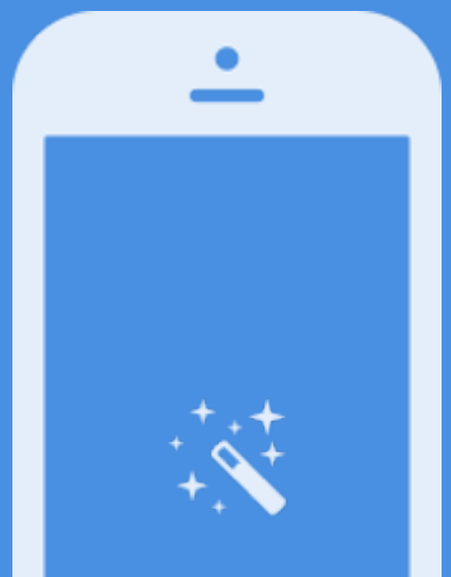
iOS Crash Course

Session Four

Janum Trivedi
iOSCrashCour.se



Session 2 Overview

- Creating classes
- Writing our own methods
- Model-View-Controller design pattern
- Interactivity with IBOutlets and IBActions

Session 3 Overview

- Fairly complex topics that ties many things we've learned so far together
  - "Component based" realized
- Very few slides this Session, lots of hands-on work

Followup and Questions

- Update on Lecture Recordings
    - We just got Session 1, Session 2 did not get recorded :(
    - However, the slides of Session 2 hold most of the information, and those are available
    - Session 3 coming in a day or two
    - Will update the website

Session 3 Overview

- Today we start our project!
- The next few sessions, we'll be building a to-do list app
- Starting with table views today, and later building on all the functionality we would expect from a to-do app

Table Views

- Primary focus today is creating table views (UITableView)
- They are everywhere
  - Contacts.app, Mail.app
  - Any time you see multiple rows of data presented in an iOS app, it's a UITableView

Table Views

- **Extremely** powerful class
- You can do even more complex things with table views– even the Instagram main feed is a table view with some image views and labels
- We are starting this today, but will likely revisit after spring break as well

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)

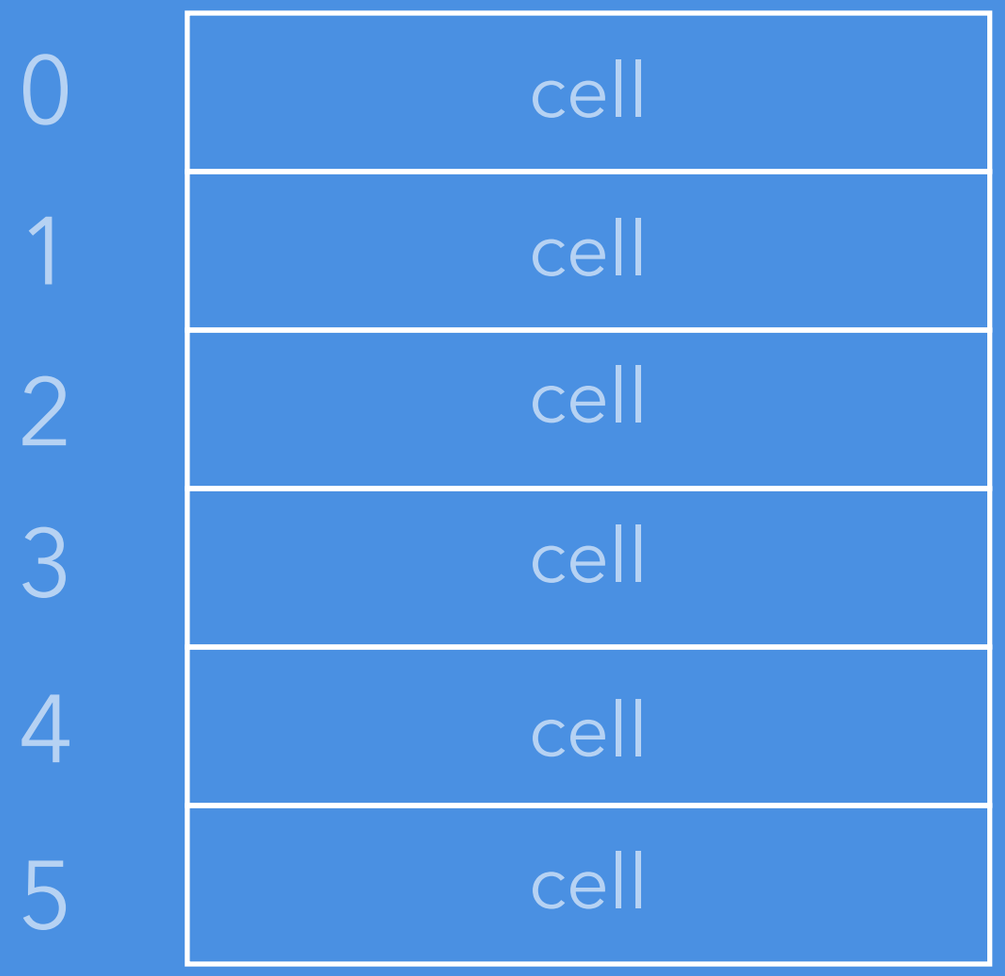


Table View Structure

- 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**

Xcode Time