GENERAL ASSEMBLY

MOBILE DEVELOPMENT
DESIGN PATTERNS IN IOS

Rudd Taylor
Founder, SALT

LEARNING OBJECTIVES

- Identify iOS design patterns and how they are used in our apps
- Define delegation and implement delegates in our apps
- Define notifications and show how to post and observe notifications
- Implement NSNotificationCenter notifications that already exist in our apps
- Identify best practices for using delegation vs notifications

REVIEW OF IOS ARRAYS, DICTIONARIES, CLOSURES

WHAT IS A DESIGN PATTERN

- A design pattern is a reusable pattern to solve common issues that come up in software development
  - ***NOT*** new syntax
- An attempt to look at common issues that pop up
- A pretty generic definition (because ‘design pattern’ is a pretty generic term)
- iOS has several such patterns

NEW SYNTAX - PROTOCOLS

Protocol is NOT a type of class.
Each type can only have one class.
class ViewController (Class):
UITableViewController,
UITableViewDataSource,
UITAbleViewDelegate (Protocol)
Does not define actions; functions
without direction.

- Like a superclass, but doesn't specify behavior
  - Only methods signatures (just the 'func' line) and variables
  - ***NO*** implementation of any methods
- If a class ***meets*** a protocol, it has all of the methods and variables the protocol specifies
- Used when a class needs to know what methods something has
- Protocols can be used as types, just like classes and structs
- When we have a variable that has a protocol type, we can use all the variables / methods that the protocol specifies (just like a class or struct)
- Classes can meet as many protocols as they like

THE DELEGATE

- The delegate is a relationship between two classes instances. One instance has a delegate variable which refers to an instance that has certain methods (***meets a protocol***). The is the original class's ***trusted friend***
  - E.g. UITableView has var delegate: UITableViewDelegate?
- Instances tell their delegates information about when things happen to them
  - Or they get critical information from them
  - Many of Apple's classes do, e.g. UITableView, UITextField, UINavigationController
- A class has a delegate when it wants to delegate some behavior to another class
  - E.g. UITextField's delegate gets called when a text field text changes, the user presses return, etc
- Classes may have ***one*** delegate

DELEGATE

CODE-ALONG

ACTIVITY

- Modify the todo list app we've been creating to accept and display multiple values for each todo item
    - Each todo item should have a name, status and due date, all strings
    - Store each of the todo items as a dictionary
    - First display the TODO item parameters in your table view cells
    - Then change your 'add' modal dialog to accept the new parameters
    - Then change your app to actually save those parameters once saved
    - Bonus: On pressing 'return' in the last text box in the modal, trigger an 'add' (hint: UITextFieldDelegate)
    - Bonus: Add the capability to remove a todo item

DESIGN PATTERNS

NOTIFICATIONS

NOTIFICATIONS

- Another pattern seen in iOS
- Any instance can ***post*** a notification to NSNotificationCenter.defaultCenter()
- Any instance can ***subscribe*** to the notifications coming out of NSNotificationCenter
- Multiple things can subscribe to the same kind of notifications
- Notifications are identified with strings
- Why?

NOTIFICATIONS

- Another pattern seen in iOS
- Any instance can ***post*** a notification to NSNotificationCenter.defaultCenter()
- Any instance can ***subscribe*** to the notifications coming out of NSNotificationCenter
- Multiple things can subscribe to the same kind of notifications
- Notifications are identified with strings
- Why?
  - Things that post notifications don’t have to know about who listens to them
  - Things that listen to notifications don’t have to know about who posts them, or if they ever get posted
  - An ***abstraction*** between two things
- Apple uses this for keyboard notifications, battery low, memory low, text field changes, etc

NOTIFICATION

CODE-ALONG

ACTIVITY

- Once finished with the previous tasks, create a label on your main view controller that says ‘todo added!’
- Set its alpha to 0, initially // someView.alpha = 0
- When your add view controller adds a new todo, it should post a notification
- Your main view controller should subscribe to that notification and make that view visible when it is posted
- Bonus: Animate the view coming into view, look at UIView.animateWithDuration
- Bonus: Animate the view coming into view, then animate it out of view. Look at UIView.animateWithDuration (with completion)