Quirky Swift

… and other Swift stuff

Overview

- Getting To Know Swift
- Confusing Stuff
- Advanced Stuff

Getting To Know Swift

Seems Pretty stable, maybe wait 2 more weeks?


My body is ready.

Apple Swift Book

Literal	Type	Value
`__FILE__`	String	The name of the file in which it appears.
`__LINE__`	Int	The line number on which it appears.
`__COLUMN__`	Int	The column number in which it begins.
`__FUNCTION__`	String	The name of the declaration in which it appears.

println()

I should have written my own custom one a long time ago.

Swift Summit

Videos

λ

Functional Swift Conference

December 6th, Brooklyn

http://2014.funswiftconf.com

SHE-RA PRINCESS OF POWER

Magazine

Welcome to Xcode

Version 6.3.2 (6D2105)

Get started with a playground
Explore new ideas quickly and easily.

Create a new Xcode project
Start building a new iPhone, iPad or Mac application.

Check out an existing project
Start working on something from an SCM repository.

Show this window when Xcode launches

scratch.playground
~

Stocks
~/Desktop

Employees
~/Desktop

Groceries
~/Desktop

NameSearch
~/Desktop

ager
~/Desktop

readlineNSString
~/Desktop

HostName
~/Desktop

WorkWeek
~/dev

PBR
~/Desktop

Open another project…

```
import XCPlayground

func XCPCaptureValue<T>(identifier: String, value: T)

func XCPSetExecutionShouldContinueIndefinitely(continueIndefinitely: Bool = default)

func XCPExecutionShouldContinueIndefinitely() -> Bool

func XCPShowView(identifier: String, view: UIView)
```

Enums

Are awesome.

Have methods.

Conform to Protocols

Are not Just Ints.

```
enum CompassPoint {
    case North
    case South
    case East
    case West
}
```

```
enum Barcode {
    case UPCA(Int, Int, Int, Int)
    case QRCode(String)
}
```

```
enum ASCIIControlCharacter: Character {
    case Tab = "\t"
    case LineFeed = "\n"
    case CarriageReturn = "\r"
}
```

```
enum ConstructionStatus {
  case OnTime
  case Delayed(Int)
}

var deathStarConstructionStatus = ConstructionStatus.Delayed(7)

switch deathStarConstructionStatus {
case .OnTime:
  println("The Death Star is complete. The Emperor is pleased.")
case .Delayed(0...3):
  println("Construction is slightly behind schedule.")
case .Delayed(4...6):
  println("Construction is behind schedule.")
case .Delayed(let months):
  println("Construction is \(months) months behind. " +
    "The Emperor is most displeased.")
}
```

Code Organization

```
class MyTableViewController: UITableViewController {

    var data = Array<String>()
    override func viewDidAppear(animated: Bool) {
        tableView.reloadData()
    }
}
```

```swift
extension MyTableViewController: UITableViewDelegate {

    override func tableView(tableView: UITableView, heightForRowAtIndexPath indexPath: NSIndexPath) -> CGFloat {
        return 80
    }
}
```

```swift
extension MyTableViewController : UITableViewDataSource {

    //NO Stored Properties here
    var itemsCount: Int {
        return data.count
    }

    override func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCellWithIdentifier("CELL", forIndexPath: indexPath) as! UITableViewCell
        return cell
    }

    override func numberOfSectionsInTableView(tableView: UITableView) -> Int {
        return 1
    }

    override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return itemsCount
    }

}
```

Default Parameters

```
public func addArrival(_ date: NSDate = NSDate()){

    let newArrival = Event(inOrOut: .Arrival, date: date)
    eventsForTheWeek.addObject(newArrival)
}

addArrival()
let someCrazyInterval = 24*60*60* //MORE NUMBERS
addArrival(NSDate(timeIntervalSince1970: someCrazyInterval))
```

```swift
func postNotification(center: NSNotificationCenter = NSNotificationCenter.defaultCenter()){
    let note = NSNotification(name: "WorkWeekUpdated", object: nil)
    println("Posting Notification: \(note)")
    center.postNotification(note)
}
```

Swift StdLib

SwiftDoc.org

Swift Protocols

Continuing on SwiftDoc.org

```swift
let stooges = ["Moe", "Larry", "Curly"]

var all = reduce(stooges, "", {
    $0 + " " + $1
})

all // " Moe Larry Curly"

extension String {
    mutating func dropFirstChar(){
        self.removeAtIndex(self.startIndex)
    }
}

all.dropFirstChar() // "Moe Larry Curly"
```

```
["Moe", "Larry", "Curly"]

" Moe Larry Curly"
(3 times)


" Moe Larry Curly"


" "


"Moe Larry Curly"
```

Stuff You Might Run Into

Casting

is as? as!

```
let item: MyClass = MyClass()
if item is MyClass {
    //do MyClass specific stuff
}
```

```swift
class B {
    func myFunc(){ println("B myFunc") }
}

class I: B{
    override func myFunc() { println("Inherited overridden myFunc") }
}
```

```swift
let b:B = I()
b.myFunc() //"Inherited overridden myFunc"

let i = b as? I
i?.myFunc() // "Inherited overridden myFunc"

let t = b as! I
t.myFunc() // "Inherited overridden myFunc"
```

- `indexPathsForSelectedRows`

Returns the index paths representing the selected rows.

Declaration

SWIFT

```swift
func indexPathsForSelectedRows() -> [AnyObject]?
```

OBJECTIVE-C

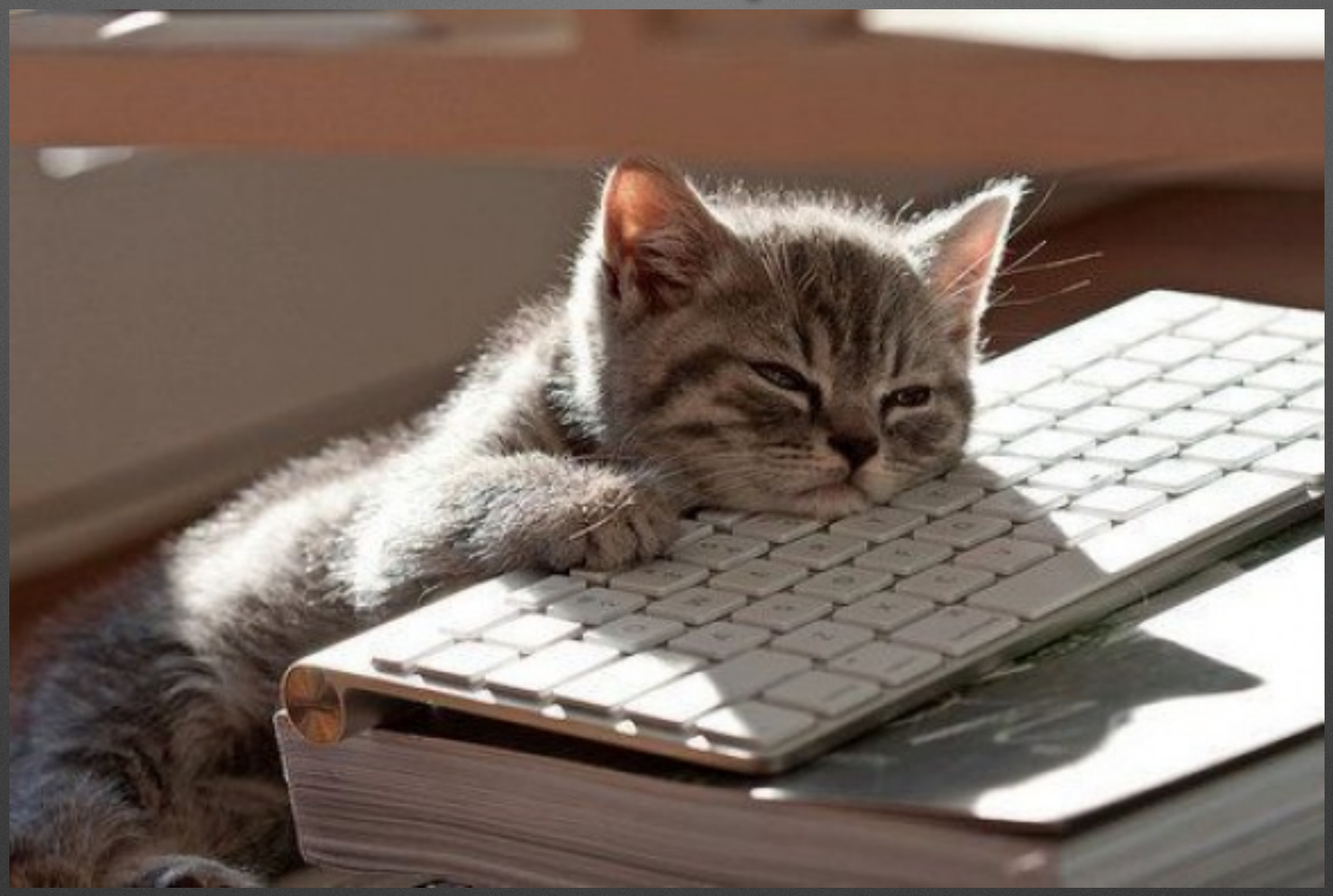
```objc
- (NSArray *)indexPathsForSelectedRows
```

Return Value

An array of index-path objects each identifying a row through its section and row index. Returns `nil` if there are no selected rows.

Properties and Methods

Stored Properties



```
struct Scene {
    var background: UIView
}
```

```
struct MyStruct {
    var notify: ()-> Void = {
        let note = NSNotification(name: "NOTE", object: nil)
        NSNotificationCenter.defaultCenter().postNotification(note)
        println("Original Notification")
        }
}

var c = MyStruct()
c.notify()
c.notify = { println("Notify Called") }
c.notify()
```

Computed Properties

```
extension NSDate {
    var isInThePast: Bool {
        return self.timeIntervalSinceNow < 0
    }
}

let d = NSDate(timeIntervalSince1970: 0)
d.isInThePast
```

```
true


"Dec 31, 1969, 6:00 PM"
true
```

WAT

```
4  import UIKit
3
2  struct Scene {
1      var color = UIColor.redColor()
!  5      var cg = color.CGColor   ! 'Scene.Type' does not have a member named 'color'
1  }
2
```

Lets get Lazy

```
public class DayTimePicker: NSObject {
    lazy var dateFormatter: NSDateFormatter = {
        return NSDateFormatter()
    }()
    private let calendar: NSCalendar
    public init(calendar: NSCalendar = NSCalendar.currentCalendar()){
        self.calendar = calendar
    }
}
```

Look at those awesome Parens

```
class LM {
    lazy var manager: CLLocationManager = {
        let m = CLLocationManager()
        m.desiredAccuracy = kCLLocationAccuracyBest
        m.distanceFilter = 200
        m.pausesLocationUpdatesAutomatically = true
        return m
    }()
}
```

Property Observers

```
class Person {
    var name = "Bill" {
        willSet { println("Setting name to \(newValue)")  }
        didSet  { println("Changed name from \(oldValue)")}
    }
}
```

```
class Person {
    let dateOfBirth: NSDate
    var age: Int {
        get {
            let years = 60 * 60 * 24 * 365.0
            return Int( -dateOfBirth.timeIntervalSinceNow / years)
        }
        set {
            //some math here, change dob to match age given
        }
        willSet {
            println("Changed to \(newValue)")
        }
    }
    init(dob: NSDate){
        dateOfBirth = dob
    }
}
```

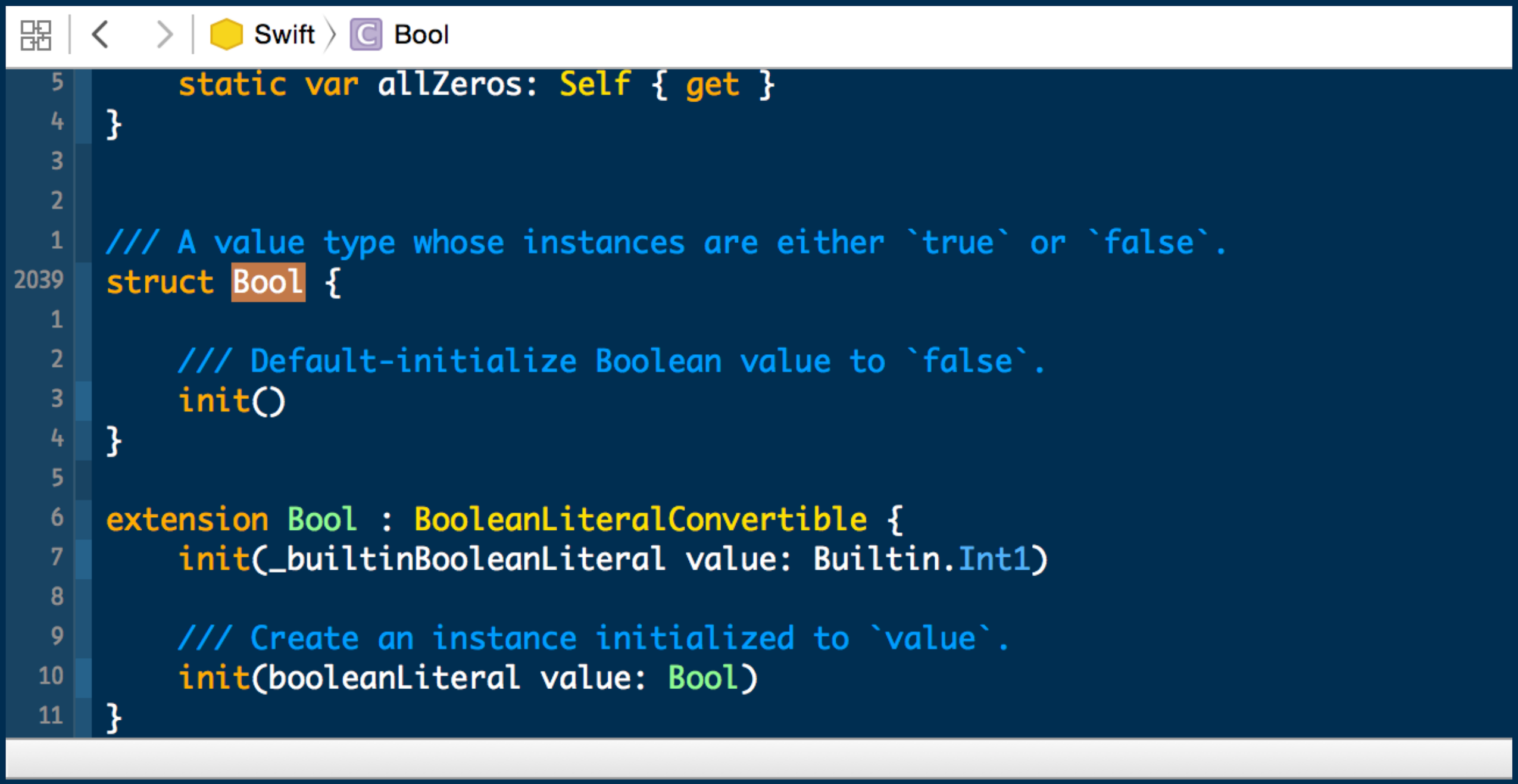
WillSet variable may not also have a set specifier

Switch on Bool

```swift
let imHere = true

switch imHere {
case true:
    println("Horray! Hope you are having fun")
case false:
    println("Bummer, Maybe It will be recorded")
//default:
//    println("HM.. Not sure")
}
```

Switch must be exhaustive, consider adding a default clause


Swift
Bool
5 static var allZeros: Self { get }
4 }
3
2
1 /// A value type whose instances are either `true` or `false`.
2039 struct Bool {
1
2 /// Default-initialize Boolean value to `false`.
3 init()
4 }
5
6 extension Bool : BooleanLiteralConvertible {
7 init(_builtinBooleanLiteral value: Builtin.Int1)
8
9 /// Create an instance initialized to `value`.
10 init(booleanLiteral value: Bool)
11 }

Generators

Map

```
4
3  let stooges = ["Moe", "Larry", "Curly"]
2
1  let tooges = stooges.map{                 Cannot invoke 'map' with an argument list of type '((_) -> _)'
7      var name = $0
1      name.removeAtIndex(name.startIndex)
2      println(name)
3      return name
4  }
5  tooges
6
7  // Want ["oe", "arry", "urly"]
8
```

```
let stooges = ["Moe", "Larry", "Curly"]

let tooges = stooges.map{ (stooge: String) -> String in
    var name = stooge
    name.removeAtIndex(name.startIndex)
    println(name)
    return name
}
tooges
```

```
["Moe", "Larry", "Curly"]

["oe", "arry", "urly"]
(3 times)
(3 times)
(3 times)

["oe", "arry", "urly"]
```

Pointers

```
CGColorCreate(CGColorSpaceCreateDeviceCMYK(),
      components: UnsafePointer<CGFloat>)
```

```
var darkGrey: [CGFloat] = [ 0.2, 0.2, 0.2, 1.0 ]
let color = CGColorCreate(CGColorSpaceCreateDeviceRGB(), &darkGrey)
UIColor(CGColor: color)
```

NSData

`- getBytes:length:`

Copies a number of bytes from the start of the receiver's data into a given buffer.

Declaration

SWIFT

```swift
func getBytes(_ buffer: UnsafeMutablePointer<Void>, length length: Int)
```

OBJECTIVE-C

```objc
- (void)getBytes:(void *)buffer length:(NSUInteger)length
```

Parameters

buffer	A buffer into which to copy data.
length	The number of bytes from the start of the receiver's data to copy to *buffer*.

Discussion

The number of bytes copied is the smaller of the *length* parameter and the `length` of the data encapsulated in the object.

```swift
/// Grabs the next two bytes from data. Interprets these as a UInt16 Sequence number
///
/// Note: Caller should call readAndIgnoreBytes(data, 2) after this call.
///
/// :param: data NSData to read the first 2 bytes from
/// :returns: A sequence number
func getSequenceNumber(data: NSData) -> UInt16 {
    var seq: UInt16 = 0
    data.getBytes(&seq, length: 2)
    return seq.byteSwapped
}
```

Scripting

```
#!/usr/bin/env swift

import Foundation
```

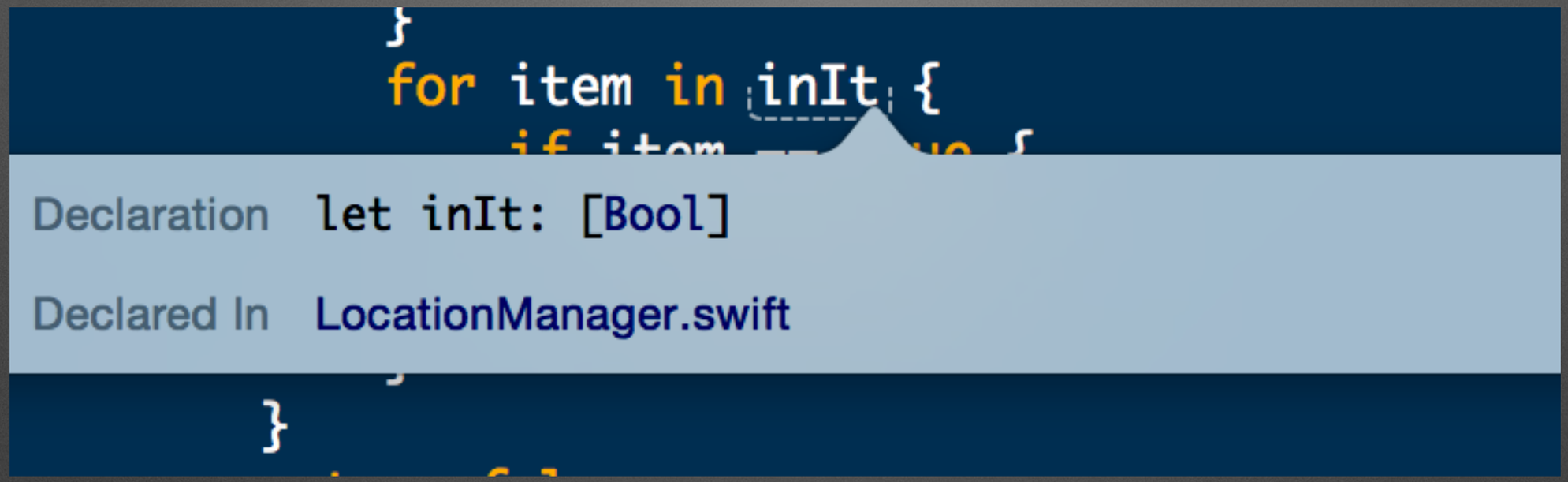
```
chmod +x script.swift
./script.swift
```

https://speakerdeck.com/ayanonagon/swift-scripting

Advanced Stuff

Type Inference

ABC: Always Be option-Clicking



Use the Option-click pop up

Since Swift is Strongly Typed you must know the types

Where Do T an U come from

```
public func restoreCollectionFromPath<T>(path: String) -> Array<T>?

public func restoreCollectionFromPath<T>(path: String) -> Set<T>?

public func restoreCollectionFromPath<T,U>(path: String) -> Dictionary<T,U>?
```

Maybe Here?

```
public func restoreFromArchiveArray<T>( array: NSMutableArray) -> Array<T>

public func restoreFromArchiveArray<T>(array: NSMutableArray) -> Set<T>

public func restoreFromArchiveArray<T,U>(array: NSMutableArray) -> Dictionary<T,U>
```

Curried Functions

```
func add(a: Int, b:Int) -> Int {
    return a + b
}
```

```swift
func add(a: Int, b:Int) -> Int {
    return a + b
}
```

```swift
func addTwo(a: Int) -> Int {
    return add(a, 2)
}
```

```
func add(a: Int) -> (Int -> Int) {
    return { b in a + b }
}
```

```
func addTwo(a: Int) -> Int {
    return add(a, 2)
}
```

```
func add(a: Int) -> (Int -> Int) {
    return { b in a + b }
}
```

```
let addTwo = add(2)
```

```swift
func add(a: Int)(b: Int) -> Int {
    return a + b
}
```

```swift
let addTwo = add(2)
```

“Instance Methods are really just curried class functions”

–Ole Begemann

```
class BankAccount {
    var balance: Double = 0.0

    func deposit(amount: Double) {
        balance += amount
    }
}
```

```
let account = BankAccount()
account.deposit(100) // balance is now 100
```

```swift
class BankAccount {
    var balance: Double = 0.0

    func deposit(amount: Double) {
        balance += amount
    }
}
```

```swift
let depositor = BankAccount.deposit
```

```swift
depositor(account)(100) // balance is now 200
```

```
var s = "Hello"
String.removeAtIndex(&s)(s.startIndex)
```

importing Swift

scratch.playground

scratch.playground 〉 No Selection

```
2  import UIKit
1
3  import Swift
1
```

– 30 sec +

-- INSERT --

Swift 〉 No Selection

```
infix operator & {
    associativity left
    precedence 150
}

infix operator >= {
    associativity none
    precedence 130
}

infix operator ~= {
    associativity none
    precedence 130
}

infix operator !== {
    associativity none
    precedence 130
}

infix operator ~> {
    associativity left
    precedence 255
}

infix operator | {
    associativity left
    precedence 140
}

infix operator >> {
    associativity none
```

That looks cool

Swift No Selection

Find !== 2 Done

```
274
1
2  /// Returns true if the arrays do not contain the same elements.
3  func !=<T : Equatable>(lhs: [T], rhs: [T]) -> Bool
4
5
6  /// Returns true if the arrays do not contain the same elements.
7  func !=<T : Equatable>(lhs: _UnitTestArray<T>, rhs: _UnitTestArray<T>) -> Bool
8
9  func !==(lhs: AnyObject?, rhs: AnyObject?) -> Bool
10
11 func %(lhs: UInt8, rhs: UInt8) -> UInt8
12
13 func %(lhs: Int8, rhs: Int8) -> Int8
14
```

Swift Header is not any help

130)

- < Less than
- <= Less than or equal
- > Greater than
- >= Greater than or equal
- == Equal
- != Not equal
- === Identical
- !== Not identical
- ~= Pattern match

Thanks

@john_regner