SWIFT

FOR JAVASCRIPT DEVELOPERS

WHO AM I?

JP SIMARD

@SIMJP

REALM.IO

Realm β

GITHUB.COM/REALM/REALM-COCOA

SPONSORS

- PubNub
- DeNA
- FastForwardJS
- newcircle
- loop/recur
- wework
- SiliconValley CodeCamp

at WWDC...

JAVASCRIPT DEVS
I KNOW IOS
memegenerator.net

RUNS IN SWIFT & JS

```
var strings = ["a", "b"]
strings.reverse()
strings[0]
```

RUNS IN SWIFT & JS

```
var strings = ["a", "b"] // => [a, b]
strings.reverse() // => [b, a]
strings[0] // => Swift: a, JS: b
```

SIMILARITIES

- Syntax
- REPL
- Scripting feel

DIFFERENCES

- Swift is still a compiled language
- API's, Libraries & Frameworks
- Type safety & generics
- Functional concepts
- Swift will never work in-browser

WHAT WOULD IT TAKE TO...

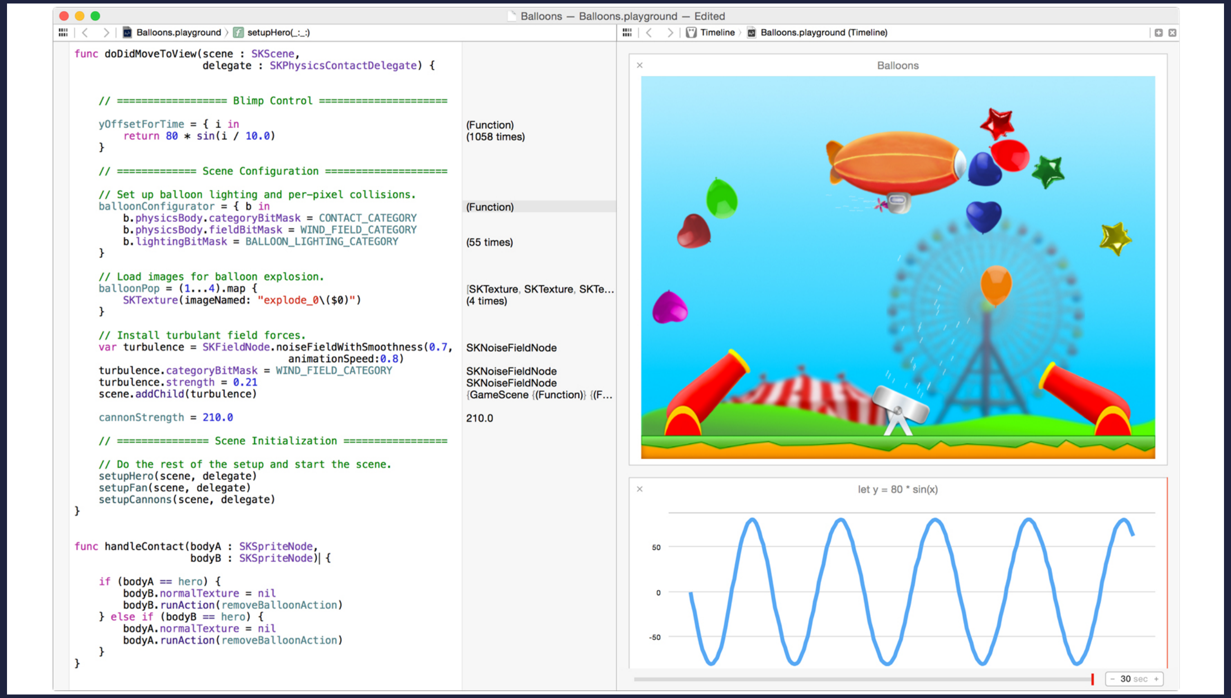
... RUN SWIFT OUTSIDE IOS/OSX

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.

NODE REPL
==
SWIFT REPL + PLAYGROUNDS

xcrun swift



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


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

    yOffsetForTime = { i in                                      (Function)
        return 80 * sin(i / 10.0)                                (1058 times)
    }

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

    // Set up balloon lighting and per-pixel collisions.
    balloonConfigurator = { b in                                 (Function)
        b.physicsBody.categoryBitMask = CONTACT_CATEGORY
        b.physicsBody.fieldBitMask = WIND_FIELD_CATEGORY
        b.lightingBitMask = BALLOON_LIGHTING_CATEGORY            (55 times)
    }

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

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

    cannonStrength = 210.0                                       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)
    }
}
```

DEMO

1. Classes
2. Callbacks
3. Promises
4. Type Safety & Inference
5. Tuples
6. Mutability
7. Functional Programming
8. Generics

1. CLASSES

JS "CLASS"

```
function Car(model){
  this.model = model;
}

Car.prototype.drive = function() {
  return 'Driving my ' + this.model;
}

var car = new car('Batmobile');
car.drive(); // => Driving my Batmobile
```

SWIFT CLASS

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

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

2. CALLBACKS

JS CALLBACKS

```
var log = function(txt, done) {
  setTimeout(function() {
    console.log('callbacks are ' + txt);
    done();
  }, 1000)
}

log('awesome', function() {
  console.log('and done');
});
```

SWIFT CALLBACKS

```
func log(txt: String, completion: () -> ()) {
    var delta = 1 * Int64(NSEC_PER_SEC)
    var time = dispatch_time(DISPATCH_TIME_NOW, delta)

    dispatch_after(time, dispatch_get_main_queue()) {
        println("closures are " + txt)
    }
}

log("not the same as JS closures") {
    println("and done")
}
```

3. PROMISES

JS PROMISES

```
var log = function(txt) {
  return new Promise((resolve) => {
    setTimeout(function() {
      console.log('promises are ' + txt);
      resolve();
    }, 1000)
  })
}

log('the future').then(() => {
  console.log('and done');
});
```

SWIFT PROMISES

```
func log(txt: String, #resolve: () -> (), #reject: () -> ()) {
    var delta = 1 * Int64(NSEC_PER_SEC)
    var time = dispatch_time(DISPATCH_TIME_NOW, delta)

    dispatch_after(time, dispatch_get_main_queue()) {
        println("closures are " + txt)
        resolve()
    }
}

log("not the same as JS closures",
    resolve: {
        println("and done")
    },
    reject: {
        // handle errors
})
```

4. TYPE SAFETY & INFERENCE

TYPE SAFETY & INFERENCE

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

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

LIKE RUST & SCALA

5. TUPLES

TUPLES

```
let http404Error = (404, "Not Found")
http404Error.0 // => 404
http404Error.1 // => Not Found
```

SWIFT ❤ JAVASCRIPT

6. MUTABILITY

RUNS IN SWIFT & JS

```
var strings = ["a", "b"] // => [a, b]
strings.reverse() // => [b, a]
strings[0] // => Swift: a, JS: b
```

MUTABILITY IN SWIFT

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

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

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

MUTABILITY IN JAVASCRIPT

- `var` is mutable
- `let` is mutable (only limits scope)
- `const` is immutable (only in FireFox & Chrome)

Object.freeze() IN JAVASCRIPT

```
var obj = {
  foo: "bar"
};

obj.foo = "baz"; // works
Object.freeze(obj); // freezes obj
obj.foo = "bar"; // silently does nothing
```

7. FUNCTIONAL PROGRAMMING

FUNCTIONAL PROGRAMMING

```
let numbers = [1, 5, 3, 12, 2]
numbers.map {
    (number: Int) -> Int in
    return 3 * number
} // => [3, 15, 9, 36, 6]
numbers.filter {$0 % 2 == 0} // => [12, 2]
```

LIKE UNDERSCORE.JS

8. GENERICS

```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"]
```

LOTS MORE!

- Optionals
- Super-Enums ™
- Structs
- Pattern Matching
- Runtime

SWIFT != JS

But!!!

SWIFT ❤ JS

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
- **MircoZeiss:** Swift for JavaScript Developers (👍🏼x💯)
- **ModusCreate:** JavaScript Take on Swift
- **DockYard:** Swift and JavaScript
- **Swift on** StackOverflow

THANK YOU!

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

`Meetup().questions?.askThem!`

JP SIMARD, @SIMJP, REALM.IO