol

→ Native ObjC Types <-> Native Swift Types

_ObjectiveCBridgeable

```
public protocol _ObjectiveCBridgeable {
  associatedtype _ObjectiveCType : AnyObject

  static func _isBridgedToObjectiveC() -> Bool

  func _bridgeToObjectiveC() -> _ObjectiveCType

  static func _forceBridgeFromObjectiveC(_ source: _ObjectiveCType, result: inout Self?)

  static func _conditionallyBridgeFromObjectiveC(
    _ source: _ObjectiveCType,
    result: inout Self?
  ) -> Bool

  static func _unconditionallyBridgeFromObjectiveC(_ source: _ObjectiveCType?) -> Self
}
```

Swift <-> ObjC

Swift <-Pitfalls-> ObjC

Swift <-Pitfalls-> ObjC

→ Closure -> Block Performance

→ Type Compatibility

→ Objective-C Runtime Hackery

Type Compatibility

```
NSArray<NSString *> *names = @[ @"John", @"Jane" ];
MSArray<id <NSCopying>> *array = names;
```

Type Compatibility

```
let names: [String] = ["John", "Jane"]
let array: [CustomDebugStringConvertible] = names
```

Type Compatibility

```
let names: [String] = ["John", "Jane"]
let array: [CustomDebugStringConvertible] = names
```

```
fatal error: can't unsafeBitCast between types of
different sizes
```

Type Compatibility

```
let names: [String] = ["John", "Jane"]
let array: [CustomDebugStringConvertible] = names.map({
  $0 as CustomDebugStringConvertible
})
```

Objective-C Runtime in Swift

```swift
@objc class Animal: NSObject {
    let name = "Animal"
    override init() {
        super.init()
    }
}
```

```objc
@interface Plant: NSObject
@end

@implementation Plant

- (NSString *)name {
    return @"Plant";
}

@end
```

Objective-C Runtime in Swift

```
// ObjC
SEL selector = @selector(name);

Method originalMethod = class_getInstanceMethod([Animal class], selector);
Method swizzledMethod = class_getInstanceMethod([Plant class], selector);

method_exchangeImplementations(originalMethod, swizzledMethod);

// ObjC
NSLog(@"I am %@!", name);

// Swift
print("I am \(name)!")
```

Objective-C Runtime in Swift

```
// ObjC
SEL selector = @selector(name);

Method originalMethod = class_getInstanceMethod([Animal class], selector);
Method swizzledMethod = class_getInstanceMethod([Plant class], selector);

method_exchangeImplementations(originalMethod, swizzledMethod);

// ObjC
NSLog(@"I am %@!", name); // I am Plant!

// Swift
print("I am \(name)!") // I am Animal!
```


HELLO
YES, THIS IS DOG

Objective-C Runtime in Swift

```swift
@objc class Animal: NSObject {
    let name = "Animal"
    override init() {
        super.init()
    }
}
```

```objc
@interface Plant: NSObject
@end

@implementation Plant

- (NSString *)name {
    return @"Plant";
}

@end
```

Objective-C Runtime in Swift

```
@objc class Animal: NSObject {
    dynamic let name = "Animal"
    override init() {
        super.init()
    }
}
```

```
@interface Plant: NSObject
@end

@implementation Plant

- (NSString *)name {
    return @"Plant";
}

@end
```

Objective-C Runtime in Swift

```
// ObjC
SEL selector = @selector(name);

Method originalMethod = class_getInstanceMethod([Animal class], selector);
Method swizzledMethod = class_getInstanceMethod([Plant class], selector);

method_exchangeImplementations(originalMethod, swizzledMethod);

// ObjC
NSLog(@"I am %@!", name);

// Swift
print("I am \(name)!")
```

Objective-C Runtime in Swift

```
// ObjC
SEL selector = @selector(name);

Method originalMethod = class_getInstanceMethod([Animal class], selector);
Method swizzledMethod = class_getInstanceMethod([Plant class], selector);

method_exchangeImplementations(originalMethod, swizzledMethod);

// ObjC
NSLog(@"I am %@!", name); // I am Plant!

// Swift
print("I am \(name)!") // I am Plant!
```

Yes, this is dog!

Swift <-> ObjC

→ You don't need this!

→ Migrate to Swift!

→ OpenGL Graphics

→ Interacting with C++

→ System calls (server side ftw!)

→ Super-über-high-performance

Thank you!

Questions?

github.com/nlutsenko/swift-objc-interoperability

facebook.com/nsunimplemented

twitter.com/nlutsenko

github.com/nlutsenko