iOS Application Development
Lecture 2: Seminar and Unit 1

Dr. Simon Völker & Philipp Wacker
Media Computing Group
RWTH Aachen University

Winter Semester 2017/2018

http://hci.rwth-aachen.de/ios
Swift
History

• Introduced at WWDC 2014

• Influenced by C and Objective-C

• But designed to be simpler to learn and not dependent on older programming languages

• “Safe, fast, and expressive”

• Open source since 2015
Characteristics

- Clean syntax
- Optionals
- Type inference
- Type safety
- Automatic Reference Counting (ARC)
Characteristics

- Tuples and multiple return values
- Generics
- Fast and concise iteration over collections
- Structs that support methods, extensions, and protocols
- Map, filter, reduce, and other functional programming patterns
- Powerful error handling
Playgrounds

```swift
// Playground - noun: a place where people can play

import UIKit

var str = "Hello, playground"

print("Hello, world!")

"Hello, playground"
"Hello, world!
"
Variables and Constants

• Variables are declared with `var`

```swift
var x = 100
```

• Constants are declared with `let`

```swift
let pi = 3.14
pi = 3.1415926 // error! a constant can’t be changed afterwards
```
1 //: Playground - noun: a place where people can play

2

3 import UIKit

4

5 let x = 1

6 var y = 2

7

8 y = 3

9 x \equiv 4 \quad \text{Cannot assign to value: 'x' is a 'let' constant}

10

11
Type Inference

• Swift automatically chooses the adequate data type for a variable/constant

```swift
var x = 100
x = 99.5 // error! x is of type Integer

var x = 99.5
x = 100 // correct (x = 100.0, type: Double)
```

• You can also explicitly specify the type

```swift
var aString : String
```
Data Types & Type Inference

```
var x1 = 100 // Int
var x2 = 0.5 // Double
var x3 = x1 + x2 // error! can’t add Int and Double
var x3 = Double(x1) + x2 // works! explicit type casting to Double
var x4 = 0.5 + 100 // works! compiler adds before setting the data type
print("x4 = \(x4)") // x4 = 100.5
var 😄 = "LOL" // String
```
Optionals

- By default, variables and constants cannot be nil
- Optionals: variables that can also be nil

```swift
var opt: Int? = 3
opt = nil
```

- Normal variables and Optionals are incompatible to each other

```swift
var number = Int("abc") // nil. Type of Number: Int?
print(number + 3)       // error! Int? != Int
var i: Int = number     // error!
```
Tuples

- Tuples can have multiple elements of different types

```swift
var tuple = (1, 2.0, "hello", true)
var (a, b, c, d) = tuple // a = 1, ...
print(tuple.2) // "hello" (starts at 0)
```
//: Playground - noun: a place where people can play

import UIKit

let x = 1
var y = 2

y = 3
// x = 4

var tuple = (1, 2.0, "hello", true)
var (a,b,c,d) = tuple
print(tuple.2)
print(b)

print(tuple.)

Int 0
Double 1
String 2
Bool 3
Control Flow

- If/else

```swift
var x = 3
if x < 0 {
    print("x is negative")
} else if x == 0 {
    print("x is zero")
} else {
    print("x is positive")
}
```

- Ternary Operator

```swift
var largest: Int
let a = 15
let b = 4
if a > b {
    largest = a
} else {
    largest = b
}
largest = a > b ? a : b
```
Control Flow

• Switch

```
let pt = (0.0, 0.0)

switch pt {
    case (0,0):
        print("Origin.")
    case (_,0):
        print("On x-axis.")
    case (0, _):  
        print("On y-axis.")
    default:
        print("Elsewhere.")
}
```

```
switch distance {
    case 0...9:
        print("You are close")
    case 10...500:
        print("Take a car")
    default:
        print("Too far away.")
}
```
Development Environment
Xcode
Playground Demo

1 //: Playground – noun: a place where people can play
2
3 import UIKit
4
5 var str = "Hello, playground"
6 print("Hello, world!")
7
8

"Hello, playground"
"Hello, world!

Hello, world!
• 5 areas

1. Editor
2. Toolbar
3. Navigator
4. Debugging
5. Utility
.xcodeproj File

- Settings file for your project
Building/Running

What do you want me to say?
Say "Hello!"
Warnings & Errors

- Warnings don’t prevent your app from compiling & running
  - Code that never gets executed
  - Variable that does not change
  - Deprecated code
- Errors prevent the app from building
  - Invalid code (typo, variable declaration, function calling)
  - Xcode often provides suggestions & fixes
Debugging

- Set breakpoints for execution on simulator and device
- Continue, Step over, Step into, Step out
• Quick Help (Option+Click)
• Documentation Browser
• Programming Guides
Interface Builder

- Visually define your UI
Outlets & Actions

• Connect your UI elements with your code: Right-click + Drag

```swift
// The view, typically from the view, typically from
// print("Hey iOS class")
print("This is the Editor")
print("To the left is the Navigation Bar")
print("The Toolbar is on top of the Navigation Bar")
print("The Debugging area is")
print("To the right is the Debugging area")

override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Dispose of any resources recreated.
}
```
Outlets & Actions

- **IBOutlet**
  - Access the UI element from code
  ```swift
  @IBOutlet weak var.textLabel: UILabel!
  ```

- **IBAction**
  - Receive UI events
  ```swift
  @IBAction func buttonPressed(_ sender: Any) {}
  ```