Swift Hands-on

Introduction

- Swift is a new programming language for **iOS** and **OS X** apps
- Development started in **2010**, boosted in July **2013**.
- Available on iOS 7+ OS X 10.9+

The basics / 1

- ARM and x86-64 native code
- C-Like procedural performance
- Scripting console (REPL, Read-Eval-Print-Loop), available in Terminal
- Implicit namespaces from modules/frameworks
- Integrates with existing objc codebase

The basics / 2

- No headers
- No semicolons
- Multiple return values as tuples
- Extensions and protocols
- Functions as first-class citizens
- Optional arguments
- Closures
- Generics

The basics / 3

- objc headers translated to swift documentation
- Identical object and memory management models than objc
- Directly import existing system APIs
- Mix and match objc and swift in the same project

…but still beta

- Source compatibility between Xcode releases NOT guaranteed
- private/public visibility not implemented yet
- instances of objects sometimes won’t get deallocated
- Swift does not support object initialisers that fail by returning null
- Crashes
- More crashes

Let’s get started

Hello, world

```
println("Hello, world")
```

Simple values

```
var myVariable = 42

myVariable = 50


let myConstant = 42

~~myConstant = 50~~
```

Safe typing

```
var myVariable = 42

myVariable = 50

~~myVariable = “Poney”~~
```

Safe typing

```
var myVariable:Int = 42

myVariable = 50

~~myVariable = “Poney”~~
```

Optionals

```
var myVar:Int?
myVar = 50
if let myDefVar = myVar {
  println(“yo!")
}
```

Strings

```
var myString = “Poney”


var myString = “Pon” + “ey”


var myNum = 42
var life = “meaning is \(myNum)”
```

Arrays

```
var shoppingList = [
  "catfish",
  "water",
  "tulips",
  "blue paint"
]
```

Arrays

```
var shoppingList:String[] = [
  "catfish",
  "water",
  "tulips",
  "blue paint"
]
```

Dictionaries

```
var occupations = [
  "Malcolm": “Captain”,
  "Kaylee": “Mechanic”,
]
```

Dictionaries

```
var occupations:Dictionary<String, String> = [
  "Malcolm": “Captain”,
  "Kaylee": “Mechanic”,
]
```

Functions

```
func greet(name: String, day: String) -> String {
  return "Hello \(name), today is \(day)."
}
```

Functions

```
func greet(name: String, day: String) -> String {
  return "Hello \(name), today is \(day)."
}


func getGasPrices() -> (Double, Double, Double) {
  return (3.59, 3.69, 3.79)
}
```

Functions are first-class citizens and can return other functions

```
func makeIncrementer() -> (Int -> Int) {
  func addOne(number: Int) -> Int {
     return 1 + number
  }
  return addOne
}
```

Generic functions

```
protocol MyProtocol {
  var simpleDescription: String { get }
}

func scaleBySizingFactor<T: MyProtocol>(array:
T[], factor:Double) -> T[] {
  ...
}
```

Swift VS Obj-C / 1

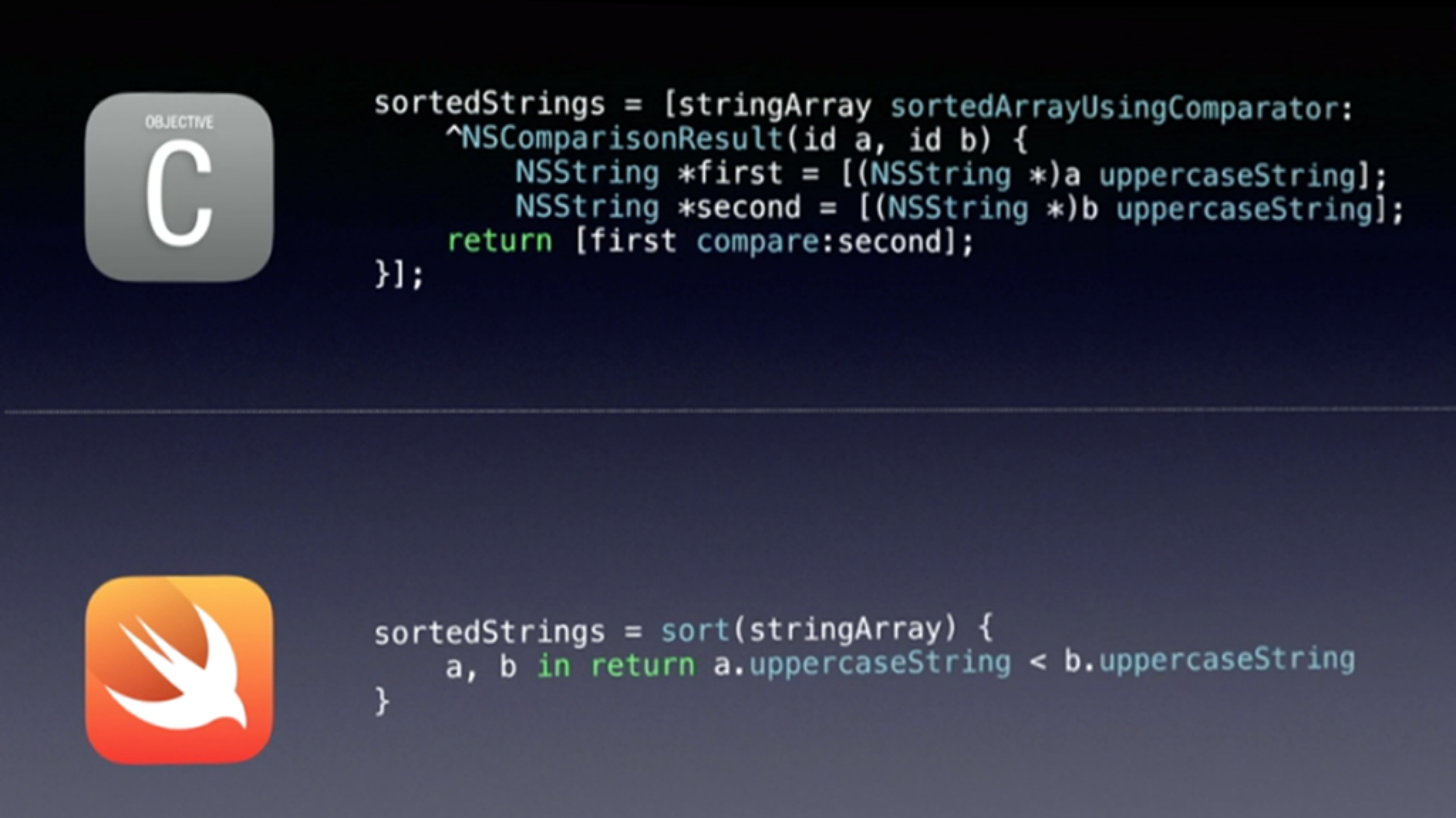
```objc
NSDictionary *dict = @{@"hero":image1, @"balloon":image2};

for (NSString *key in dict) {
    id value = dict[key];
    NSLog(@"%@ %@", key, value);
}
```

```swift
var dict = ["hero":image1, "balloon":image2]

for (key, value) in dict {
    NSLog("\(key) \(value)")
}
```

Swift VS Obj-C / 2



```
sortedStrings = [stringArray sortedArrayUsingComparator:
    ^NSComparisonResult(id a, id b) {
        NSString *first = [(NSString *)a uppercaseString];
        NSString *second = [(NSString *)b uppercaseString];
    return [first compare:second];
}];
```

```
sortedStrings = sort(stringArray) {
    a, b in return a.uppercaseString < b.uppercaseString
}
```

Swift VS Obj-C / 3

```
if ([delegate respondsToSelector:
    @selector(application:willFinishLaunchingWithOptions:)]) {
    [delegate application:app
        willFinishLaunchingWithOptions:options];
}
```

```
delegate.application?(app,
    willFinishLaunchingWithOptions:options)
```