Pace University - PCBT

Advanced iOS Class Spring 2015

SWIFT INTRODUCTION

- This lecture will provide a background and context for using the SWIFT language in iOS.
- We will discuss the history, purpose, and features.
- This lecture will also summarize the key language grammar and technical aspects.
- The lecture should be accompanied by an XCode playground exercise.

What is Swift?

- Exciting because departure from legacy languages – does not go back 20 or 30 years before modern machine architectures and problems to be solved.
- Written from scratch, no legacy issues, a clean language.
- Written by Chris Lattner Apple's lead compiler engineer and principal developer of LLVM and clang compilers.
- Borrows good ideas from Rust, Haskell, Ruby, Python, others.
- Considered a Safe language, should help you have less bugs.

Some Basics

- Classes are still the main structure used.
- Code is compiled.
- Type names start with a capital letter such as MyClass or ViewController.
- Variable names start with lower case such as dataSource or userInteractionEnabled.

Simple class that has only properties

```
class ClassStoringProperties {

    let constantProperty: String = "never changes"
    var variableProperty: Int = 1

    let constantPropertyInfer = "also never changes"
    var variablePropertyInfer = 1
}
```

Mutable and Immutable properties

- let – immutable property
- var – mutable property
- SWIFT will infer (guess) a type if you initialize
- You don't technically have to initialize a property, but lets not worry about that for now, assume you do.

Computed Properties

- Properties calculated by a block of code.
- Can use like normal properties

Computed Property Class

```
class ClassCalculatingProperties {
    var height: Double = 0.0
    var width: Double = 0.0
    var area: Double {
        get {
            return height*width;
        }
        set(newArea) {
            width = sqrt(newArea)
            height = width
        }
    }
}
```

Use of a Computed Property

```
let calculatingObject = ClassCalculatingProperties()
calculatingObject.height = 9.0
calculatingObject.width = 5.0
let area = calculatingObject.area
```

Property Observers

- Can set observer code that is executed when a property is set.
- didSet is called after a property is updated
  - the parameter contains the previous value
- willSet is called before a property is updated
  - the parameter is the new value to be set

Example Property Observer

```
class ObservantView: UIView {
    var text: String = "" {
        didSet(oldText) {
            if text != oldText {
                setNeedsDisplay()
            }
        }
    }
}
```

Methods

- Different names same thing.. Functions, instance Method, instance Function, member Function.
- Has parameters and return value.

Example Method and Use

```
func pokeUser(name: String, message:String) -> Bool {
        return true
}

let firstClassObject = MyFirstClass()
let result = firstClassObject.pokeUser("bob",
message:"hi")
```

Class Methods

- You don't need an object reference for this

```
class func timeRemaining() -> Double {
      return 5.5
}

let remaining =
MyFirstClass.timeRemaining()
```

External and Internal Parameter Names

- External First
- Internal Second
- External is used when your calling the method.
- Internal is used inside the method implementation.
- These are optional.
- Makes it similar to Objective-C.

Example Internal and External

```
func sendMessage(message:String, toPerson:String) {
}

func poke(user name:String, say message: String,
important priority:Bool) -> Bool {
        if priority {
            sendMessage(message, toPerson:name)
            return true
        }
        return false
 }

let didPoke = firstClassObject.poke(user:"bob",
say:"yes", important:true)
```

Inheritance

- Very similar to Java and Objective-C
- Use the override keyword to override existing methods.
- super keyword is supported.
- You can also override properties.

Simple Inheritance Example

```
class BaseClass {
    func pokeUser() {
        println("poke in the base class"); }
    func messageUser() {
        println("message in the base class"); }
}
class DerivedClass: BaseClass {
    override func pokeUser() {
        super.pokeUser()
        println("poke in the derived class”) }
}
let someClass = DerivedClass()
someClass.pokeUser()
someClass.messageUser()
```

Some Basic Types

- Int
- UInt
- Double
- Float

Some Literals

- 123
- -86
- 0x1234abbd
- 0b01011111
- 1.5

Loops

- Allows iteration through a range of integer values
- Allows iteration through a collection
- Each iteration allows access to a variable of the current iteration

Simple For Loop

```
func forLoop() {
        for i in 0..<10 {
            println("looping \(i)")
        }
}
```

Arrays and Dictionaries

- Both are considered SWIFT collections
- Arrays are ordered lists.
- Dictionaries are name value pairs with a key for lookup of the values.
- They are mutable.

Loop through Collection

```
let weather = ["rain", "snow", "sleet"]
    func forArrayLoop() {
        for percip in weather {
            println(percip)
        }
}
```

Loop through Dictionary

```
let capitals = ["new york":"albany",
        "new jersey":"trenton",
        "california":"sacremento"]
    func forDictionaryLoop() {
        let someValue = capitals["new jersey"]
        for caps in capitals {
            println("\(caps.0) captial city is \
(caps.1)")
        }
}
```

End of Lecture

rpascazio@pace.edu