GENERAL ASSEMBLY

MOBILE DEVELOPMENT

POWERING INTERFACES WITH CODE

William Martin
VP of Product, Floored

Angel X. Moreno
EIR, Developer

LEARNING OBJECTIVES

- Create hooks from UI Elements to Swift code.
- Identify and implement methods associated with view controllers and UI elements inherited from their superclasses.
- Recognize and implement different view controllers.
- Present views programmatically.
- Point out ability to access Xcode documentation for any external classes.
- Identify how to access documentation for IB elements and controllers.
- BONUS Apply special Gesture handlers to enliven apps with interaction.

REVIEW

WHERE HAVE WE BEEN?

VALUES AND TYPES

```
// Integers
–1, 0, 1, 2, 3
// Doubles (and Floats)
3.14159265358979
// Bools
true and false
// Strings
“Hello, Toshi!”
```

SYNTAX: VARIABLES

```
// Declare and initialize
var myName : String = “Toshi”

// Assign a variable
myName = “Toshi the Shiba”
```

SYNTAX: CONDITIONALS (IF-STATEMENTS)

```
if x > 0 {
    println(“positive”)
} else if x < 0 {
    println(“negative”)
} else {
    println(“zero”)
}
```

SYNTAX: LOOPS (WHILE)

```
var x = 1
while x < 15 {
    println(“x is \(x)”)
    x++
}
```

SYNTAX: LOOPS (FOR)

```
for (var x = 0; x < 10; x+=2) {
    println(“x = \(x)!”)
}

for x in 0..<10 {
    println(“x = \(x)!”)
}
```

SYNTAX: FUNCTIONS

```
func isPrime(number:Integer) -> Boolean {
    for i in 2..<number {
        if number % i == 0 {
            return false
        }
    }
    return true
}
```

SYNTAX: CLASSES

```
class Dog {
    var name : String
    init(name:String) {
        self.name = name
    }
    func bark() {
       println(“\(self.name) says BARK!”)
    }
}
```

SYNTAX: CLASSES

```
// Instantiate
var myDog : Dog = Dog(name:“Toshi”)

// Access properties and methods
println(myDog.name)
myDog.bark()
```

SYNTAX: EXTENDING CLASSES

```
class Shiba : Dog {
    var temperament : String = “stubborn”
    override func bark() {
       println(“Shibas don’t bark.”)
    }
}
```

REVIEW LESSON 6

INTERFACE BUILDER + CODE

WHERE DOES CODE LIVE IN AN IPHONE APP?

- View Controllers (subclasses of UIViewController)
- AppDelegate
- Custom Views
- Custom Classes (your own .swift files)

LET’S TAKE A TOUR OF VIEWCONTROLLER.SWIFT

- Create an Xcode project.
    - Template: Single View Project
    - Disable Autolayout in Main.storyboard
- ViewController.swift includes the code behind the ViewController in the storyboard.
- ViewController is a subclass of UIViewController, provided by UIKit.
- All UIViewControllers have a .view property that includes the base view being rendered in IB and when the app starts.

INTERFACE BUILDERS'S (IB) FUNCTIONS

- Events dragged into code from IB define **IBAction methods**.
- Outlets dragged into code from IB define **IBOutlet properties**.
- These IBActions can use your class's properties, methods, and IBOutlets to make things happen in your app.
- Do you have a text field, label, or text view for which you've made an IBOutlet?
  - You can get its text by using: `var text = element.text`
  - You can set its text by using: `element.text = “Some text”`

XCODE: IBACTION / IBOUTLET DEMO

POWERING INTERFACES WITH CODE

EXERCISE

KEY OBJECTIVE(S)

Hook interface builder into Swift code and update interface builder from swift code.

TIMING

30 min	1. Code with partner
5 min	2. Debrief

DELIVERABLE

Create a new app with two screens:
- One has a button, a text field, and a text label, when the button is pressed, make the label say: `"Hello, \(textFromTextField)!"`
- The second has two text fields, a button, and a text label. When the button is pressed, make the label print the sum of the ints in the text fields.

Bonus: Make the keyboards on the second screen numerical; display an error if the number is invalid. Perhaps change the .textColor to red for the label as well.

POWERING INTERFACES WITH CODE

EXERCISE

KEY OBJECTIVE(S)

Identify and create classes.

TIMING

40 min	1. Code with partner
5 min	2. Debrief

DELIVERABLE

Create a Math class that can add two numbers, multiply two numbers, divide two numbers.

Create an app that contains two text fields and several buttons, representing math operations. Hook this up to your Math class.

DOCUMENTATION DEMO