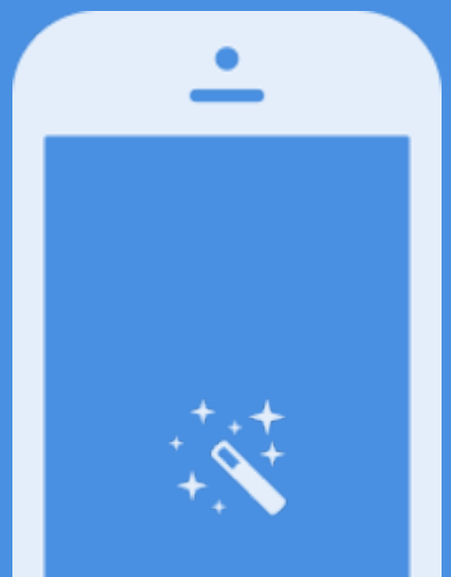
iOS Crash Course

Session Five

Janum Trivedi
iOSCrashCour.se



Session 4 Overview

- We started our to-do app project
- Introduced UITableView and UITableViewCell, along with the cell re-use pattern
- Delegates (UITableViewDelegate and UITableViewDataSource)
    - cellForRowAtIndexPath, numberOfRowsInSection

Session 5 Overview

- Understanding the application lifecycle
- We're going to keep working on our app
- Creating an Item class to contain more data
- Create our own custom UITableViewCells
- Customize the appearance of our view controller
- Use UIGestureRecognizers to support "completing" an item

Before we start

- Questions from last week?
- This week is another hands-on day, very few slides
- If your project doesn't compile (or you weren't here for Session 4), download the project from the S4 GitHub repo so we're all on the same page

Before we start

- Office hours?
  - I'll send out a poll with times if there's interest
    - Likely be on the weekend
- Quick review on UITableView

Table Views

- Table views live in the V in MVC (they are Views)
- We give the table view some data, and it will present it
- ex., we could have an NSArray of strings and tell our table view to populate itself with that data
    - That’s what we’ll be doing today!

Table View Structure

- One column, many rows
- Each row contains one cell (UITableViewCell)
- Indexed (base 0)

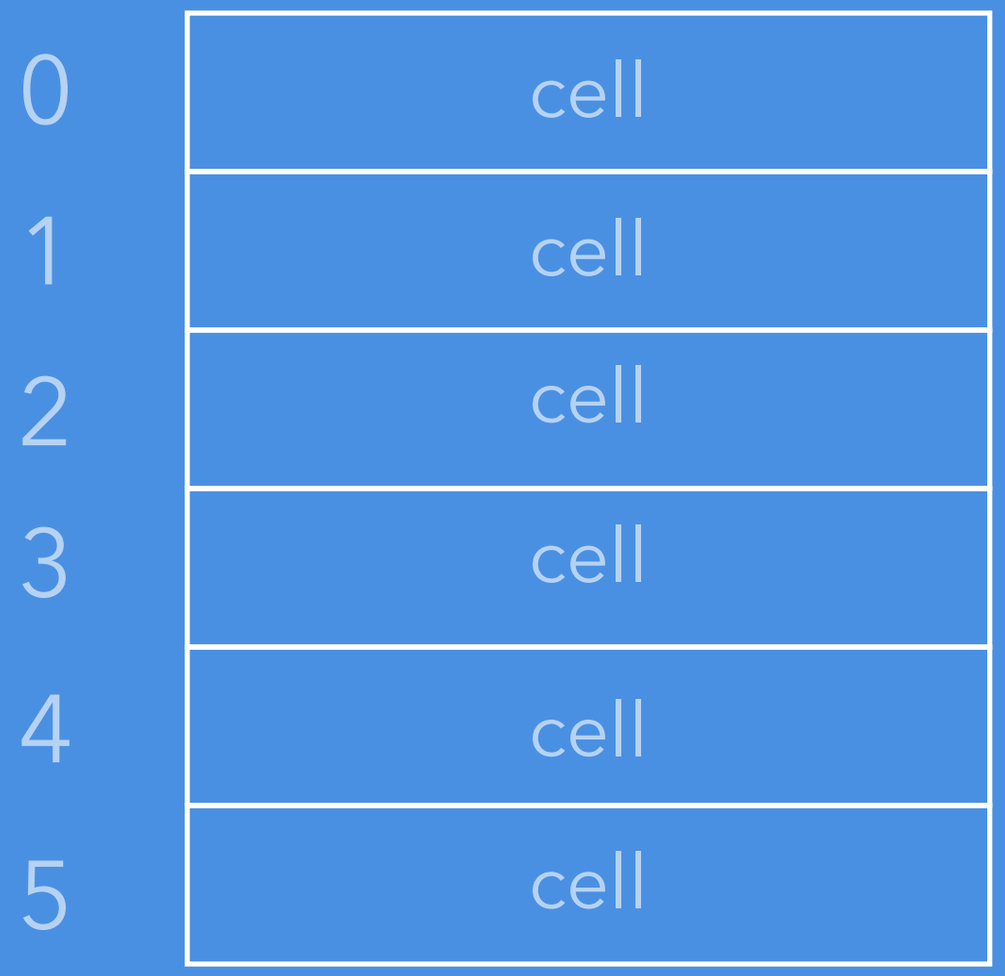


Table View Structure

- One column, many rows
- Each row contains one cell (UITableViewCell)
- Indexed (base 0)

Table Views

- We need some way of actually telling the table view *what* to present
- Our Controller (UIViewController) will be the one to tell our table view about that

Delegation

- Core concept in iOS like MVC
- This is how our table view will know about its data
- Delegation is a way of “delegating” responsibility to another object or class
- Abstract idea right now, but it’ll make sense

Delegation

- We can tell a class to “subscribe” to some events of another class
- ex., “When *this* button is tapped, let *that* class know about it, so it can respond
- A class is said to **conform** to a **protocol**

Delegation

- Once we create a UITableView property in ViewController and connect it to the Storyboard, we will set ViewController as the *delegate* of the tableView

- Our tableView will *ask its delegate* (ViewController) how many rows it should show, what to put on the cells, etc.

- **Our table view will ask its delegate for information**

App Lifecycle

- The application lifecycle describes what exactly happens when we start our app for the first time
- Starting with the very first function call to main() (in main.m) to our view controller's viewDidLoad method
- Right now, the process is a bit opaque, so let's go through it

App Lifecycle

- User taps app icon
- The first file that is loaded is main.m, which contains our special main function
- Same with all C-family programs
- Reserved function that marks the first execution point in a program

```
//  Copyright (c) 2015 Janum Trivedi. All rights reserved.
//

#import <UIKit/UIKit.h>
#import "AppDelegate.h"

int main(int argc, char * argv[]) {
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));
    }
}
```

- We will never modify main

- Makes 1 call to UIApplicationMain, which is a pre-defined function in UIKit

- Creates an instance of UIApplication and performs all the basic app initialization

- UIApplication creates a main run loop (NSRunLoop) which keeps our app alive and responding to events

```
//  Copyright (c) 2015 Janum Trivedi. All rights reserved.
//

#import <UIKit/UIKit.h>
#import "AppDelegate.h"

int main(int argc, char * argv[]) {
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));
    }
}
```

- UIApplicationMain also loads the app's main interface file (as defined in Info.plist)

- If using storyboards, this will be our Main.storyboard

- Note at this point, nothing is shown on-screen yet– our interface is loaded into memory, but we have not initialized our window

```
8
9  #import <UIKit/UIKit.h>
10
11 @interface AppDelegate : UIResponder <UIApplicationDelegate>
12
13 @property (strong, nonatomic) UIWindow *window;
14
15
16 @end
17
18
```

- We may need to perform some application-level setup before showing our main view

- How? Delegation

- The app delegate handles creating the UIWindow property that “backs” all subviews on screen

```objc
#import "AppDelegate.h"

@interface AppDelegate ()

@end

@implementation AppDelegate


- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
    {
    // Override point for customization after application launch.
    return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application {
    // Sent when the application is about to move from active to inactive state. This can occur for certain
        types of temporary interruptions (such as an incoming phone call or SMS message) or when the user
        quits the application and it begins the transition to the background state.
    // Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame rates.
        Games should use this method to pause the game.
}
```

- Our app delegate (which conforms to the UIApplicationDelegate protocol) is called at critical lifecycle points (first launch, resigned active, terminated, etc.)

- didFinishLaunchingWithOptions is called whenever our app is first loaded into memory

- Finally, our Initial View Controller is loaded into memory
  - Specified in our Storyboard
  - Can be a UIViewController or parent UINavigationController
- The first method that's called in a UIViewController subclass is the special viewDidLoad method
  - Used for handling one-time view setup

- viewDidLoad is a part of the UIViewController lifecycle family of methods

- Called just once– when the VC is allocated memory and initialized for the first time

- There's also viewWillAppear, which is called **every** time the VC reappears

- viewDidDisappear when leaving the VC, viewDidUnload (counterpart to viewDidLoad, refers to unloading the memory addresses)

Xcode Time