HOW DO I EVEN *Swift?*

WHO AM I?

JP SIMARD

@SIMJP

REALM.IO

Realm β

GITHUB.COM/REALM/REALM-COCOA

WHY SWIFT > OBJC?

- Optionals
- Type safety & inference
- Closures
- Tuples
- Super-Enums & their pattern-matching sidekick
- Functional programming

Q: WHAT DOES IT LOOK LIKE?

OPTIONALS

NIL CHECKS
NIL CHECKS EVERYWHERE
memegenerator.net

OPTIONALS

```swift
let possibleNumber = "123"
let convertedNumber = possibleNumber.toInt()
// convertedNumber is inferred to be of type "Int?", or "optional Int"

if convertedNumber != nil {
  println("convertedNumber: \(convertedNumber!)")
}

if actualNumber = convertedNumber { // optional binding
  println("actualNumber: \(actualNumber)") // => not an optional
}
```

TYPE SAFETY & INFERENCE

```
let anInt = 3
let aDouble = 0.1416
var pi = anInt + aDouble // Compile warning

pi = 3 + 0.1416
// Compiles: number literals are untyped
```

LIKE RUST & SCALA

CLOSURES

```
func backwards(s1: String, s2: String) -> Bool {
  return s1 > s2
}
sort(["b", "a"], backwards) // => ["a", "b"]
```

SWIFT CLOSURES 👉 OBJC BLOCKS

TUPLES

```
let http404Error = (404, "Not Found")
```

LIKE HASKELL & SCALA

SUPER-ENUMS* & THEIR PATTERN-MATCHING SIDEKICK

*OK, NOT EXACTLY THE CORRECT TECHNICAL TERM

SUPER-ENUMS*

```
enum Suit: String {
    case Spades = "Spades",
    Hearts = "Hearts",
    Diamonds = "Diamonds",
    Clubs = "Clubs"
}

let card: (Suit, UInt) = (.Spades, 1)
card.0.toRaw() // => "Spades"
```

PATTERN MATCHING

```
let card: (Suit, UInt) = (.Spades, 1)
switch card {
  case (let suit, 1):
    println("Ace of \(suit.toRaw())") // => Ace of Spades
  case (let suit, let number):
    println("\(number) of \(suit.toRaw()")
}
```

FUNCTIONAL PROGRAMMING

```
let numbers = [1, 2, 3, 4]
numbers.map {
    (number: Int) -> Int in
    return 3 * number
} // => [3, 6, 9, 12]
numbers.filter {$0 % 2 == 0} // => [2, 4]
```

LIKE HASKELL, SCALA & MANY OTHERS

GENERICS

LIKE... UH... EVERY MODERN LANGUAGE!

```swift
// Reimplement the Swift standard
// library's optional type
enum OptionalValue<T> {
   case None
   case Some(T)
}
var maybeInt: OptionalValue<Int> = .None
maybeInt = .Some(100)

// Specialized Array
var letters: [Array]
letters = ["a"]
```

Q: WHAT HAPPENED TO MY BELOVED

*

Q: WHAT HAPPENED TO MY BELOVED *?

- concepts are still there: reference types and value types
  - pointers still exist to interact with C APIs: `UnsafePointer<T>`, etc.

Q: WHAT HAPPENED TO MY BELOVED *?

C APIS ARE STILL USABLE

```
import Foundation
import Security

let secret = "Top Secret".dataUsingEncoding(NSUTF8StringEncoding)
let dict = [kSecClass as String: kSecClassGenericPassword,
    kSecAttrService as String: "MyService",
    kSecAttrAccount as String: "Some Account",
    kSecValueData as String: secret] as NSDictionary
let status = SecItemAdd(dict as CFDictionaryRef, nil)
```

Q: THAT'S COOL, BUT HOW DO I EVEN...

Q: THAT'S FINE, BUT

WHEN DO I USE IT?

WHEN TO USE SWIFT

- New apps
- Personal projects
- Scripts
- Bribe your boss to use it in production*

* I am not liable

Q: THAT'S FINE, BUT

HOW DO I INTERACT WITH C/ OBJC?

INTERACTING WITH C/OBJC

- `UnsafePointer<T>` is typed `COpaquePointer`
  - `UnsafeMutablePointer<T>`

```
var aString = "Barcelona"
withUnsafePointer(&aString) { (arg: UnsafePointer<String>) in
    println("Hello " + arg.memory) // => Hello Barcelona
}
```

Q: SURE, BUT
HOW DO I INTERACT WITH C++?

Q: SURE, BUT HOW DO I INTERACT WITH C++?

A: DON'T!

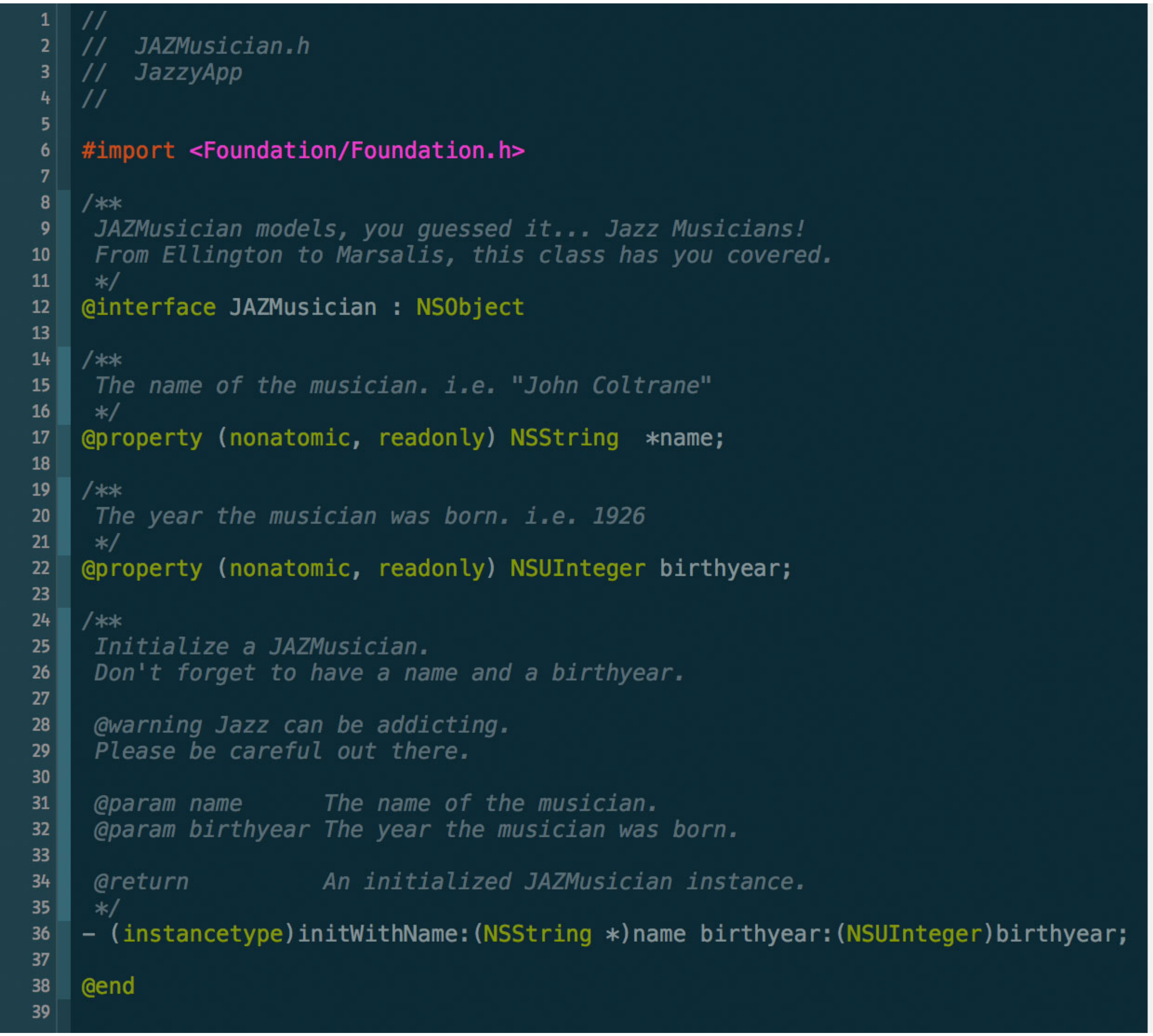
USE OBJECTIVE-C++ WRAPPERS

Q: HOW DO I EVEN GENERATE DOCS?

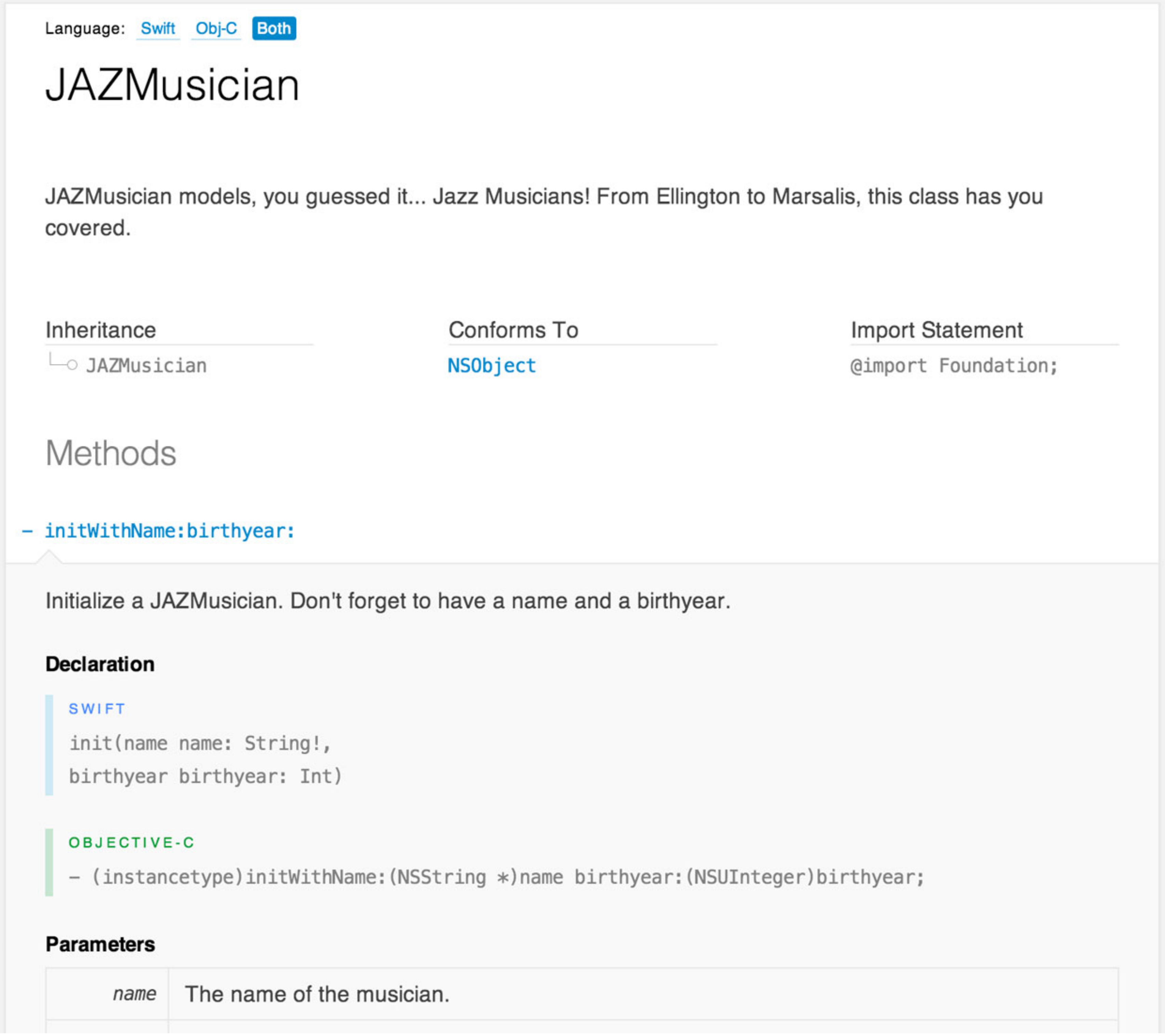
JAZZY♪♫

GITHUB.COM/REALM/JAZZY

A SOULFUL WAY TO GENERATE DOCS FOR SWIFT & OBJECTIVE-C



```objc
//
//  JAZMusician.h
//  JazzyApp
//

#import <Foundation/Foundation.h>

/**
 JAZMusician models, you guessed it... Jazz Musicians!
 From Ellington to Marsalis, this class has you covered.
 */
@interface JAZMusician : NSObject

/**
 The name of the musician. i.e. "John Coltrane"
 */
@property (nonatomic, readonly) NSString  *name;

/**
 The year the musician was born. i.e. 1926
 */
@property (nonatomic, readonly) NSUInteger birthyear;

/**
 Initialize a JAZMusician.
 Don't forget to have a name and a birthyear.

 @warning Jazz can be addicting.
 Please be careful out there.

 @param name      The name of the musician.
 @param birthyear The year the musician was born.

 @return          An initialized JAZMusician instance.
 */
- (instancetype)initWithName:(NSString *)name birthyear:(NSUInteger)birthyear;

@end

```



Language: Swift Obj-C Both

JAZMusician

JAZMusician models, you guessed it... Jazz Musicians! From Ellington to Marsalis, this class has you covered.

Inheritance	Conforms To	Import Statement
JAZMusician	NSObject	@import Foundation;

Methods

- initWithName:birthyear:

Initialize a JAZMusician. Don't forget to have a name and a birthyear.

Declaration

SWIFT

```swift
init(name name: String!,
birthyear birthyear: Int)
```

OBJECTIVE-C

```objc
- (instancetype)initWithName:(NSString *)name birthyear:(NSUInteger)birthyear;
```

Parameters

name	The name of the musician.

LINKS ()

- Official Swift blog
- The Swift Programming Language Book
- WWDC Videos
- WWDC Sample Code
- Xcode 6 (and other resources)

Free Apple Developer Account Required

LINKS (!)

- This talk: github.com/jpsim/talks
- Other Swift talks: realm.io/news
- Airspeed Velocity: airspeedvelocity.net
- ObjC/Swift doc generator: github.com/realm/jazzy
- Swift on StackOverflow

THANK YOU!

```
Meetup().questions?.askThem!!
Meetup().questions?.askThem!!
```

JP SIMARD, @SIMJP, REALM.IO