MY FIRST iOS APP

(PART 3) - Outlets

Prepared by Farhaj Ahmed

WHAT ARE FRAMEWORKS?

UIKit and other frameworks

iOS offers a lot of building blocks in the form of frameworks or "kits." The UIKit framework provides the user interface controls such as buttons, labels and navigation bars. It manages the view controllers and generally takes care of anything else that deals with your app's user interface. (That is what UI stands for: User Interface.)

If you had to write all that stuff from scratch, you'd be busy for a long while. Instead, you can build your app on top of the system-provided frameworks and take advantage of all the work the Apple engineers have already put in.

Any object you see whose name starts with UI, such as `UIButton`, comes from UIKit. When you're writing iOS apps, UIKit is the framework you'll spend most of your time with, but there are others as well.

Examples of other frameworks are Foundation, which provides many of the basic building blocks for building apps; Core Graphics for drawing basic shapes such as lines, gradients and images on the screen; AVFoundation for playing sound and video; and many others.

The complete set of frameworks for iOS is known collectively as Cocoa Touch.

HOW VIEW DID LOAD() WORKS?

Instance Method

viewDidLoad()

Called after the controller's view is loaded into memory.

Declaration

```
func viewDidLoad()
```

Discussion

This method is called after the view controller has loaded its view hierarchy into memory. This method is called regardless of whether the view hierarchy was loaded from a nib file or created programmatically in the `loadView()` method. You usually override this method to perform additional initialization on views that were loaded from nib files.

ENOUGH TALK, LET'S JUMP TO <CODE/>

SCOPES

When I first introduced variables, I mentioned that each variable has a certain lifetime, known as its *scope*. The scope of a variable depends on where in your program you defined that variable.

There are three possible scope levels in Swift:

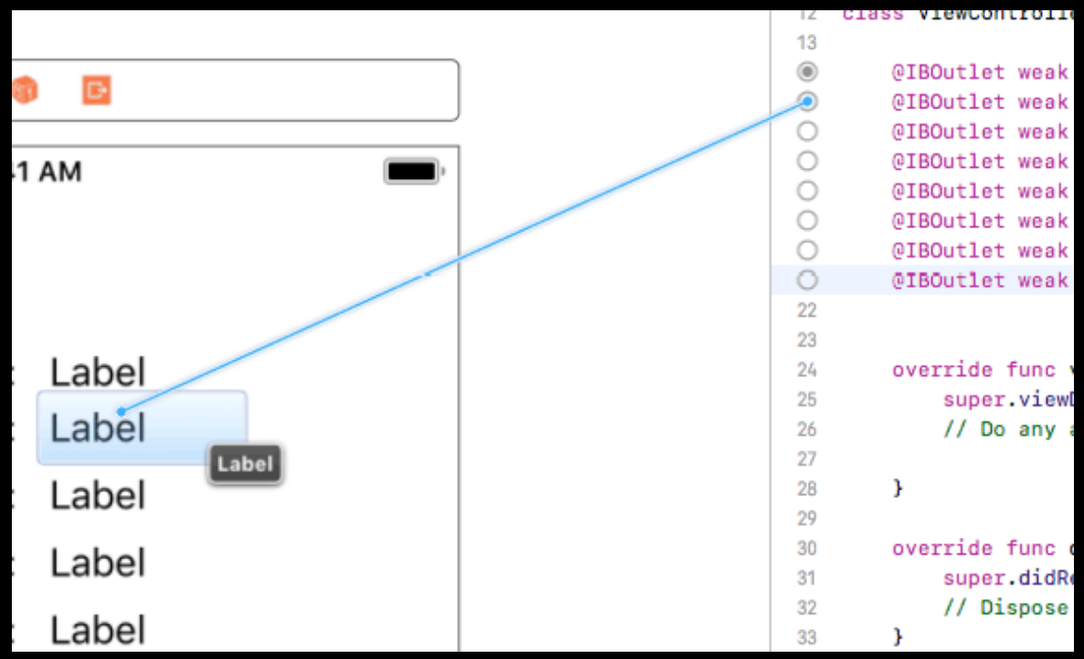
1. **Global scope**: These objects exist for the duration of the app and are accessible from anywhere.

2. **Instance scope**: This is for variables such as `currentValue`. These objects are alive for as long as the object that owns them stays alive.

3. **Local scope**: Objects with a local scope, such as the `slider` parameter of `sliderMoved()`, only exist for the duration of that method. As soon as the execution of the program leaves this method, the local objects are no longer accessible.

WHAT ARE OUTLETS?

`@IBOutlet` is a connection from an Interface Builder user interface component – e.g. a **`UIButton`** – to a property in a view controller or other piece of Swift code. To the left of the code you should see a black circle with a ring around it, which is Xcode's visual confirmation that a given **`@IBOutlet`** has an active connection.




```
11  class ProfileHeaderContentView: UIView {
12      // MARK: - Properties
        @IBOutlet weak var profileBackgroundImageView:
            UIImageView!
        @IBOutlet weak var profileImageView: UIImageView!
        @IBOutlet weak var nameLabel: UILabel!
        @IBOutlet weak var screennameLabel: UILabel!
        @IBOutlet weak var descriptionLabel: UILabel!
        @IBOutlet weak var locationLabel: UILabel!
        @IBOutlet weak var followingCountLabel: UILabel!
        @IBOutlet weak var followerCountLabel: UILabel!
        @IBOutlet weak var tweetCountLabel: UILabel!
```

YOU CAN FIND THIS iOS APP's SOURCE CODE, LECTURE SLIDE AND MY FIRST iOS APP PART 3 INSTRUCTION GUIDE PDF in the description of my YouTube Channel.

Subscribe now to my YouTube Channel:

http://www.youtube.com/c/FarhajAhmed

YOU CAN FIND ME @:

farhaj.ahmed@live.com

https://www.linkedin.com/in/farhajahmed1

https://www.facebook.com/LearnFromFarhaj

THANK YOU