SWIFT

FOR RUBY DEVELOPERS

WHO AM I?

JP SIMARD

@SIMJP

REALM.IO

Realm β

GITHUB.COM/REALM/REALM-COCOA

& OBJECTIVE-C

RUBY & OBJECTIVE-C COEXIST

- RubyMotion
- CocoaPods
- liftoff
- jazzy
- xcpretty
- lots more

RUBY & OBJECTIVE-C ARE SIMILAR

- common ancestor: smalltalk
- dynamic dispatch
- dynamic typing
- `kind_of?` 👉 `isKindOfClass:`
- `respond_to?` 👉 `respondsToSelector:`

NIL CHECKS
NIL CHECKS EVERYWHERE
memegenerator.net

&

FEW SIMILARITIES BETWEEN RUBY & SWIFT

- REPL
- Good for scripting: `#!/usr/bin/xcrun swift`
- Functional concepts in the standard library
- String interpolation

DIFFERENCES

- Swift is still a **compiled language**
- **API's**, Libraries & Frameworks
- Type safety & generics
- Swift doesn't work outside

WHAT WOULD IT TAKE TO...

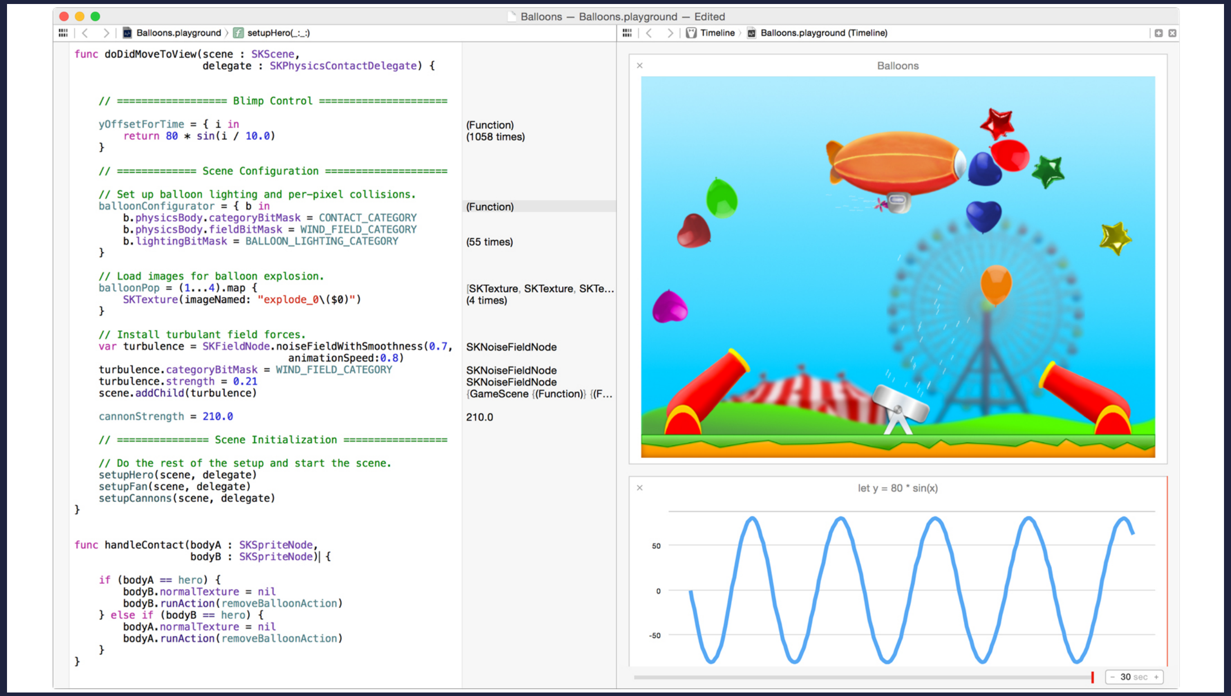
... RUN SWIFT OUTSIDE IOS/OSX?

1. Open source Swift compiler
2. Open source Swift runtime
3. Open source Swift standard library

Objective-C is 30 years old and they still haven't done #3.

RUBY REPL
==
SWIFT REPL + PLAYGROUNDS

xcrun swift



```swift
func doDidMoveToView(scene : SKScene,
                     delegate : SKPhysicsContactDelegate) {


    // ================== Blimp Control =====================

    yOffsetForTime = { i in
        return 80 * sin(i / 10.0)
    }

    // ============= Scene Configuration ====================

    // Set up balloon lighting and per-pixel collisions.
    balloonConfigurator = { b in
        b.physicsBody.categoryBitMask = CONTACT_CATEGORY
        b.physicsBody.fieldBitMask = WIND_FIELD_CATEGORY
        b.lightingBitMask = BALLOON_LIGHTING_CATEGORY
    }

    // Load images for balloon explosion.
    balloonPop = (1...4).map {
        SKTexture(imageNamed: "explode_0\($0)")
    }

    // Install turbulant field forces.
    var turbulence = SKFieldNode.noiseFieldWithSmoothness(0.7,
                                       animationSpeed:0.8)
    turbulence.categoryBitMask = WIND_FIELD_CATEGORY
    turbulence.strength = 0.21
    scene.addChild(turbulence)

    cannonStrength = 210.0

    // =============== Scene Initialization =================

    // Do the rest of the setup and start the scene.
    setupHero(scene, delegate)
    setupFan(scene, delegate)
    setupCannons(scene, delegate)
}


func handleContact(bodyA : SKSpriteNode,
                   bodyB : SKSpriteNode) {

    if (bodyA == hero) {
        bodyB.normalTexture = nil
        bodyB.runAction(removeBalloonAction)
    } else if (bodyB == hero) {
        bodyA.normalTexture = nil
        bodyA.runAction(removeBalloonAction)
    }
}
```

```
(Function)
(1058 times)

(Function)

(55 times)

[SKTexture, SKTexture, SKTe...
(4 times)

SKNoiseFieldNode

SKNoiseFieldNode
SKNoiseFieldNode
{GameScene {(Function)} {(F...

210.0
```

Demo

1. Classes
2. Closures
3. Type Safety & Inference
4. Mutability
5. Functional Programming
6. Optionals
7. Generics

1. CLASSES

RUBY CLASS

```ruby
class Vehicle
end

class Car < Vehicle
  def initialize(model)
    @model = model
  end

  def drive
    "driving my " + @model
  end
end

car = Car.new('Batmobile')
car.drive # => Driving my Batmobile
```

SWIFT CLASS

```swift
class Vehicle {}
class Car: Vehicle {
    var model = ""
    func drive() -> String {
        return "Driving my " + model
    }
}

let car = Car()
car.model = "Batmobile"
car.drive()
```

2. CLOSURES

RUBY CLOSURES

```ruby
def say_hello(&block)
  block.call
end

say_hello { puts "Hello there" } # => "Hello there"
```

SWIFT CLOSURES

```
func sayHello(block: () -> ()) {
  block()
}

sayHello { println("Hello there") } // => "Hello there"
```

3. TYPE SAFETY & INFERENCE

RUBY'S DYNAMIC TYPE

```
name = "John"
name = Time.now()
name = 123.45
```

SWIFT'S TYPE SAFETY & INFERENCE

```swift
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

4. MUTABILITY

MUTABILITY IN RUBY

```
str = "abc".freeze
# => "abc"
hash = { str => { str => "value" } }.freeze
# => {"abc"=>{"abc"=>"value"}}
hash[str] = "foo"
# => RuntimeError: can't modify frozen Hash
hash[str][str] = "bar"
# => "bar"
hash
# => {"abc"=>{"abc"=>"bar"}}
```

MUTABILITY IN RUBY

```
let str = "abc"
// => "abc"
let hash = [str: [str: "value"]]
// => ["abc": ["abc": "value"]]
hash[str] = [str: "foo"]
// => compile error
hash[str]![str] = "bar"
// => compile error
```

MUTABILITY IN SWIFT

- `var` is mutable
- `let` is immutable

```
var letter = "a"
letter = b // works

let a = "a"
a = "b" // compilation error
```

5. FUNCTIONAL PROGRAMMING

FUNCTIONAL PROGRAMMING IN RUBY

```
numbers = [1, 2, 3, 4]
numbers.map { |n|
  3 * n
} # => [3, 6, 9, 12]
numbers.select { |n| n % 2 == 0 } # => [2, 4]
```

FUNCTIONAL PROGRAMMING IN SWIFT

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

6. OPTIONALS

OPTIONALS

```
var string = ""
if string == nil {} // => compilation error: can never be nil

var optString: String?

if optString == nil {
  optString = "foobar"
}

if let forSureAString = optString {
  println("forSureAString: " + forSureAString)
}
```

7. GENERICS

```swift
// Re-implement 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: [String]
letters = ["a"]
```

LOTS MORE!

- Protocols
- Super-Enums ™
- Structs
- Pattern Matching
- Objective-C interoperability
- Runtime

SWIFT != RUBY

FUTURE

- Swift will displace Ruby for Mac-only scripting
- Tools like RubyMotion likely won't be too affected

LINKS ()

- Official Swift website (and 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
- From Ruby to Objective-C: speakerdeck.com/eddie
- Closures in Ruby
- Immutability in Ruby
- Why Rubyist Will Love Swift

THANK YOU!

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

JP SIMARD, @SIMJP, REALM.IO