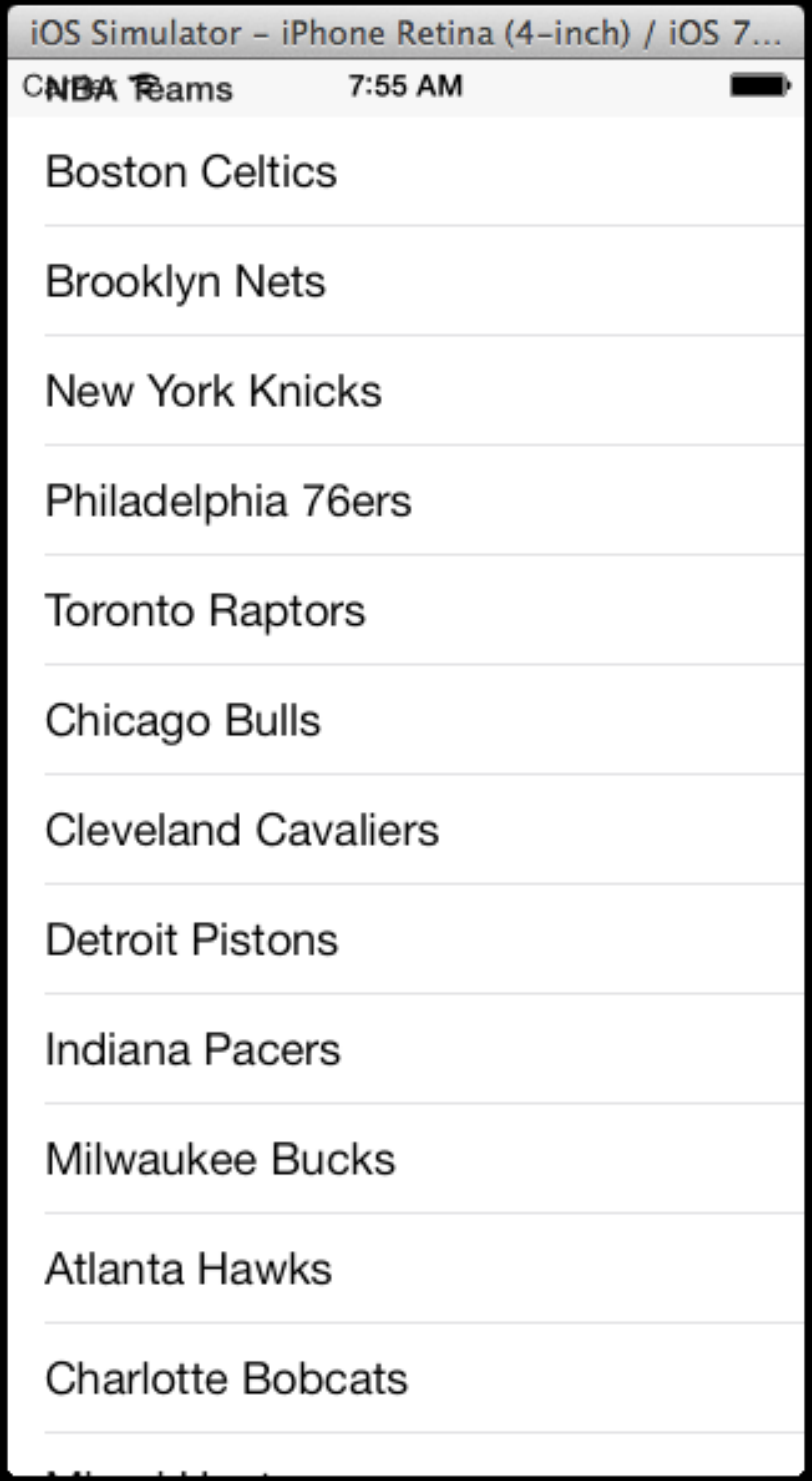
iOS Foundations II
Day 3

- Homework Review
- TableViews
- More Git Stuff

UITableView



Boston Celtics
Brooklyn Nets
New York Knicks
Philadelphia 76ers
Toronto Raptors
Chicago Bulls
Cleveland Cavaliers
Detroit Pistons
Indiana Pacers
Milwaukee Bucks
Atlanta Hawks
Charlotte Bobcats

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.

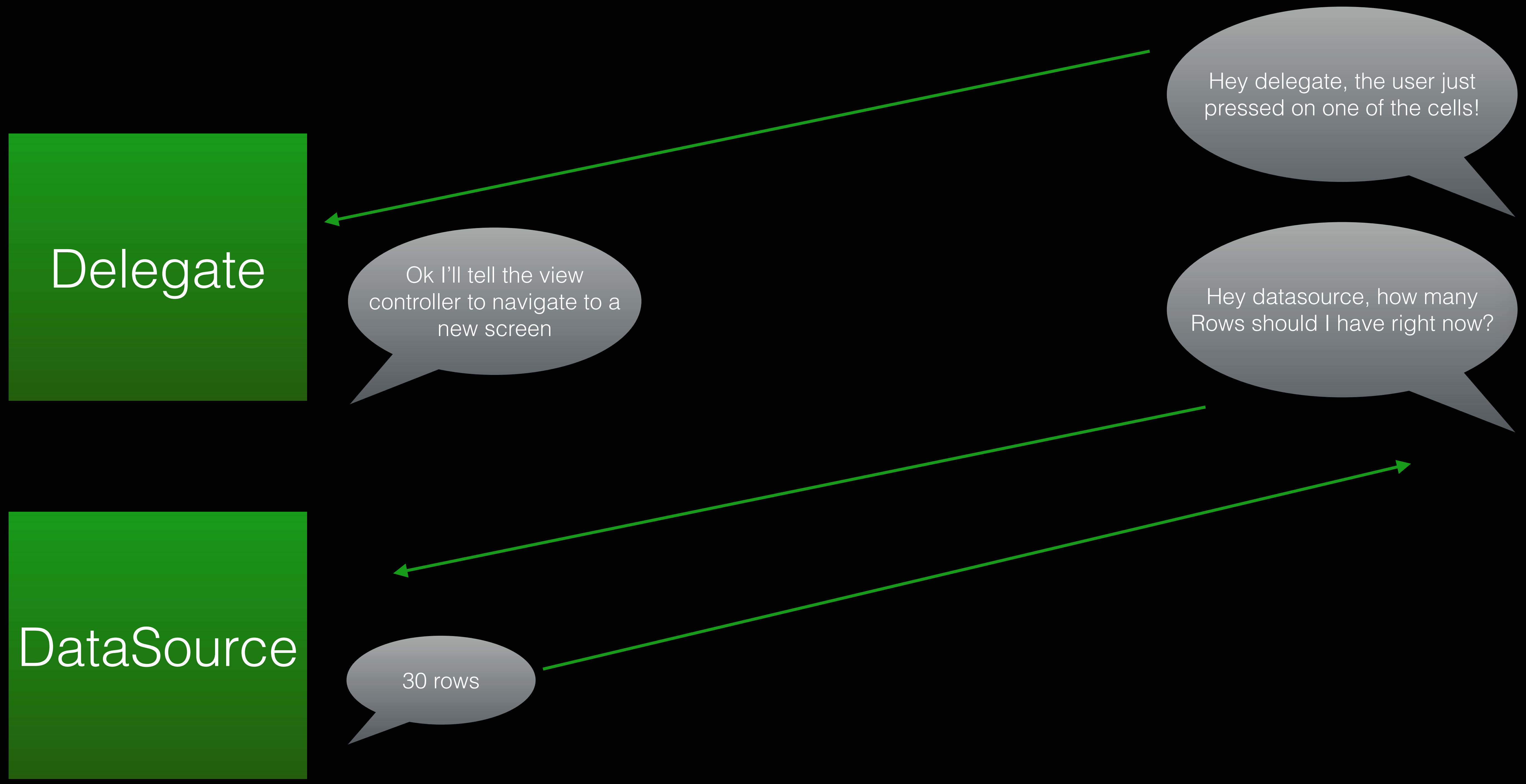
How do Tableviews work?

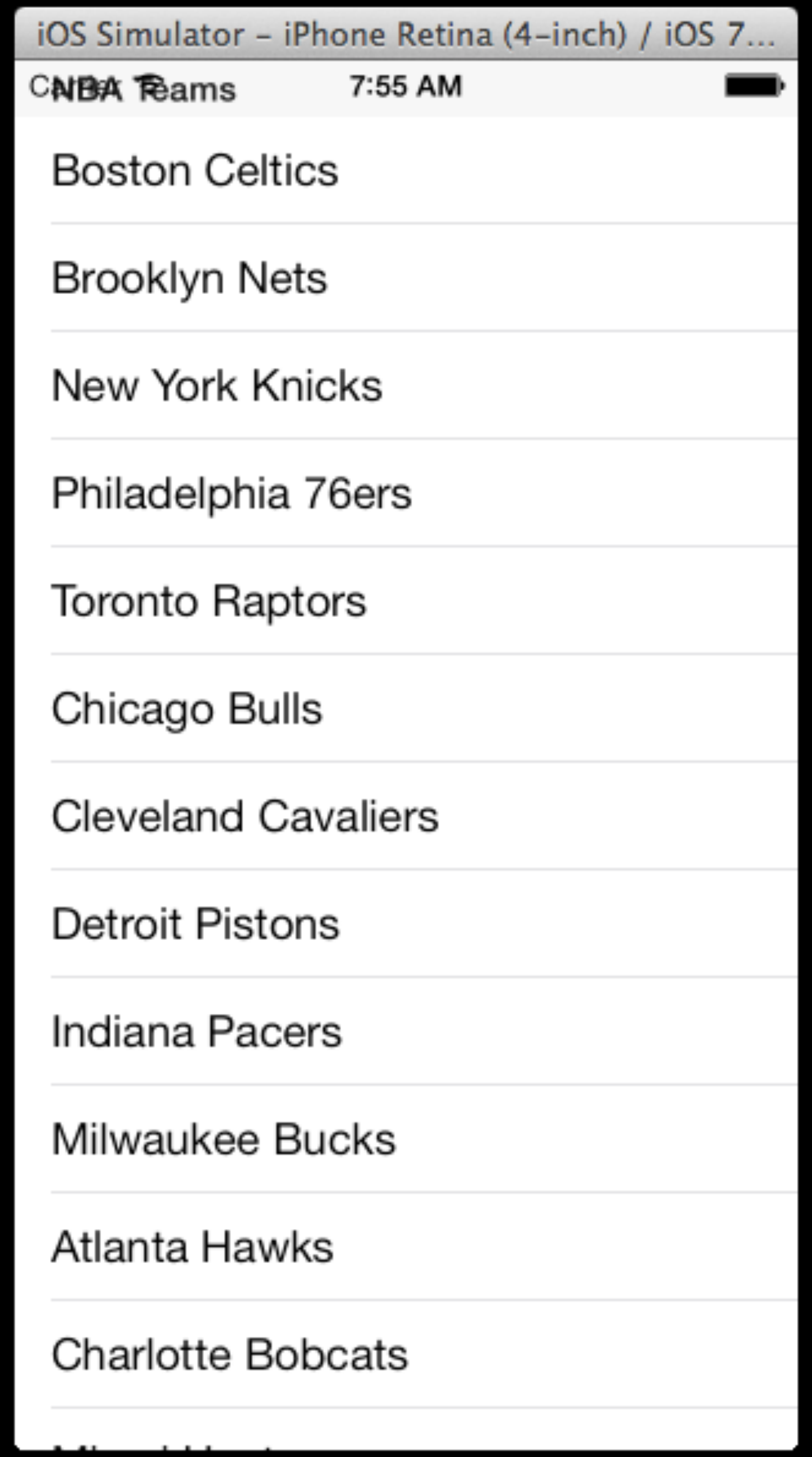
- Tableview’s rely, or **delegate**, onto other objects in order to function.
- A tableview asks another object how many rows and sections it should display, and what to display in each row.
- A tableview notifies another object when it is interacted with, like when the user selects a certain cell (aka row).
- So what is this pattern of one object relying on another called?

Delegation

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

TableViews and Delegates


Hey delegate, the user just pressed on one of the cells!
Delegate
Ok I'll tell the view controller to navigate to a new screen
Hey datasource, how many Rows should I have right now?
DataSource
30 rows

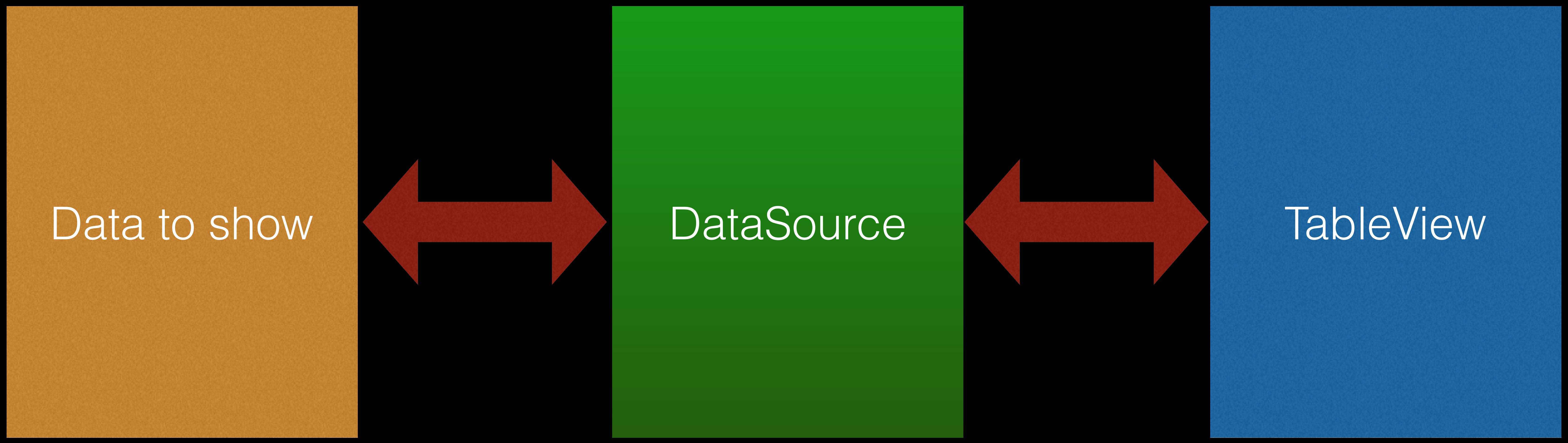

iOS Simulator – iPhone Retina (4-inch) / iOS 7...

NBA Teams 7:55 AM

Boston Celtics
Brooklyn Nets
New York Knicks
Philadelphia 76ers
Toronto Raptors
Chicago Bulls
Cleveland Cavaliers
Detroit Pistons
Indiana Pacers
Milwaukee Bucks
Atlanta Hawks
Charlotte Bobcats

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

TableViews and its data source

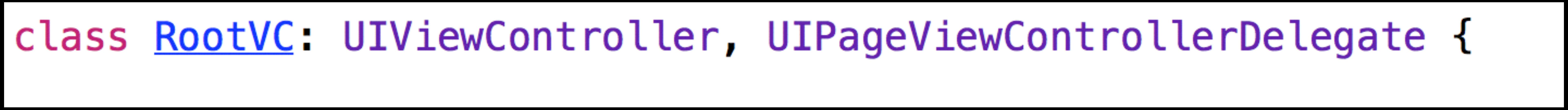


Delegation

- 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.

Protocols

- When an object wants to be something's delegate (or datasource), it needs to first conform to a protocol.
- Conforming to a protocol is like saying "hey, i can do all of these things, so let me be your delegate!"
- Sort of like signing a contract.
- To have your custom class conform to a protocol, just add the name of the protocol after the classes superclass:



```
class RootVC: UIViewController, UIPageViewControllerDelegate {
```

TableViews

- A tableview requires 2 questions to be answered (aka methods to be implemented) by their 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.

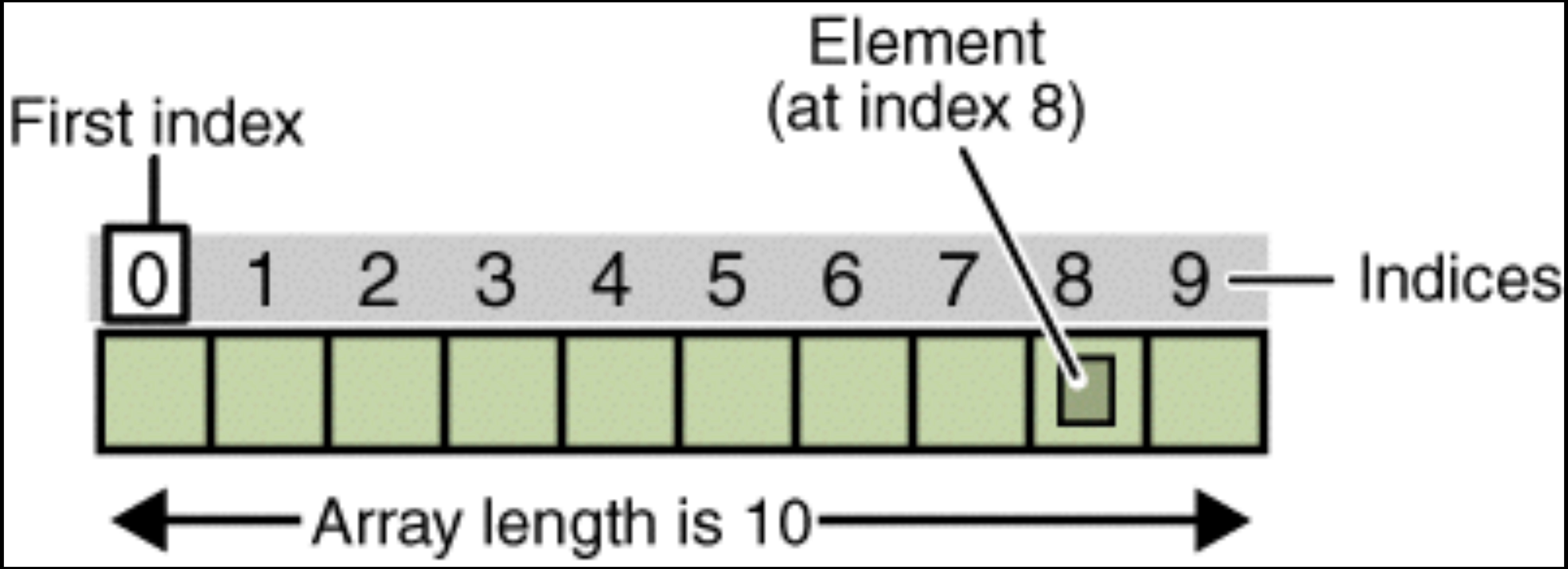
Demo

Table View and arrays

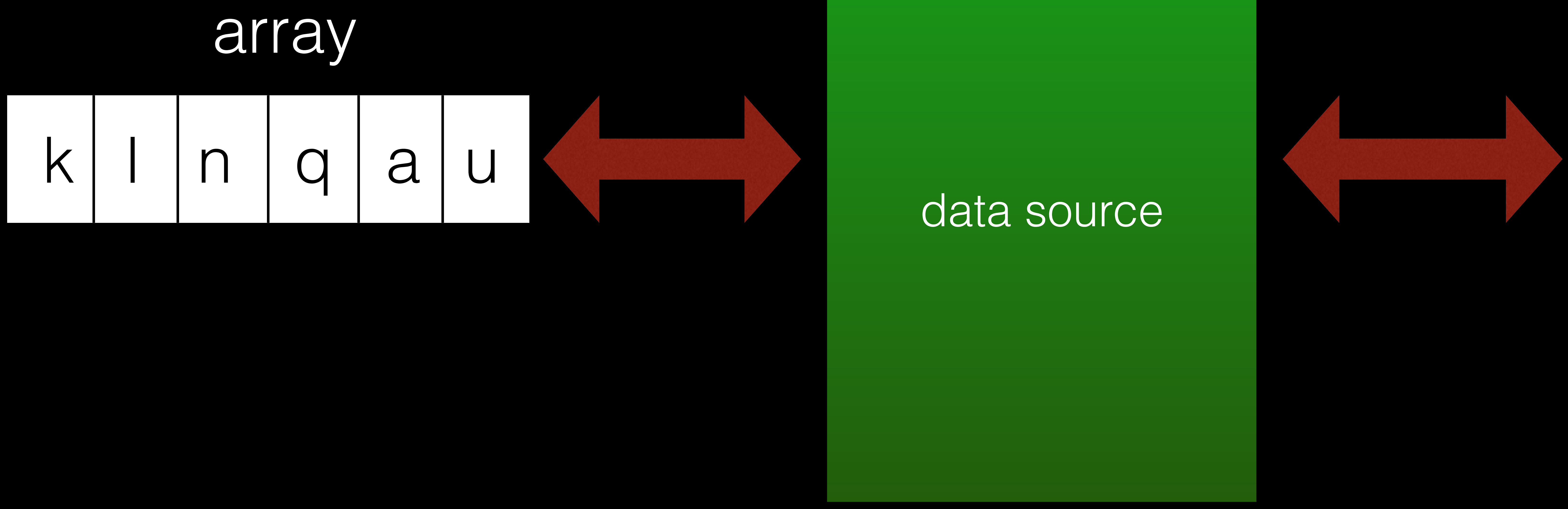
TableView+Array

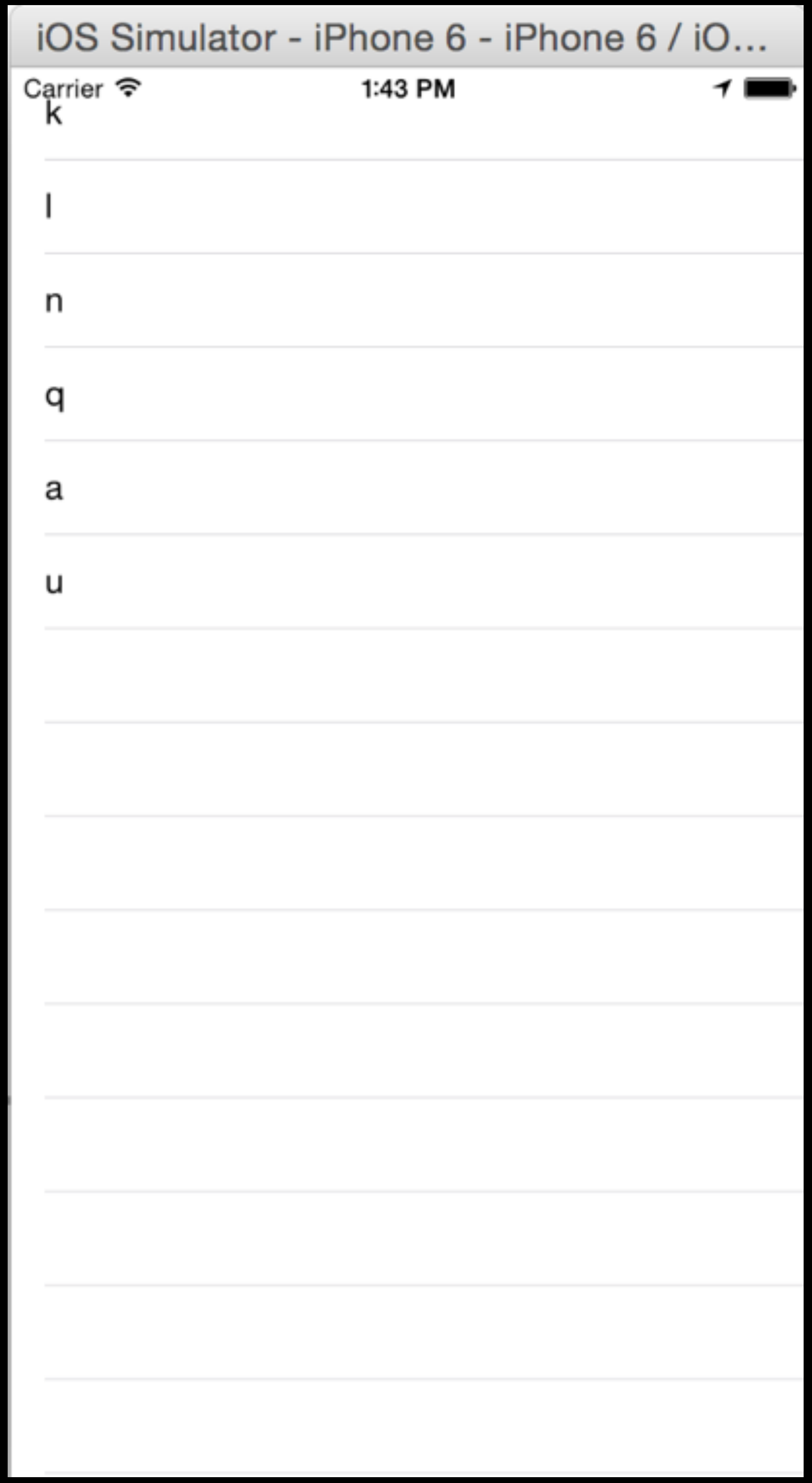
- 99% of the time, table views are ‘backed’ by an array.
- So basically the table view’s data source is the middle man between the array, and where its going to be shown, the table view
- The table view uses the count of the array to figure out how many cells its going to display
- And then for each cell, it displays whatever is at that index of the array

Array Visual



TableView+Array





k
l
n
q
a
u

Demo

Cell Reuse

Reusing Cells

- Table views are very efficient
- They can be asked to show lists of thousands, and even millions of objects
- They do this by only generating enough cells to fill the screen, and then reusing cells as they are scrolled off the screen, immediately placing them back on screen.
- So showing an array of ten thousand strings, the table view only has to generate 7 or 8 cells (enough to fill the screen, depends on screen size and row height)

Properly reusing cells

- Inside tableView:cellforRowAtIndexPath: you need to ask the tableview for a reusable cell
- You do this by calling the method dequeueReusableCellForIdentifier:indexPath on the table view
- The identifier you pass in must match the reuse identifier you set on storyboard

Demo

TableViews Gotchas

1. Did you give the tableview a data source? (by setting up an outlet to it and then settings it datasource property, or wiring it up via storyboard)
2. Does your view controller conform to the data source protocol?
3. Did you implement the required 2 methods? (how many rows, and what cell for each row)
4. Did you drag out a tableview cell, and then set its reuse identifier?
5. Does the reuse identifier in your storyboard match the one in your code?

Demo

GitHub

- Github is a repository web-based hosting service.
- Github hosts people's repositories, and those repositories can be public or private.
- The repositories on Github are just like the repositories that you have on your own machine, except Github's web application has additional features that makes working in a team much easier.

- https://help.github.com/articles/set-up-git/

Remotes

- Remote repositories are versions of your project that are hosted on the Internet or network somewhere.
- So when you have a repository on Github, it is considered a remote repository.
- When you import a directory into a git project with git init, eventually you may want to create a remote repository on Github and push to it. This gives you a backup in the cloud, and also lets your work be seen by others.
- Or you can clone an already existing remote repository (your own or someone elses) to get a local copy of the remote repository on your machine.

GitHub and Git

- Github and your local git install have 2 ways of communicating:
  - https (recommended)
  - ssh
- Both of these forms of communication require authentication.
- For https, it is recommended you use a credential helper so you don't have to enter in your credentials every time you interact with a remote repository.
- For SSH, you can generate SSH keys on your computer and then register those SSH keys to your Github account.
- Lets all follow the steps at https://help.github.com/articles/set-up-git/ together to get our github & git properly setup.

demo

Git Remote Commands

- git remote -v shows you all the remotes you have configured for your local repository on your machine
- use git remote add <nickname> <url> to add a remote repo
- after committing, use git push <nickname> <branch> to push your committed changes to your remote
- use git pull to automatically fetch and merge a remote branch into your current local branch

Demo