iOS Dev Accelerator
Week 1 Day 2

- View Controllers
- TableViews and Delegation
- App Delegate
- Navigation Controllers

ViewControllers

- Basic Visual Component of iOS.
- Everything onscreen is managed by ViewControllers and their subviews.They are the link between your model layer and view layer.
- iOS provides many built-in ViewControllers to support standard interfaces.
- The most important property of a ViewController is it’s view property.

ViewControllers

- Really only 4 ways to get a ViewController's view on screen:
    1. Make the ViewController a window's rootViewController.
    2. Make the ViewController a child of a container View Controller
    3. Show the ViewController in a popover
    4. Present it from another View Controller (By far the most common scenario)

ViewControllers

- There are many ways ViewControllers can communicate with each other, and this particular topic is often hard to grasp at first.
- Delegation and notifications are common techniques, but an even simpler way is just passing a reference to the necessary model object that a particular child ViewController will need.
- Remember, ViewController are objects just like any other object in Swift, they can be held in Arrays or Dictionaries, and passed around in methods or held as properties.

ViewControllers

- **The concept of ViewController's lifecycle is important to understand. So important we bolded it.**
- There are 4 methods to know:
    1. ViewDidLoad - called when the ViewController's view is loaded into memory
    2. ViewWillAppear - called before the ViewController's view appears onscreen
    3. ViewDidAppear - called immediately after the ViewController's view has appeared onscreen
    4. ViewWillDisappear - called when the ViewController's view is about to be removed from the view hierarchy
- And 2 methods that are related to subview layout:
    5. viewWillLayoutSubviews - Called before the ViewController lays out the subviews
    6. viewDidLayoutSubviews - Called after the ViewController lays out subviews

UITableView and Delegation

Together forever.

TableViews

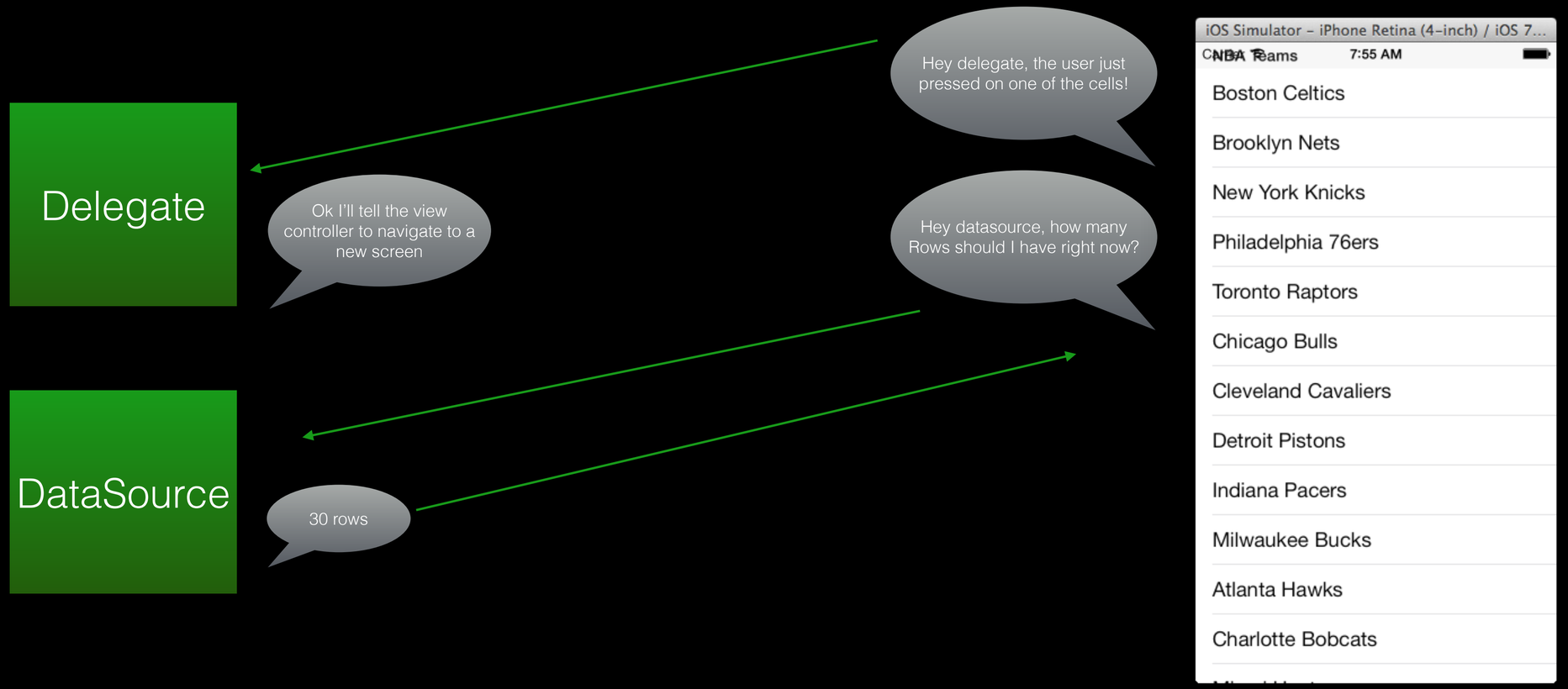
- “A Tableview presents data in a scrollable list of multiple rows that may be divided into sections.”
- A Tableview only has one column and only scrolls vertically.
- A Tableview has 0 through n-1 sections, and those sections have 0 through n-1 rows. A lot of the time you will just have 1 section and its corresponding rows.
- Sections are displayed with headers and footers, and rows are displayed with Tableview Cells, both of which are just a subclass of UIView.

So how do Tableviews work?

- Tableviews rely on the concept of delegation to get their job done.

- Picture time:

TableViews and Delegates



A tableview has 2 delegate objects. One actually called delegate and one called datasource. The datasource pattern is just a specialized form of delegation that is for data retrieval only.

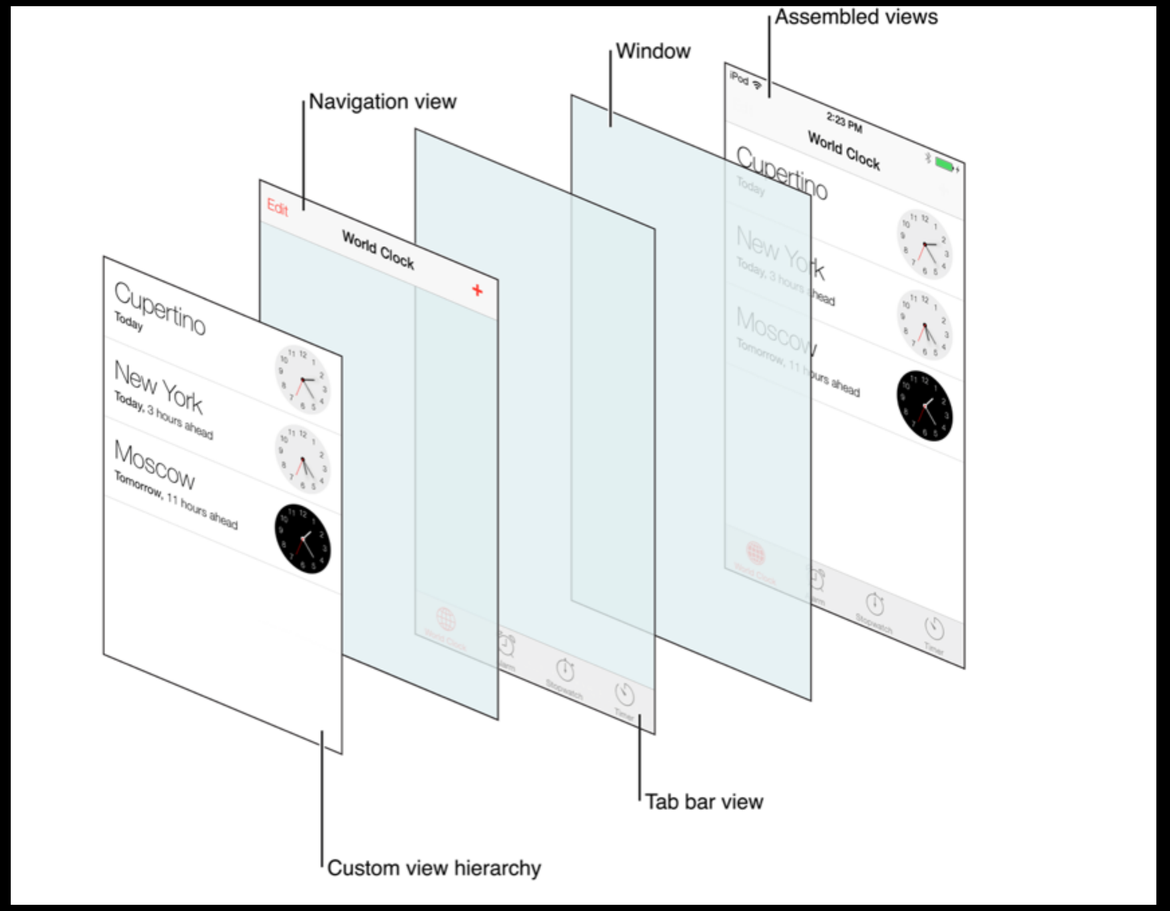
Delegation

- The whole point of delegation is to allow you to implement custom behavior without having to subclass.
- Apple could have designed UITableView's api so you would have to subclass UITableView, but then you would have to understand UITableviews in a lot more detail(which methods can I override? Do I have to call super? omfg?)
- Delegates must adopt the protocol of the object they are being delegated from.
- Delegation is used extensively in a large portion of Apple's frameworks.

TableViews

- A tableview requires 2 questions to be answered (aka methods to be implemented) and they are all in the datasource. At the very least the tableview needs data to display. It doesn't have to respond to user interaction, so the delegate object is completely optional.
- `tableView(numberOfRowsInSection:) How many rows am I going to display?`
- `tableView(cellForRowAtIndexPath:) What cell do you want for the row at this index?`
- Number of sections is actually optional, and is 1 by default.

NavigationControllers



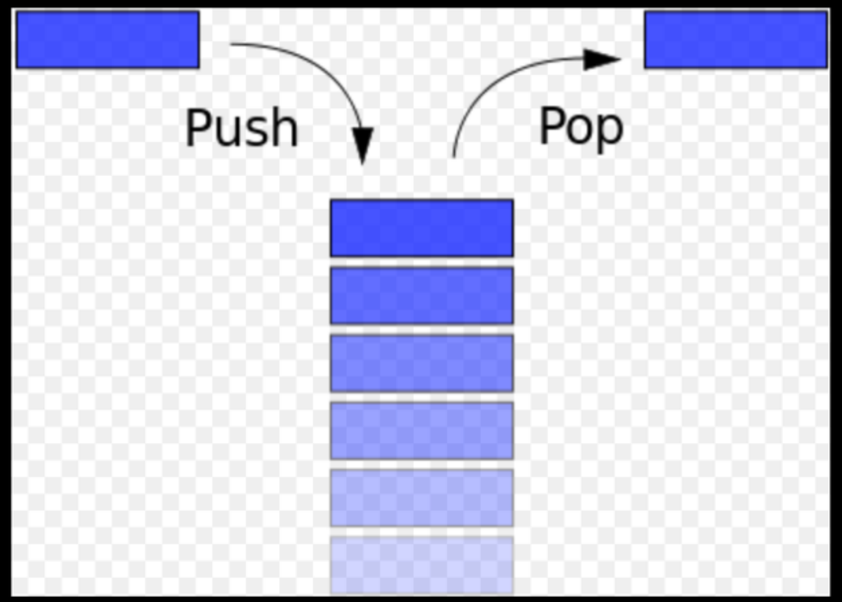
- “A navigation controller manages a stack of view controllers to provide a drill-down interface for hierarchical content.”

NavigationControllers

- “The navigation controller’s primary responsibility is to respond to user actions by pushing new content view controllers onto the stack or popping content view controllers off of the stack. Each view controller you push on the navigation stack is responsible for presenting some portion of your app’s data.”
- The first ViewController you push onto the stack becomes the rootviewcontroller and is never popped off because then no view would be on screen
- Has a property to its topViewController and and array property for all its viewControllers.

Stack Data Structure

- “A stack is particular kind of abstract data type or collection in which the only operations on the collection is adding (push) or removal (pop).” - Wikipedia
- LIFO : Last-In-First-Out. The last item added is the first to be removed.



NavigationControllers

- Storyboards make navigation controllers extremely easy to install into your app, but first its good to know how to do it programatically.
- `init(rootViewController:)` UINavigationController is initialized with a rootviewController.
- `pushViewController(animated:)` To add or ‘push’ a view controller onto the stack.
- `popViewController(animated:)` To remove or ‘pop’ a view controller from the stack.