A Swift Introduction

Natasha Murashev
@NatashaTheRobot

Agenda

- Swift Basics
- Play Time
- Resources

Swift Basics

```objc
#import <Foundation/Foundation.h>

int main(int argc, const char * argv[])
  {
      @autoreleasepool {
          // insert code here...
          NSLog(@"Hello, World!");
      }
      return 0;
}
```

```
println("Hello, World!")
```

```
// explicit typing is optional
let minionJerry: String = "Jerry"
```

```
// the String type is inferred automatically
let minionBob = "Bob"
```

```
let name = "Dave"
var favorite = "Rockets and Missiles"
```

```
let name = "Dave"
var favorite = "Rockets and Missiles"

name = "Stuart"
```

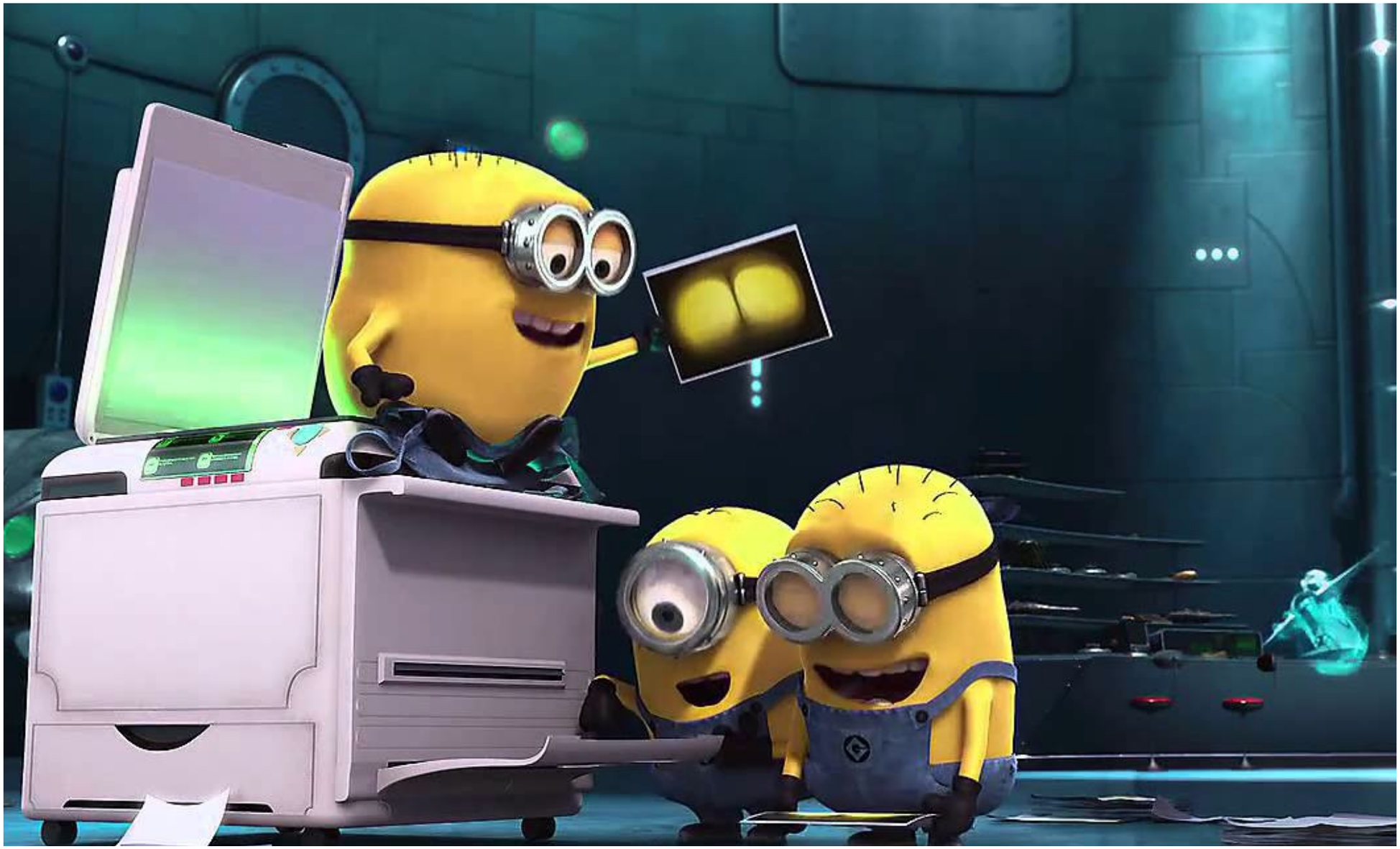
Cannot assign to 'let' value 'name'

```
let name = "Dave"
var favorite = "Rockets and Missiles"

favorite = "Singing"
```

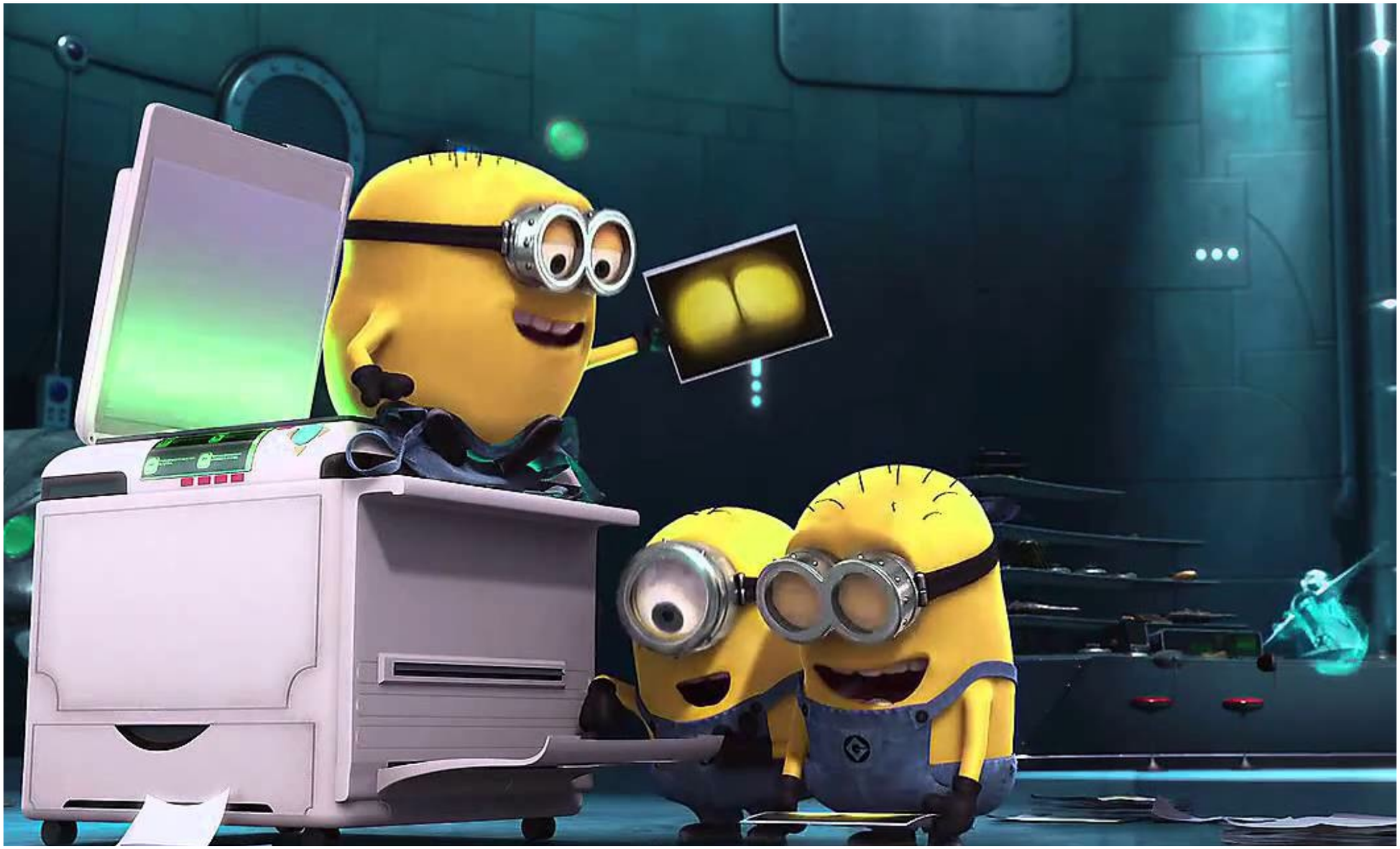
```swift
let name = "Dave"
var favorite = "Rockets and Missiles"

favorite = 54
```

Cannot convert the expression's type '()' to type 'String'

```swift
favorite = String(54)
```



```
let name = "Jorge"
let favoriteActivity = "making photocopies of his bottom"
let minionDescription = "Minion \(name) likes \(favoriteActivity)"
```



```
NSString *name = @"Jorge";
NSString *favoriteActivity = @"making photocopies of his bottom";
NSString *minionDescription = [NSString stringWithFormat:@"Minion %@ likes %@",
                                name, favoriteActivity];
```

```
var minionNames = [String]()
```

```
var minionNames: [String] = ["Bob", "Dave"]
```

```
var minionNames = [String]()

minionNames += "Bob"
minionNames += ["Dave", "Stuart", "Jerry", "Jorge",
  "Kevin", "Phil", "Tim", "Mark"]

// ["Bob", "Dave", "Stuart", "Jerry", "Jorge",
  "Kevin", "Phil", "Tim", "Mark"]
```

```
NSMutableArray *minionNames = [NSMutableArray array];
[minionNames addObject: @"Bob"];
[minionNames addObjectsFromArray:@[@"Dave", @"Stuart",
    @"Jerry", @"Jorge", @"Kevin", @"Phil", @"Tim",
@"Mark"]];
```

```
var minionNames = ["Dave", "Stuart",
    "Jerry", "Jorge", "Kevin", "Phil",
    "Tim", "Mark"]
```

```
minionNames += 5
```
'[String]' is not identical to 'UInt8'

```
NSMutableArray *minionNames = [NSMutableArray array];
[minionNames addObject: @"Bob"];
[minionNames addObjectsFromArray:@[@"Dave", @"Stuart",
    @"Jerry", @"Jorge", @"Kevin", @"Phil", @"Tim",
    @"Mark"]];

[minionNames addObject:@5];
```

```
var minionNames = [String]()

minionNames += ["Dave", "Stuart", "Jerry", "Jorge",
    "Kevin", "Phil", "Tim", "Mark"]

minionNames[5] // "Phil"

minionNames[0] = "Bob"

// ["Bob", "Stuart", "Jerry", "Jorge", "Kevin",
    "Phil", "Tim", "Mark"]
```

```swift
let minionNames = ["Dave", "Stuart", "Jerry",
    "Jorge", "Kevin", "Phil", "Tim", "Mark"]

for minionName in minionNames {
    println("Hang on \(minionName)!")
}
```

```
//Hang on Dave!
//Hang on Stuart!
//Hang on Jerry!
//Hang on Jorge!
//Hang on Kevin!
//Hang on Phil!
//Hang on Tim!
//Hang on Mark!
```

```
let minionNames = ["Dave", "Stuart", "Jerry", "Jorge",
    "Kevin", "Phil", "Tim", "Mark"]

for (index, minionName) in enumerate(minionNames) {
    println("Minion \(minionName) is number \(index + 1)
        on the wall")
}
```

```
//Minion Dave is number 1 on the wall
//Minion Stuart is number 2 on the wall
//Minion Jerry is number 3 on the wall
//Minion Jorge is number 4 on the wall
//Minion Kevin is number 5 on the wall
//Minion Phil is number 6 on the wall
//Minion Tim is number 7 on the wall
//Minion Mark is number 8 on the wall
```

```
var minionLunchOrder = [String : String]()
```

```
var minionLunchOrder: [String : String] = [:]
```

```
var minionLunchOrder = [String : String]()

minionLunchOrder["Jorge"] = "Banana"
minionLunchOrder["Mark"] = "Turkey Sandwich"

// ["Jorge": "Banana", "Mark": "Turkey Sandwich"]
```

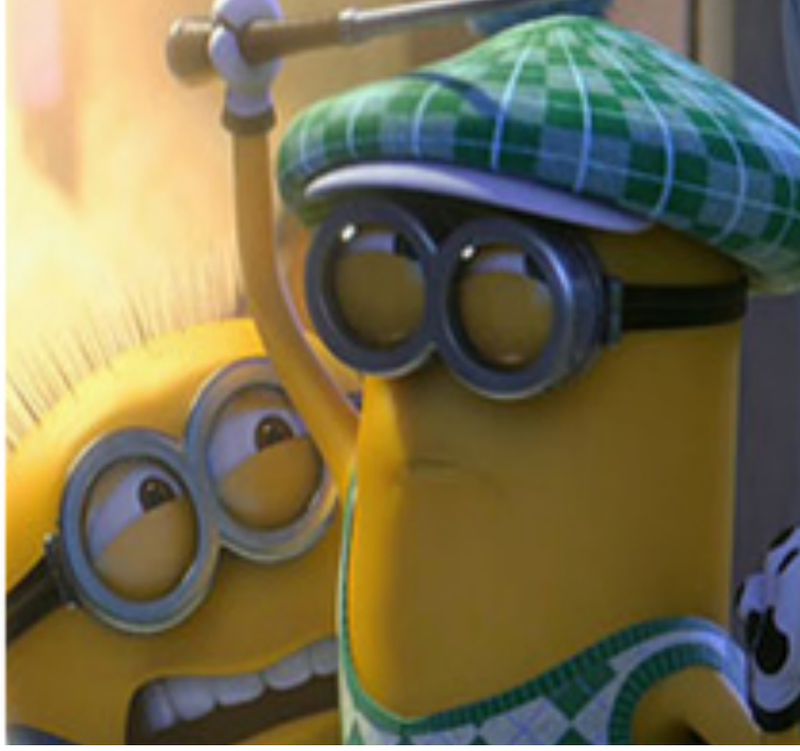
```
var minionLunchOrder = [String : String]()

minionLunchOrder["Jorge"] = "Banana"
minionLunchOrder["Mark"] = "Turkey Sandwich"

minionLunchOrder["Jorge"] // "Banana"
```

```
let minionLunchOrder = ["Jorge" : "Banana", "Mark" :
    "Turkey Sandwich", "Dave" : "Tuna Sandwich"]

for (minion, food) in minionLunchOrder {
    println("\(minion) ordered a \(food) for lunch")
}

// Jorge ordered a Banana for lunch
// Mark ordered a Turkey Sandwich for lunch
// Dave ordered a Tuna Sandwich for lunch
```

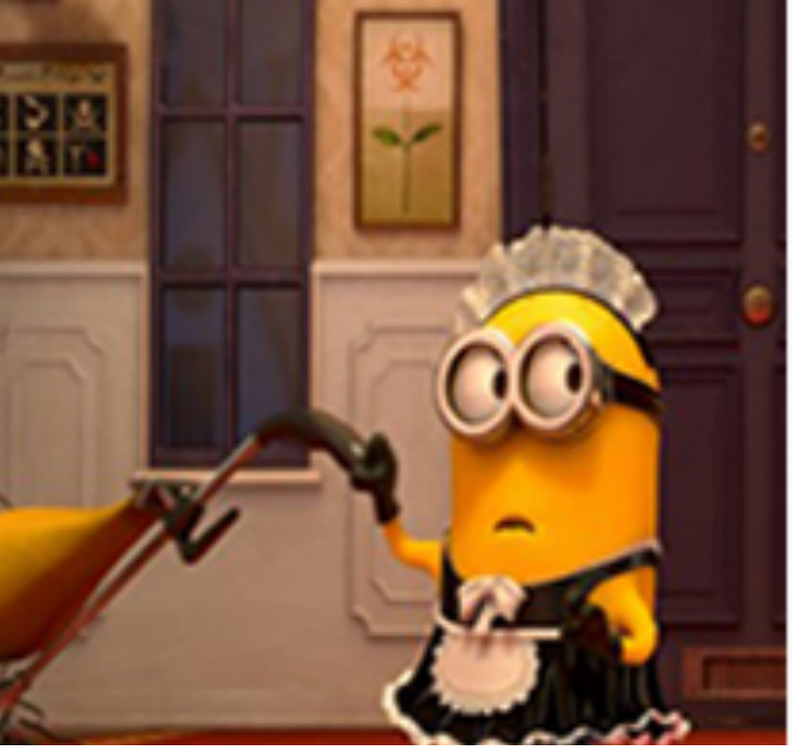
```swift
for i in 1..<5 {
    println("\(i) Chiquita Banana")
}

// 1 Chiquita Banana
// 2 Chiquita Banana
// 3 Chiquita Banana
// 4 Chiquita Banana
```

```swift
for i in 1...5 {
    println("\(i) Chiquita Banana")
}

// 1 Chiquita Banana
// 2 Chiquita Banana
// 3 Chiquita Banana
// 4 Chiquita Banana
// 5 Chiquita Banana
```

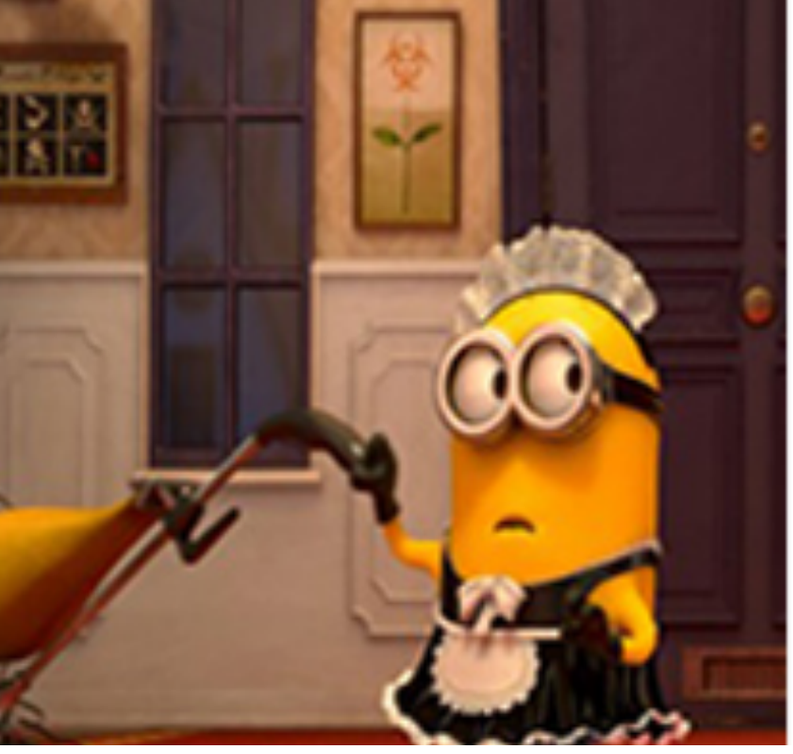




```
let minionHats = ["Phil" : "Bonnet",
                  "Carl" : "Fruit Hat",
                  "Kevin" : "Flat Cap"]

let stuartsHat = minionHats["Stuart"]
```

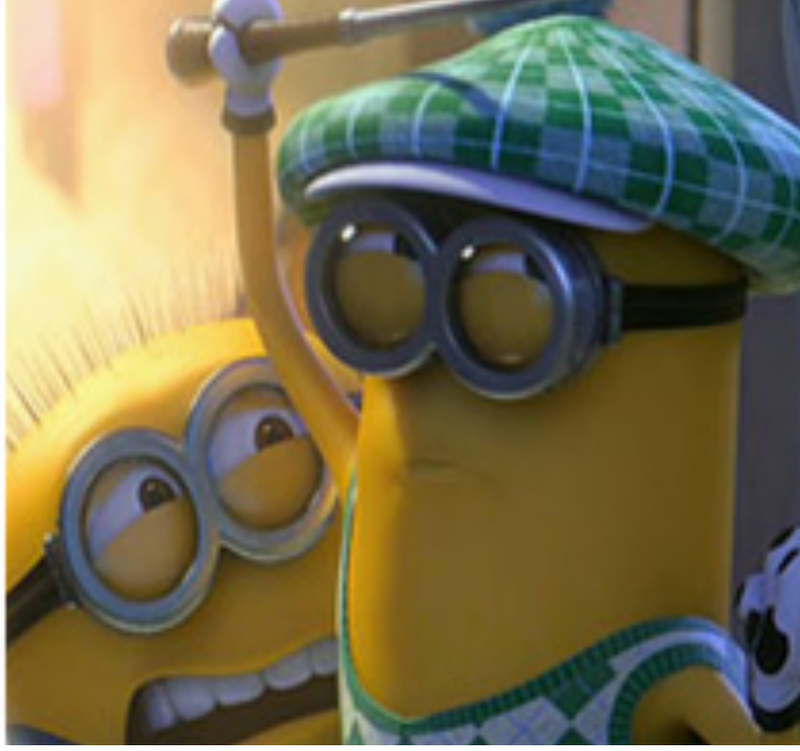


```
let minionHats = ["Phil" : "Bonnet",
                  "Carl" : "Fruit Hat",
                  "Kevin" : "Flat Cap"]

let stuartsPossibleHat : String? = minionHats["Stuart"]
```



```
let minionHats = ["Phil" : "Bonnet",
                  "Carl" : "Fruit Hat",
                  "Kevin" : "Flat Cap"]

let stuartsPossibleHat: String? = minionHats["Stuart"]

if stuartsPossibleHat == nil {
    println("Stuart is missing a hat!")
} else {
    let stuartsHat = stuartsPossibleHat!
    println("Stuart is wearing a \(stuartsHat.lowercaseString)")
}
```





```
let minionHats = ["Phil" : "Bonnet",
                  "Carl" : "Fruit Hat",
                  "Kevin" : "Flat Cap"]

let stuartsPossibleHat: String? = minionHats["Stuart"]

if let stuartsHat = stuartsPossibleHat {
    println("Stuart is wearing a \(stuartsHat.lowercaseString)")
} else {
    println("Stuart is missing a hat!")
}
```

```
func goodnight(minionName: String) -> String {
    return "Goodnight \(minionName)"
}

goodnight("Phil")

// "Goodnight Phil"
```

```
func goodnight(minionName: String) -> String {
    return "Goodnight \(minionName)"
}

goodnight(minionName: "Phil")
```

Extraneous argument label 'minionName:' in call

```
func goodnight(#minionName: String) -> String {
    return "Goodnight \(minionName)"
}

goodnight(minionName: "Phil")

// "Goodnight Phil"
```

```
func goodnight(minionName: String = "Minion") -> String {
    return "Goodnight \(minionName)"
}

goodnight(minionName: "Phil")
// "Goodnight Phil"


goodnight()
// "Goodnight Minion"
```

```
class AdorableCreature {
    let name: String
    let color: UIColor

    init (name: String, color: UIColor) {
        self.name = name
        self.color = color
    }
}
```

```
class Minion: AdorableCreature {
    let favoriteMischief: String

    init(name: String, favoriteMischief: String) {
        self.favoriteMischief = favoriteMischief
        super.init(name: name,
                   color: UIColor.yellowColor())
    }
}

class Monster: AdorableCreature {
    let scarinessLevel: String

    init(name: String, scarinessLevel: String) {
        self.scarinessLevel = scarinessLevel
        super.init(name: name,
                   color: UIColor.purpleColor())
    }
}
```

```swift
let creatures = [
    Minion(name: "Jorge", favoriteMischief: "making photocopies of his bottom"),
    Monster(name: "Mike", scarinessLevel: "meh scrary"),
    Minion(name: "Phil", favoriteMischief: "dressing up like a french maid"),
    Monster(name: "Sulley", scarinessLevel: "terrifying")
]
```

```swift
let creatures = [
    Minion(name: "Jorge", favoriteMischief: "making photocopies of his bottom"),
    Monster(name: "Mike", scarinessLevel: "meh scrary"),
    Minion(name: "Phil", favoriteMischief: "dressing up like a french maid"),
    Monster(name: "Sulley", scarinessLevel: "terrifying")
]

// the type of "creatures" is inferred as [AdorableCreature]
```

```
for adorableCreature in creatures {
    if let minion = adorableCreature as? Minion {
        println("Minion \(minion.name) loves \(minion.
     favoriteMischief)")
    } else if let monster = adorableCreature as? Monster {
        println("Monster \(monster.name) is \(monster.
     scarinessLevel)")
    }
}
```

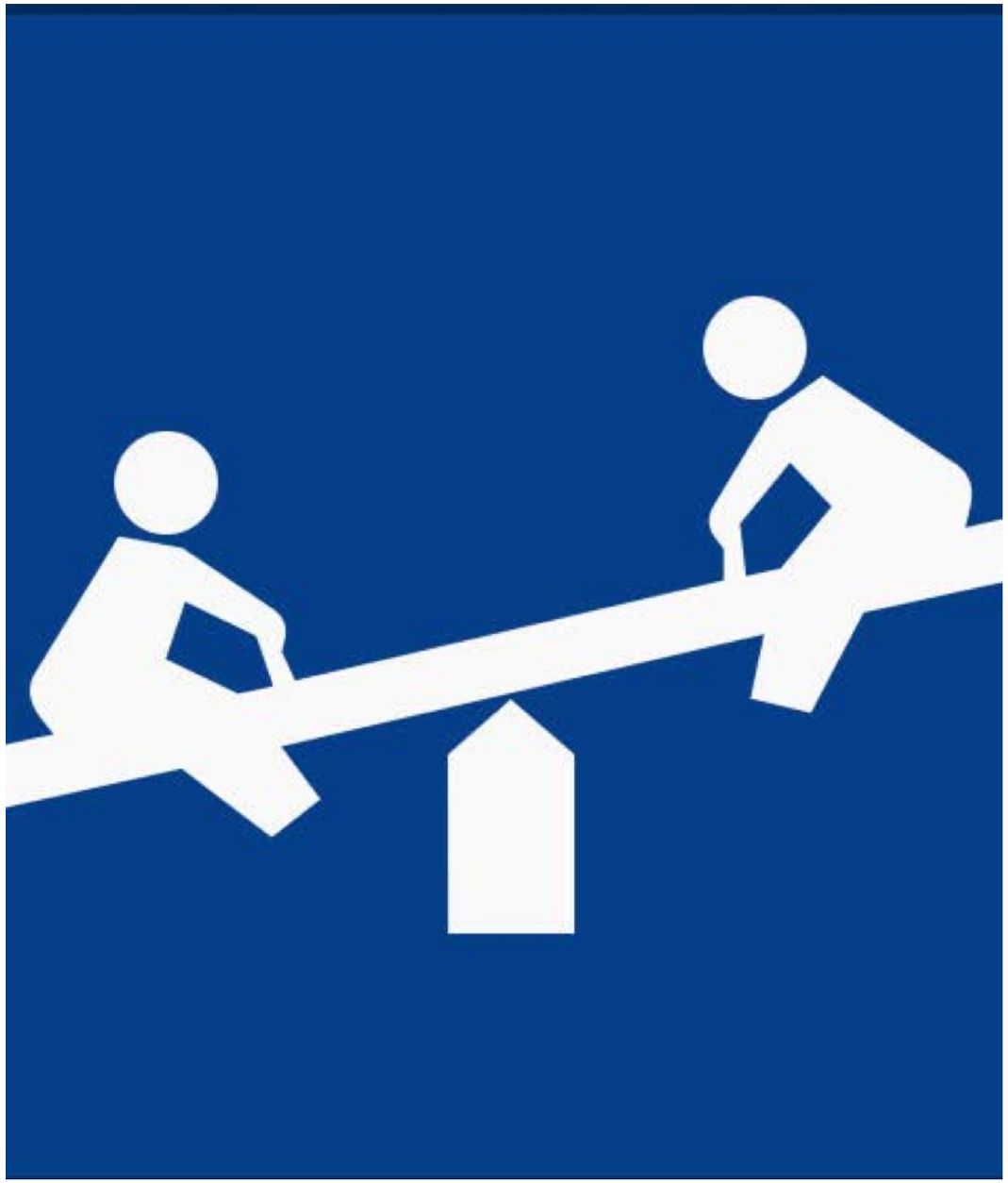
```
for adorableCreature in creatures {
    if let minion = adorableCreature as? Minion {
        println("Minion \(minion.name) loves \(minion.
            favoriteMischief)")
    } else if let monster = adorableCreature as? Monster
        {
        println("Monster \(monster.name) is \(monster.
            scarinessLevel)")
    }
}
```

```
// Minion Jorge loves making photocipies of his bottom
// Monster Mike is meh scrary
// Minion Phil loves dressing up like a french maid
// Monster Sulley is terrifying
```

Play Time!

Playgrounds

- Experiment
- Learn
- Algorithm Development
- Fun



Resources

Everyone is a beginner

Objective-C

Swift

- classes
- methods
- protocols
- extensions
- functions
- ARC
- closures
- Cocoa values & collections

+

Swift

- structs
- namespaces
- operator overloading
- ObjC interop
- Swift values & collections

+

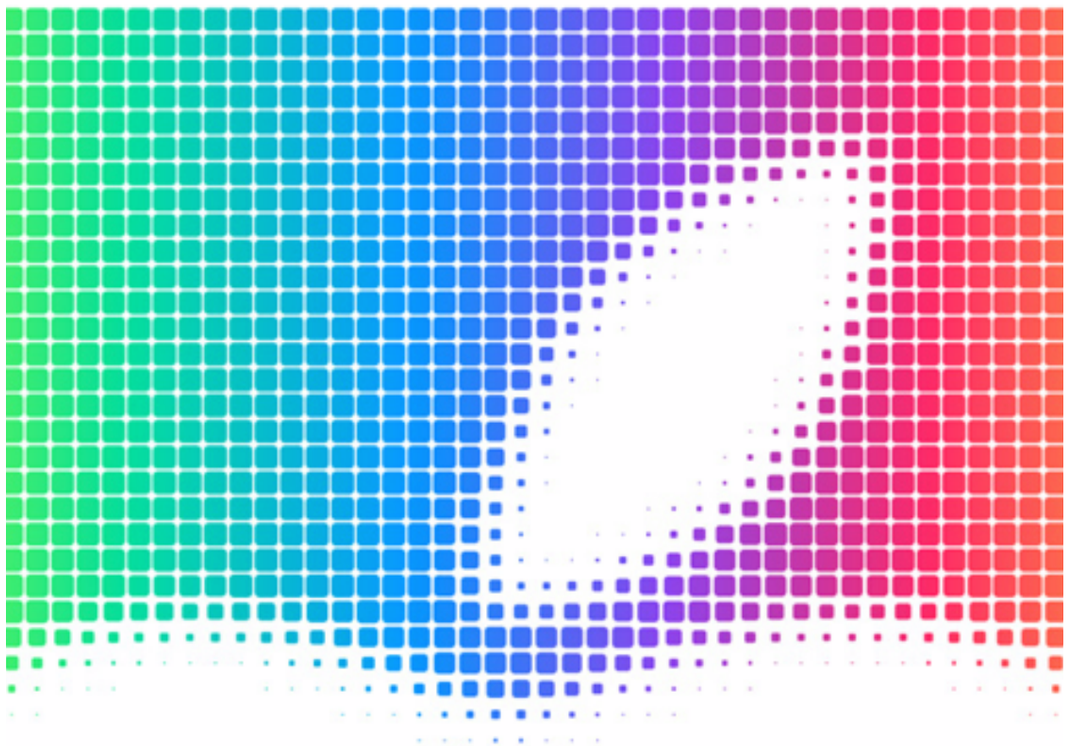
Swift

- enums with associated values
- option types
- pattern matching
- generics
- type inference
- immutability supports
- tuples

from a presentation by @alexisgallagher

WWDC14 Sessions: Swift Language

- Introduction to Swift
- Intermediate Swift
- Advanced Swift



WWDC14 Sessions: Integration with Objective-C

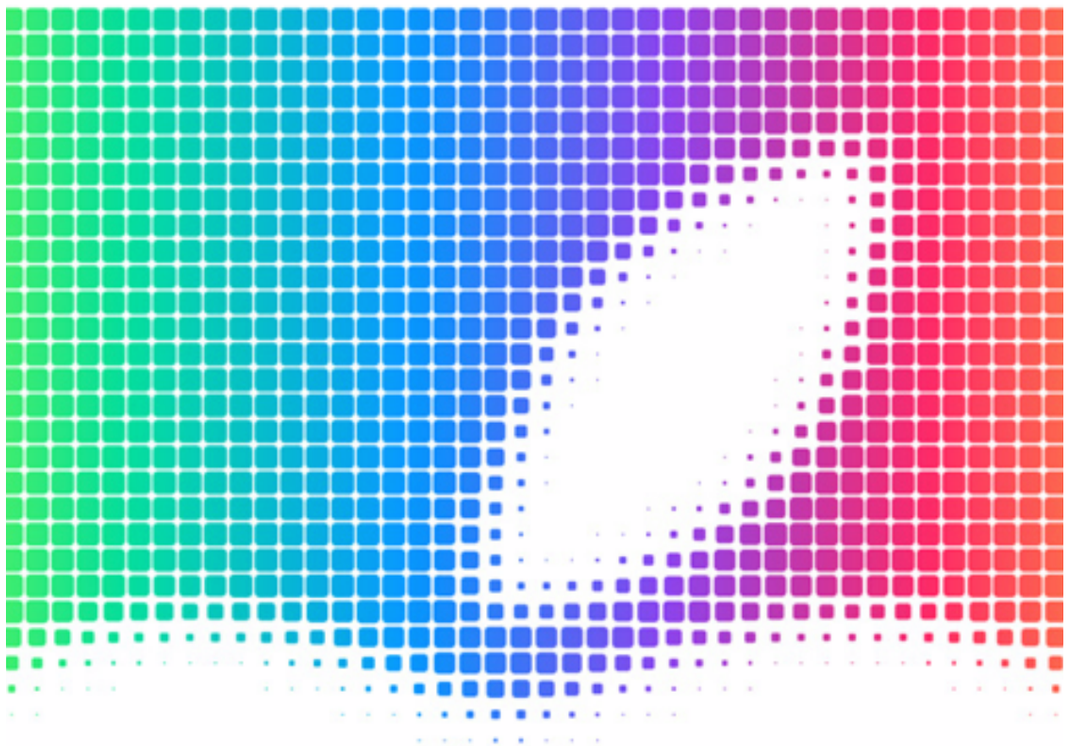
- Integrating Swift with Objective-C

- Swift Interoperability in Depth

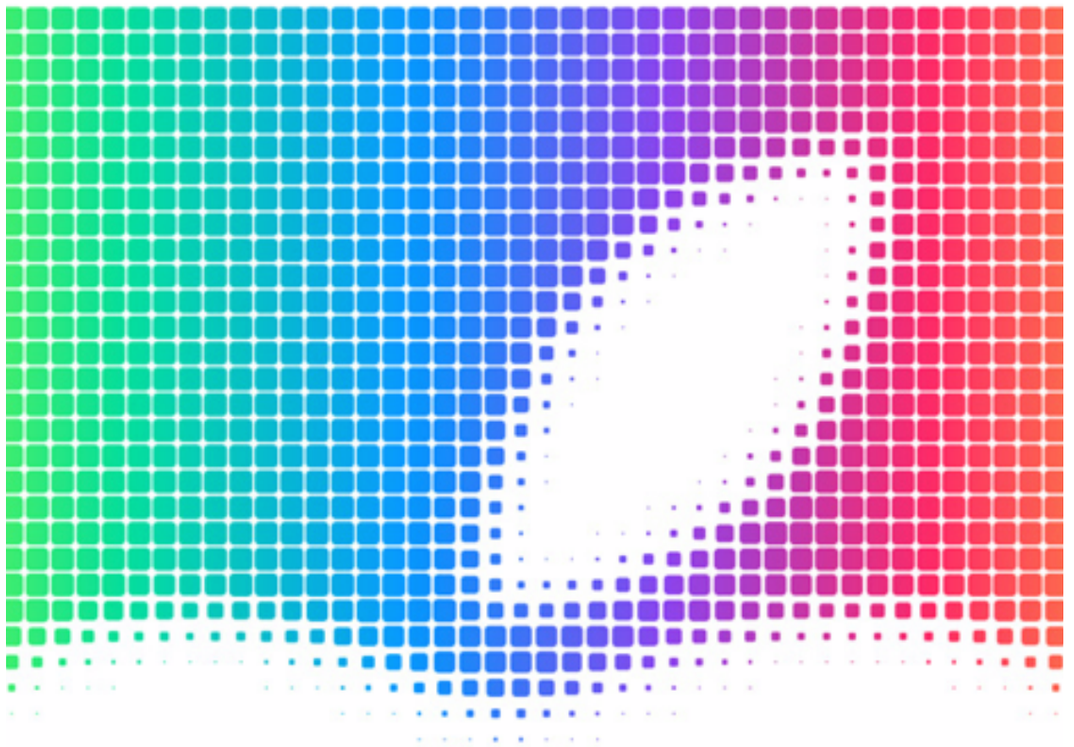
WWDC14 Sessions: Swift Debugging

- Introduction to LLDB and the Swift REPL

- Advanced Swift Debugging in LLDB

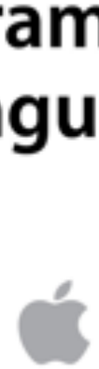


WWDC14 Sessions: Playgrounds



- Swift Playgrounds

The Swift Programming Language



Using Swift with Cocoa and Objective-C

Swift iBooks

Unofficial PDF Versions available

Tutorials

- Ray Wenderlich Swift Tutorials
- Treehouse: An Absolute Beginners Guide to Swift
- iOS Blog: Swift Tutorials
- NSScreencast Swift Videos

Websites

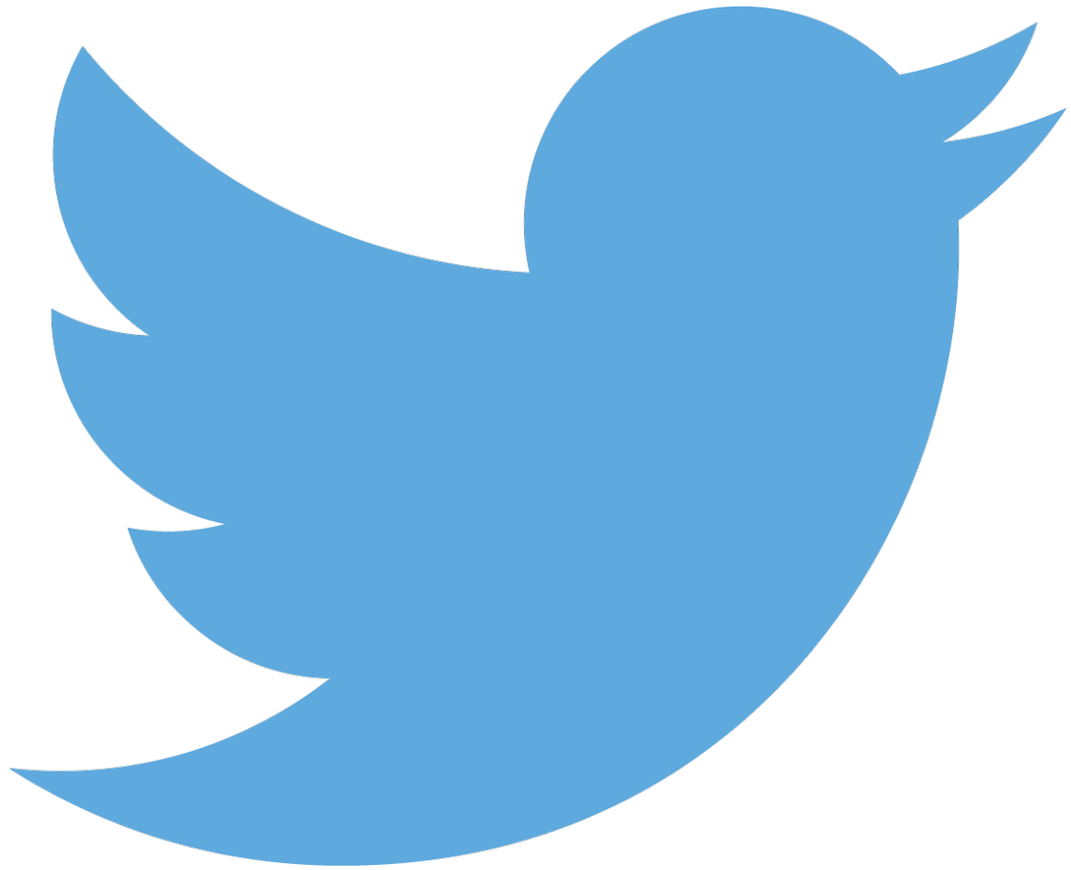
- We Heart Swift
- Code in Swift
- LearnSwift.tips
- Learn Swift
- Learn Swift Online

Blogs

- Apple's official Swift Blog
- Mike Ash
- Evan Swick
- Teton Technical
- Learn Programming iOS

Accounts to Follow

- @clattner_llvm
- @WeHeartSwift2
- @SwiftDevs
- @objectivec
- @SwiftStack_
- @iOSSwift
- @swiftLDN



Objective C

@iOSSwift

Swift vs Objective-C

Swift: **38 days**

Objective-C (with iPhone SDK): 6+ years

Objective-C (with Mac OS X): 13+ years

Objective-C (with NextStep): 20+ years

You have the opportunity to alter the future of iOS

Questions?

@NatashaTheRobot