

# SESSION 0

# WELCOME TO ADVANCED IOS DEVELOPMENT

HOW IS THIS GOING TO WORK?

# COURSE LOGISTICS

- Flipped Classroom
  - Videos of lectures
  - Meet for discussions and office hours



# COURSE LOGISTICS

- Regular Class Meeting
  - Tuesday 5:30-6:30pm
  - Lecture Video Recap/Demos
  - Friday 10:00-10:30am (See slack poll)
  - Office hours 10:30am-11:00am
- Office hours
  - Monday 10:00 - 11:00am
  - Set up a "time" via Slack

Introduce upcoming weeks material  
Introduce/update assignment  
Open Q and A

Student showcase and presentations  
(App videos/Case Studies)



# COURSE LOGISTICS

- Regular Class Meeting
  - Tuesday 5:30-6:30pm
  - Lecture Video Recap/Demos
  - Friday 10:00-10:30am (See slack poll)
  - Office hours 10:30am-11:00am
  - Office hours
  - Monday 10:00 - 11:00am
  - Set up a "time" via Slack

Ask questions about the video (session/slide#)

Questions from Slack to be addressed

Code demos and walkthroughs

# COURSE LOGISTICS

- Regular Class Meeting
  - Tuesday 5:30-6:30pm
  - Lecture Video Recap/Demos
  - Friday 10:00-10:30am (Slack poll)
  - Office hours 10:30am-11:00am
- Office hours
  - Monday 10:00 - 11:00am (Slack poll)
  - Set up a "time" via Slack

**Group vs. individual vs. working time in a "breakout" room**

# COURSE RESOURCES

# COURSE RESOURCES

- Canvas "Lite"
- Links to materials
  - Github
  - Panopto

The sidebar menu includes:

- Account
- Dashboard
- Courses** (highlighted)
- Calendar
- Inbox
- Commons
- Help

Panopto Video (highlighted with a red box)

Links to external resources:

- Purchase UChicago Bookstore Course Materials
- Purchase Seminary Co-op Course Materials
- Zoom - University of Chicago Main Account
- Collaborations
- Conferences

2020.02

## Course Syllabus

[Jump to Today](#)

[Edit](#)

[Home](#)

[Announcements](#)

See full syllabus on Github

[Syllabus](#)

<https://github.com/uchicago-mobi/mpcs51032-2020-spring/wiki/Syllabus>

[Modules](#)

[Assignments](#)

[Discussions](#)

[Library Reserves](#)

[People](#)

[Grades](#)

[Panopto Video](#)

[Purchase UChicago Bookstore Course Materials](#)

[Purchase Seminary Co-op Course Materials](#)

[Zoom - University of Chicago Main Account](#)

[Collaborations](#)

[Conferences](#)

## Course Summary:

Date	Details	
Tue Apr 21, 2020	<a href="#">Assignment 1</a>	due by 5:29pm

University focus on  
protecting student work  
and information

# COURSE RESOURCES

- Panopto video hosting
- Organized by "Sessions"
- Link directly from within canvas



THE UNIVERSITY OF CHICAGO

Powered by Panopto

Search in folder "session1"...

Create

Home My Folder Shared with Me Everything

session1

↑ Refresh Filter by date  Show scheduled recordings

Sort by: Name Duration Date ▾

Add folder

session-1a

Processing : 72%

session-1b

6 hours ago

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • 25:42

# COURSE RESOURCES

- Github  
Repository for all course materials (no website 😢)
- <https://github.com/uchicago-mobi/mpcs51032-2020-spring>

The screenshot shows a GitHub repository page for the repository `uchicago-mobi / mpcs51032-2020-spring`. The repository has 2 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. The latest commit was made 1 hour ago. A yellow callout box highlights the repository name and describes it as containing "Materials for each session (slides, videos, projects, etc.)".

uchicago-mobi / mpcs51032-2020-spring

No description, website, or topics provided.

Manage topics

2 commits 1 branch 0 packages 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

tabinks Add session 1 materials Latest commit 32e2c12 1 hour ago

session1 Add session 1 materials 1 hour ago

README.md Initial commit 3 hours ago

MPCS51032-2020-spring

Materials for each session (slides, videos, projects, etc.)

Contact GitHub Pricing API Training Blog About

# COURSE RESOURCES

A screenshot of the GitHub header bar. It features a search bar with placeholder text "Search or jump to...", navigation links for "Pull requests", "Issues", "Marketplace", and "Explore", and a notification bell icon with a blue dot indicating four notifications. On the far right is a user profile picture.

A screenshot of a GitHub repository page. The repository name is "uchicago-mobi / mpcs51032-2020-spring". Below the name are buttons for "Unwatch" (with 4 notifications), "Star" (0), and "Fork" (0). A horizontal navigation bar below the repository name includes links for "Code", "Issues 0", "Pull requests 0", "Actions", "Projects 0", "Wiki" (which is highlighted with an orange underline), "Security", "Insights", and "Settings".

A screenshot of an assignment page titled "Assignment 1". The page content asks: "What do John Travolta, Barack Obama, Shaquille O'Neil, Whoopi Goldberg, Glenn Beck, Jay Leno, Cheech Marin, Jerry Seinfeld, Sting, Queen Latifah, Jimmy Buffett, Madonna, Spike Lee, Jerry Garcia, Paul McCartney, Jamie Lee Curtis, Bob Dylan, The Dutches of York and Steve Martin have in common?". Below the question, it says: "They've all written a children's book...and so will you." To the right of the question are two buttons: "Edit" and "New Page".

tabinks edited this page 2 hours ago · 5 revisions

What do John Travolta, Barack Obama, Shaquille O'Neil, Whoopi Goldberg, Glenn Beck, Jay Leno, Cheech Marin, Jerry Seinfeld, Sting, Queen Latifah, Jimmy Buffett, Madonna, Spike Lee, Jerry Garcia, Paul McCartney, Jamie Lee Curtis, Bob Dylan, The Dutches of York and Steve Martin have in common?

They've all written a children's book...and so will you.

A screenshot of a sidebar for a wiki page. It includes a search bar labeled "Find a Page...", a "Home" link, and a "Assignment 1" link. A large yellow arrow points from the left towards the "Assignment 1" link. At the top of the sidebar, there is a section labeled "Pages 4" with a dropdown arrow.

# COURSE RESOURCES

Post all questions to the appropriate channel unless personal. **All HW questions need to be in #assignment channels.**

The screenshot shows a Slack workspace interface. On the left, there's a sidebar with various icons and a list of channels. The '# general' channel is currently selected and highlighted in blue. Other visible channels include '# assignment-1', '# office-hours', '# random', and Direct messages. A message from user 'abinkowski' in the '# general' channel is highlighted, reading: "Your first assignment Please respond to this thread with what you are most interested in learning about related to iOS, watchOS, tvOS, iPadOS, macOS development this quarter. I have a long list of topics (which we will discuss tomorrow), but I want to make sure that I cover what you are most interested you're not sure, just wait until tomorrow after class to answer." A yellow callout box with a black border and white text contains the instruction: "Post all questions to the appropriate channel unless personal. **All HW questions need to be in #assignment channels.**". The main workspace shows other messages and activity in the '# general' channel, such as a welcome message from 'abinkowski' and a message from 'Tulip Fan'.

# ZOOM

- Please make sure you are muted if you are not talking
- Questions
  - "Raise Hand" if you have a question
  - Type it in the chat
  - Post in slack
- All sessions will be recorded
  - You can opt-out by turning camera and microphone off
  - Will not be distributed outside of class



# CLASS

# COURSE SESSIONS

- Small class so we have a lot of flexibility
- Please let me know how things are going
- Having any difficulties or concerns about anything, please let me know



# ADVANCED IOS

# ADVANCED IOS DEVELOPMENT

- Everyone here can build an app
  - Building an app is easy; building a successful one is difficult
- Purpose of class:
  - Develop real-world skills on more complex projects
  - Additional platforms (watchOS, tvOS)
  - Create a portfolio of apps
  - Develop a "polished" final project



# ADVANCED IOS DEVELOPMENT

- Where can it take you
  - Professional iOS development
  - Indie developer
  - TA for this class :)



# COURSE DESIGN

# COURSE DESIGN

- Thorough instruction on the more “advanced” concepts for developing iOS apps
  - Some are just new frameworks we have not covered
    - CloudKit, TextKit, SpriteKit, CoreML, CoreImage
  - Some are “advanced”
    - Performance and optimization, CoreData, Custom Framework, Extensions

# COURSE DESIGN

- New platforms
  - tvOS
  - watchOS
  - iPadOS
  - macOS Catalina



# COURSE DESIGN



- We are going to learn Objective-C 😳

# COURSE DESIGN

- Expectations
  - You are highly motivated
  - You will do a lot of self-learning to address questions that effect your app
  - You will not wait until the last minute on assignments and projects
  - You will show up for "class" and participate

# ASSIGNMENTS

- Syllabus
- <https://github.com/uchicago-mobi/mpcs51032-2020-spring/wiki/Syllabus>

## Syllabus

tabinks edited this page now · 3 revisions

Course syllabus is subject to change.

### Session 1

- Page view controllers
- Speech synthesis
- External audio sources
- Attributed String
- Text Kit
- Nib view controllers

### Session 2

- Objective-C
- Finger painting
- UIKit Dynamics
- App Store rules for Kids applications
- Advanced UIView animations
- Property animators

### Session 3

- CloudKit
- Notifications

#### ▼ Pages 4

Find a Page...

[Home](#)

[Assignment 1](#)

[Session 1](#)

[Syllabus](#)

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/uchicago-mobi/mpcs51032-2020-spring/wiki/Syllabus>



# COURSE DESIGN

- Polled former students and colleagues about the topics that are most beneficial
- Feel free to suggest topics (Slack, email, now)

The screenshot shows the 'Jobs at Apple' website interface. At the top, there are links for Mac, iPad, iPhone, Watch, TV, Music, and Support. Below that, there are tabs for Corporate, Retail, and Students. The main title is 'Jobs at Apple' and it says '591 job(s) found'. On the left, there's a 'Filter by' sidebar with options like Keywords (selected), Location, Language Skills, Business Line, Job Function, and My Filter Mixes. Below the sidebar are 'Clear all filters' and 'Hide filters' buttons. The search bar contains the word 'ios'. A note in a box says: 'Certain filters have already been applied. To change the results, change the filters. Corporate jobs require English-language skills.' The main content area shows a table of 10 job listings, each with columns for Job Title, Job Function, and Location. All listed jobs are in Santa Clara Valley and are Software Engineering roles related to iOS.

Job Title	Job Function	Location
iOS QA Engineer	Software Engineering	Santa Clara Valley
Siri iOS Engineer	Software Engineering	Santa Clara Valley
Siri iOS Engineer	Software Engineering	Santa Clara Valley
Siri iOS Engineer	Software Engineering	Santa Clara Valley
Siri iOS Engineer	Software Engineering	Santa Clara Valley
iOS App Reviewer	Marketing	Santa Clara Valley
iOS Power QA Engineer	Software Engineering	Santa Clara Valley
iOS Home App Engineer	Software Engineering	Santa Clara Valley

# COURSE DESIGN

- 4 “App Store ready” apps as homework assignments
- Third-party Code Case Study
- Create your own app for final project

# COURSE DESIGN

- Week 1 - Homework 1 Assigned
- Week 2 -
- Week 3 -Homework 1 Due, Homework 2 Assigned
- Week 4 -
- Week 5 -Homework 2 Due, Homework 3 Assigned
- Week 6 -
- Week 7 - Homework 3 Due, Homework 4 Assigned
- Week 8 -
- Week 9 - Homework 4 Due
- Final Exam Week - Final Projects Due



**USE OFF WEEKS TO WORK ON FINAL  
PROJECT AND CASE STUDIES**

# COURSE DESIGN

- Assignments (4 \* 16.25%) 65%
- Case Study %5
- Final Project 30%

# ASSIGNMENTS

- Honor Code
  - All the assignments should be your own work
  - Department policies are strictly enforced
- Online Resources
  - Most valuable resources you have for learning to develop iOS apps
  - Attribution is required in README.md file AND retain original headers

# **ASSIGNMENTS**

# ASSIGNMENTS

- Focus on application design (MVC)
- Best practices in coding style
- User Interface design and user experience
- Use as many frameworks as possible

# ASSIGNMENTS

User interfaces  
matters

- Homework grading
  - Specific requirements to be met
  - Must compile with no warning/errors
  - Commented using HeaderDoc formatting style
  - **Included requirements**
  - Style Points
    - Application design
    - Coding factoring/style
    - Readability
    - Best practices

# ASSIGNMENTS

- Instructor Code Review
  - You have 2 weeks to do each assignment
  - Treat off week as 'code review'
    - Suggestions about coding style, refactoring
    - Help identify common issues to address in class
- Suggestion
  - Week 1: Application design, prototype basic functionality
  - Week 2: Add new functions, refinements, interface design



Feature rollout is  
purposeful

# ASSIGNMENTS

- Third Party Frameworks
  - You may **not** use 3rd party frameworks for homework unless noted or specific permission is granted
  - You may use them for your final project.
    - Please submit them with your proposal.
    - You should be able to comfortably explain what they do.
  - Some third-party frameworks are overly complex for what you are trying to do. Some (even popular ones) do not use best practices.

# CASE STUDIES

# CONTROL CASE STUDIES

- Profile an open source control or framework
  - Cocoa Controls
  - Github
  - Maniac Dev
- Aims
  - Identify best practices
  - Increase your exposure (both senses of the word)
  - See under-the-hood of design patterns
  - Code review

The screenshot shows the Cocoa Controls website interface. On the left, there's a sidebar with filter options: 'SORT' (Date selected), 'FILTER COCOAPODS' (Yes selected), and 'LICENSES' (All selected). The main content area displays two iPhone screenshots. The top screenshot shows a list of controls with a 'Date' filter applied. The bottom screenshot shows a different view with a 'Carrier' status bar.

**SORT**

Date

Rating

Apps

**FILTER COCOAPODS**

Yes

No

**LICENSES**

All

Apache 2.0

BSD

CC BY 3.0

CC BY-SA 3.0

Commercial

Custom

Eclipse Public License

GPL

ISC

MIT

In Xcode instead of running the project, choose from the menu Product/Test to run the unit tests included in the xcode project.

# CONTROL STUDIES

- 5% of final grade
- Requirements
  - ~15 minutes technical presentation and questions
  - Code review of control
  - Implement the control in a demo app that you will walk-through
  - Source code will be put in class playground to share
  - Submit a pull-request to the author
    - Just had one closed by Apple from 2 years ago on ResearchKit

Good interview prep

The screenshot shows a GitHub repository page for 'MHTabBarController' by 'hollance'. The repository has 332 stars and 5 issues. It contains code for a custom tab bar controller for iOS 5. The README.md file describes it as a custom container view controller for iOS 5 that works like a regular UITabBarController but with tabs at the top. It includes a screenshot of an iPhone displaying three tabs labeled 'Tab 1', 'Tab 2', and 'Tab 3', each showing a list of items labeled 'Tab 1 - Row 0', 'Tab 1 - Row 1', 'Tab 1 - Row 2', and 'Tab 1 - Row 3'. A note at the bottom states that customizing the tab bar's appearance requires modifying the code.

fantastical definition · [UIcollectionView Subclass](#) · [ViewController Catalog](#) · [adownload.apple.com](#) · [Twitter+client.png \(1024x1024\)](#)

**github** Explore GitHub Search Features Blog Sign up for free

**hollance / MHTabBarController**

PUBLIC [Code](#) [Network](#) [Pull Requests](#) [Issues](#) [Graphs](#)

A custom tab bar controller for iOS 5 — Read more <http://www.hollance.com/2011/11/mhtabbarcontroller-a-custom-tab-bar-for-ios-5-using-the-new-container-apis/>

[Clone in Mac](#) [ZIP](#) [HTTP](#) [SSH](#) [Git Read-Only](#) <https://github.com/hollance/MHTabBarController.git> [Read-Only](#)

[branch: master](#) [Files](#) [Commits](#) [Branches](#)

**MHTabBarController**

Uses UITabBarItem to set the tab's title and an optional image.

**hollance** authored 3 months ago latest commit 45a56d7

File	Time Ago	Description
Demo.xcodeproj	3 months ago	Uses UITabBarItem to set the tab's title and an optional image. [hollance]
Demo	3 months ago	Uses UITabBarItem to set the tab's title and an optional image. [hollance]
MHTabBarController	3 months ago	Uses UITabBarItem to set the tab's title and an optional image. [hollance]
.gitignore	a year ago	First commit. [hollance]
Design.psd	3 months ago	Uses UITabBarItem to set the tab's title and an optional image. [hollance]
README.md	3 months ago	Cleaned up and modernized the code a bit. Added @2x graphics. [hollance]
Screenshot.png	a year ago	First commit. [hollance]

**README.md**

**MHTabBarController**

This is a custom container view controller for iOS 5 that works just like a regular UITabBarController, except the tabs are at the top and look different.

To customize the tab bar's appearance you currently have to mess around in the code a bit.

The MHTabBarController source code is copyright 2011-2012 Matthijs Hollmans and is licensed under the terms of the MIT license.

# CONTROL STUDIES

- Signups will be sent out later
- Find something to use in your final project

github Explore GitHub Search Features Blog

Sign up for free Sign in

PUBLIC hollance / MHTabBarController

Code Network Pull Requests 1 Issues 5 Graphs

A custom tab bar controller for iOS 5 — Read more  
<http://www.hollance.com/2011/11/mhtabbarcontroller-a-custom-tab-bar-for-ios-5-using-the-new-container-apis/>

Clone in Mac ZIP HTTP SSH Git Read-Only https://github.com/hollance/MHTabBarController.git Read-Only access

branch: master Files Commits Branches 1 Tags 1

MHTabBarController + 4 commits

Uses UITabBarItem to set the tab's title and an optional image.

hollance authored 3 months ago latest commit 45a55d7db8

File	Time	Description
Demo.xcodeproj	3 months ago	Uses UITabBarItem to set the tab's title and an optional image. [hollance]
Demo	3 months ago	Uses UITabBarItem to set the tab's title and an optional image. [hollance]
MHTabBarController	3 months ago	Uses UITabBarItem to set the tab's title and an optional image. [hollance]
.gitignore	a year ago	First commit. [hollance]
Design.psd	3 months ago	Uses UITabBarItem to set the tab's title and an optional image. [hollance]
README.md	3 months ago	Cleaned up and modernized the code a bit. Added @2x graphics. [hollance]
Screenshot.png	a year ago	First commit. [hollance]

README.md

**MHTabBarController**

This is a custom container view controller for iOS 5 that works just like a regular UITabBarController, except the tabs are at the top and look different.



To customize the tab bar's appearance you currently have to mess around in the code a bit.

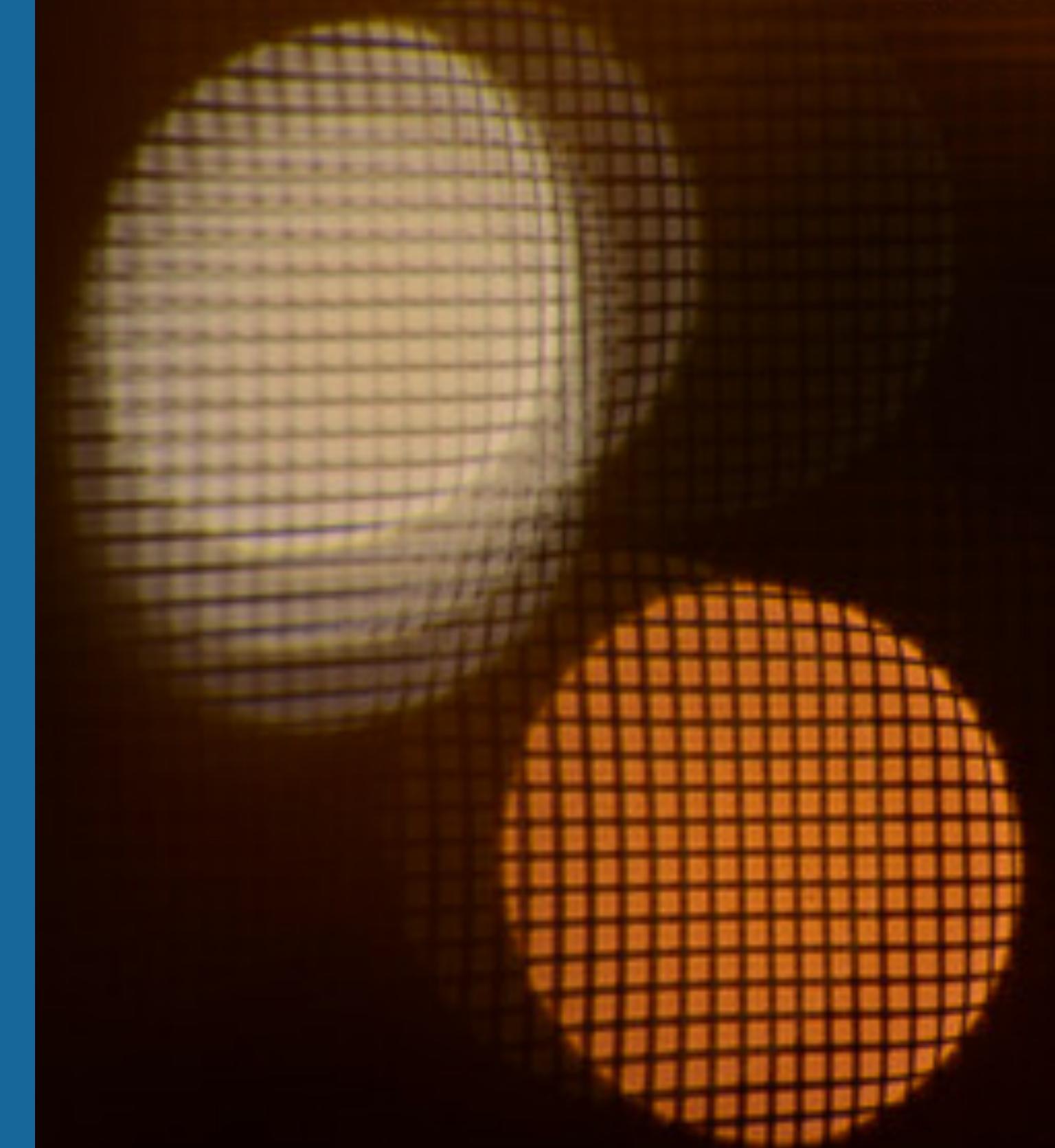
The MHTabBarController source code is copyright 2011-2012 Mathijs Hollmans and is licensed under the terms of the MIT license.

# **FINAL PROJECTS**

# FINAL PROJECTS

- Create new application
- Continue your app with major advancements
- Submit proposals anytime
- Start discussing it now

Sound On Sou  
by Diomed



be creative, again.

# ADVANCED IOS

- Questions??



# ASSIGNMENT 1

# POP QUIZ

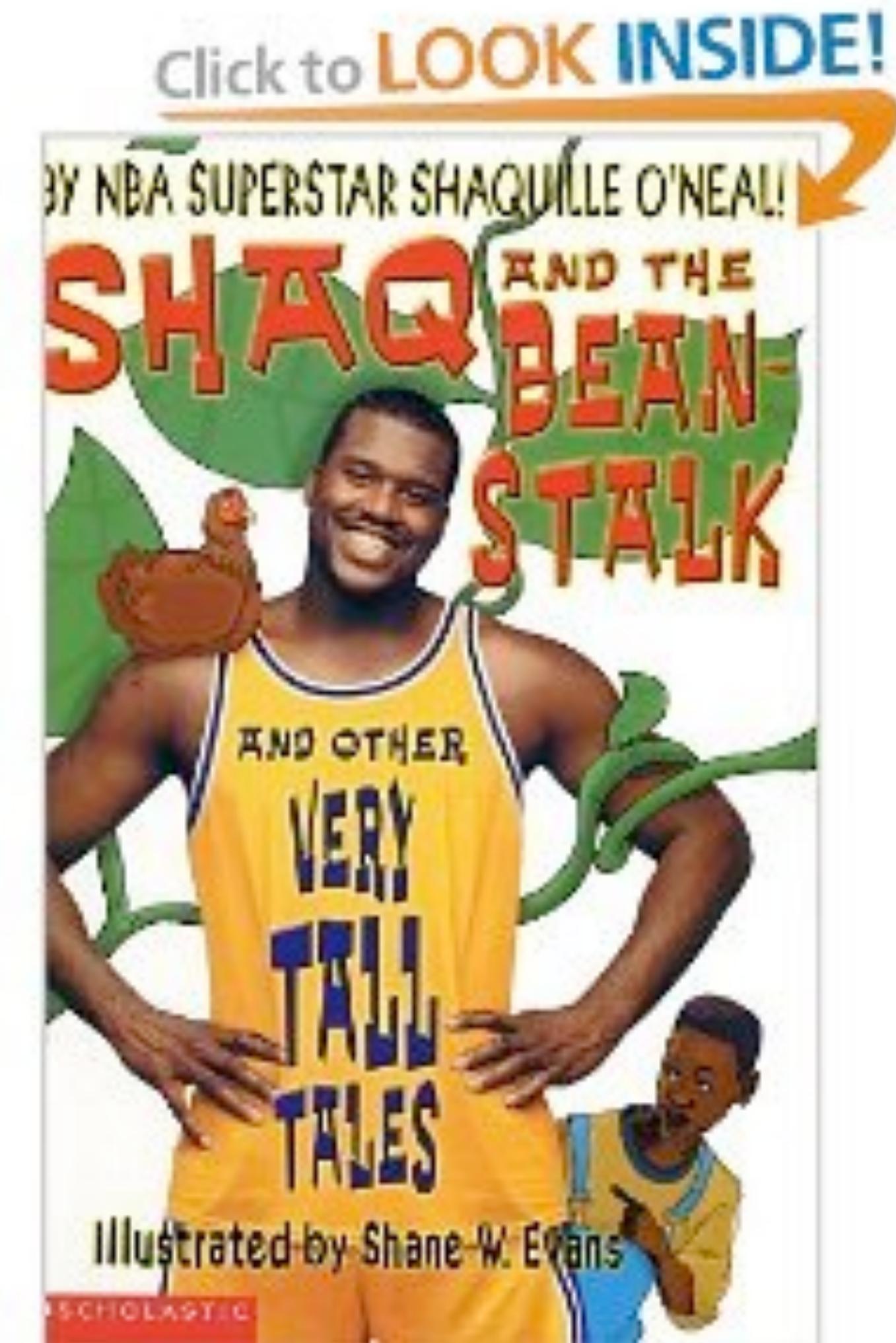
What do John Travolta, Barack Obama, Shaquille O'Neil,  
Whoopi Goldberg, Glenn Beck, Jay Leno, Cheech Marin,  
Jerry Seinfeld, Sting, Queen Latifah, Jimmy Buffett,  
Madonna, Spike Lee, Jerry Garcia, Paul McCartney, Jamie  
Lee Curtis, Bob Dylan, The Dutchess of York and Steve  
Martin have in common?

# ASSIGNMENT 1



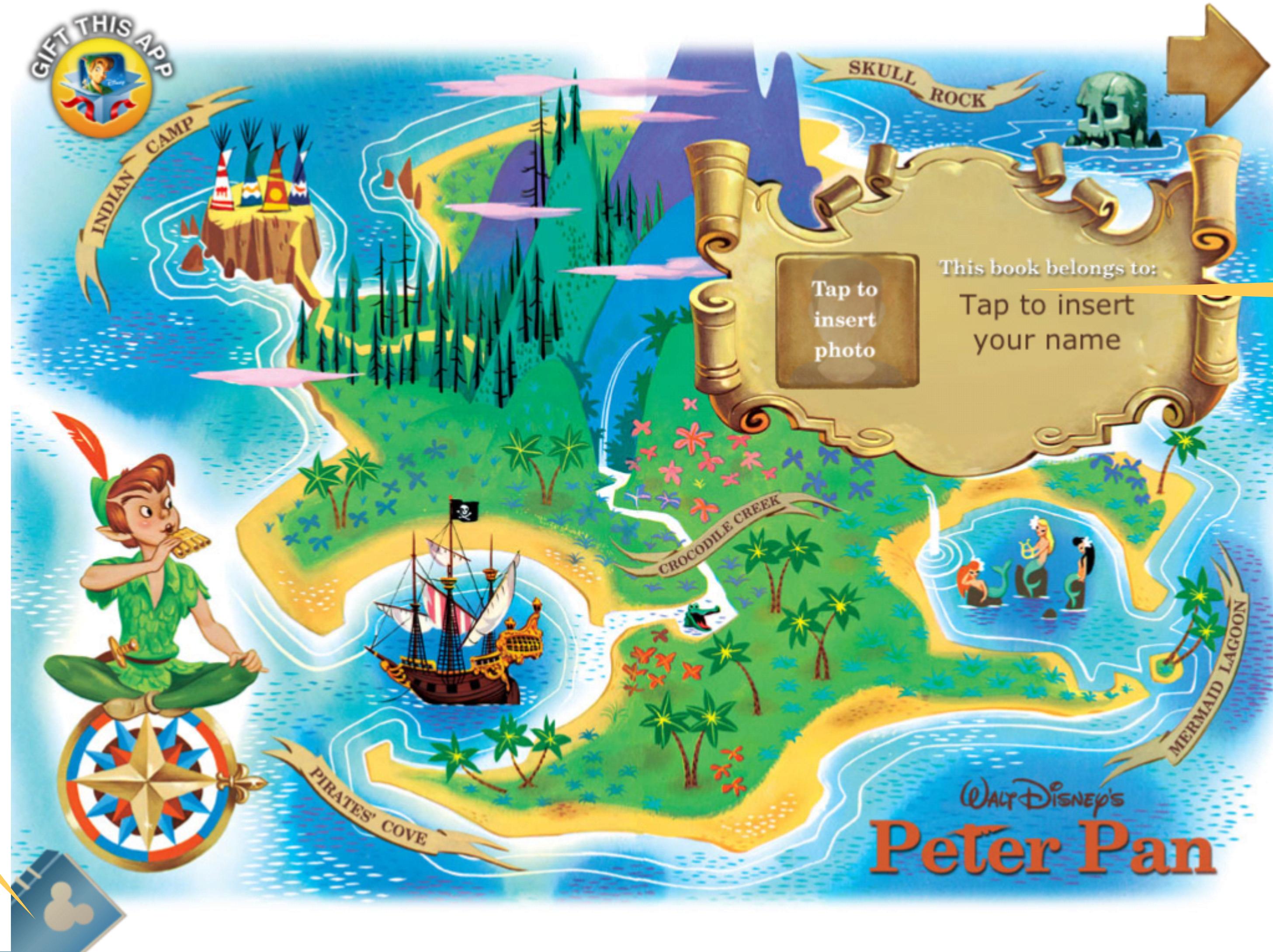
# IF A CELEBRITY CAN DO IT, SO CAN YOU

- Create a children's book using `UIPageViewController
- A hilarious, heartwarming and/or educational tale
  - At least 6 pages of unique content
- Must be better than "Shaq and Beanstalk"



# ASSIGNMENT 1

About the author

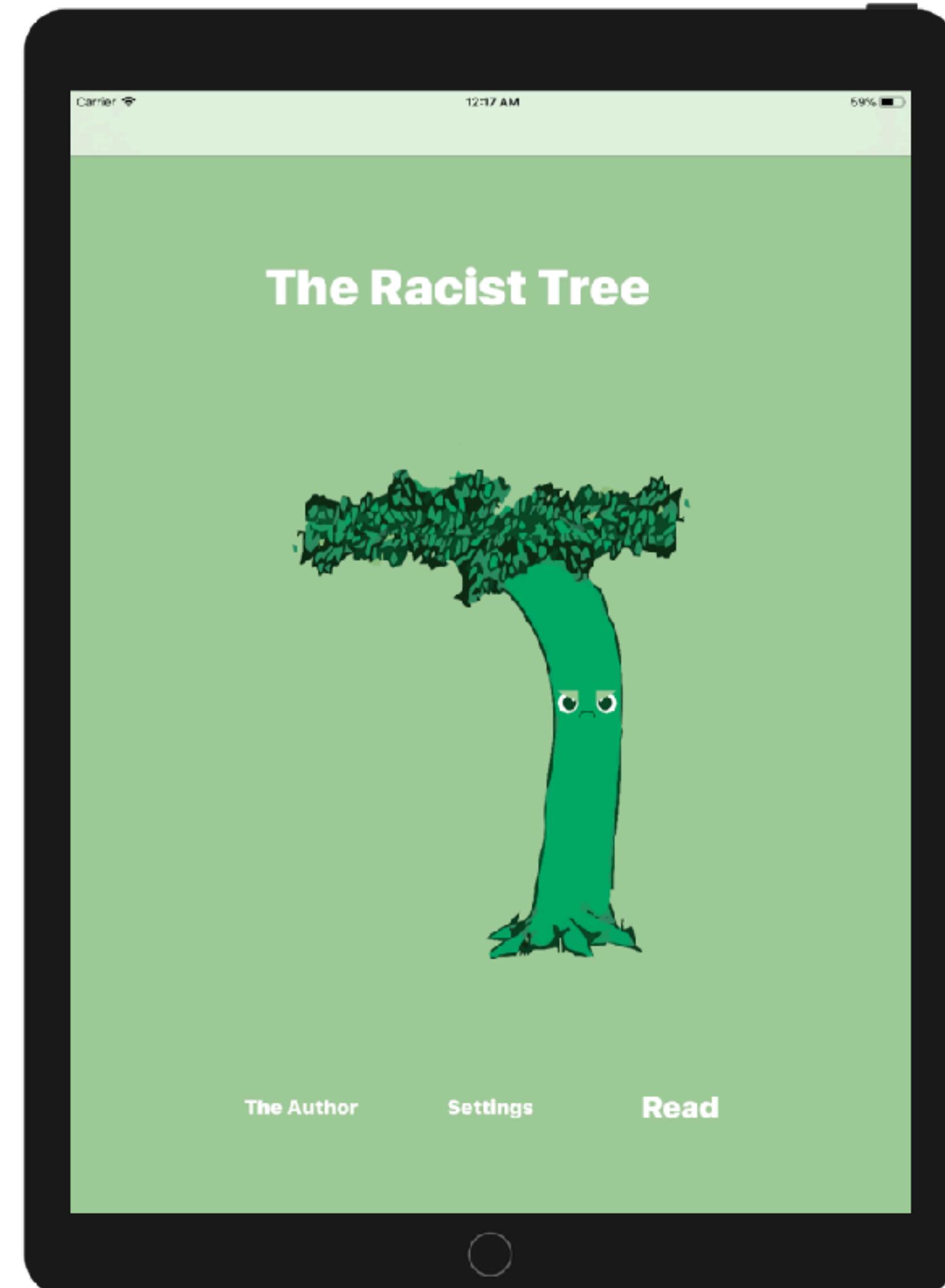
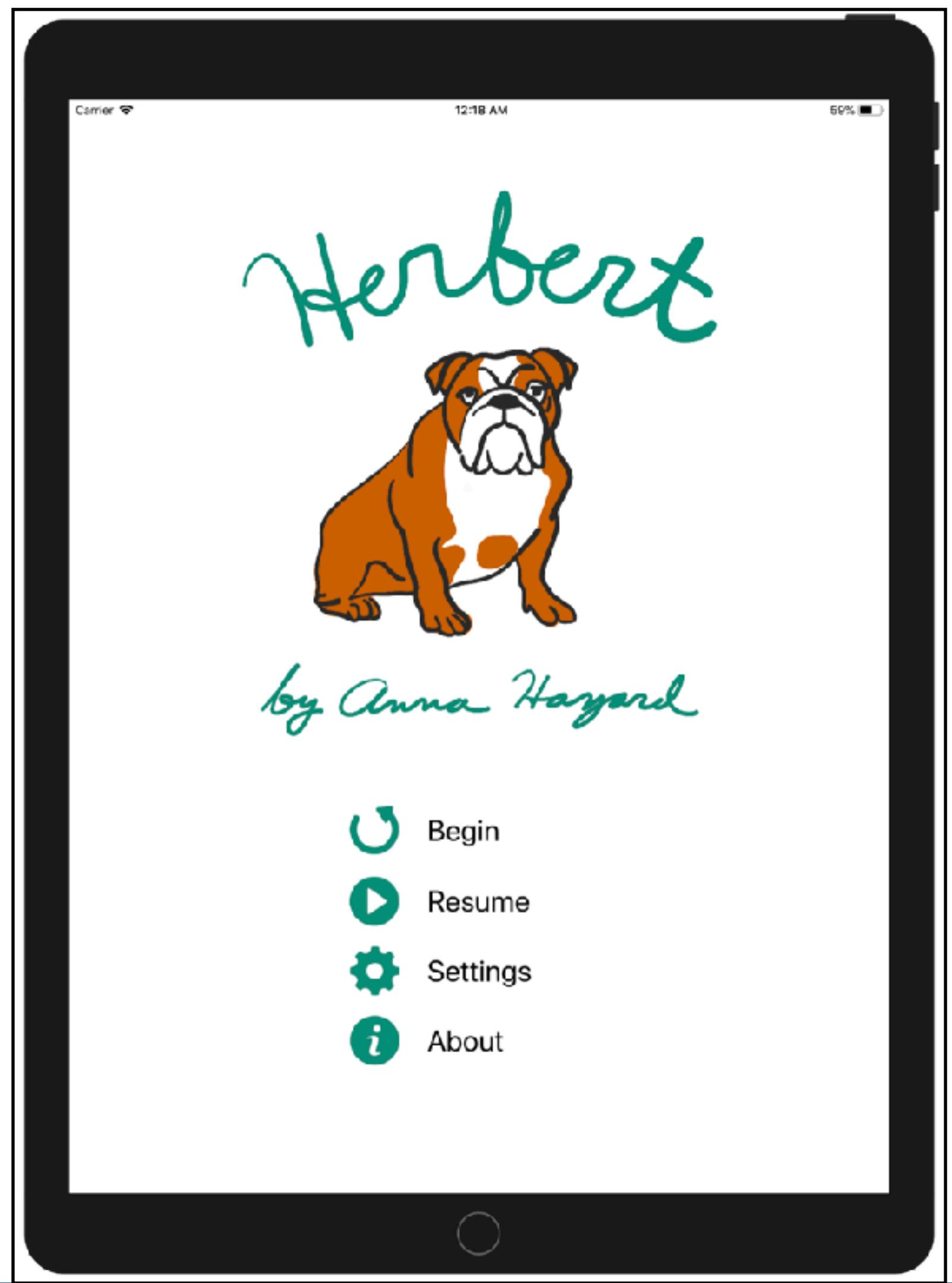


Settings

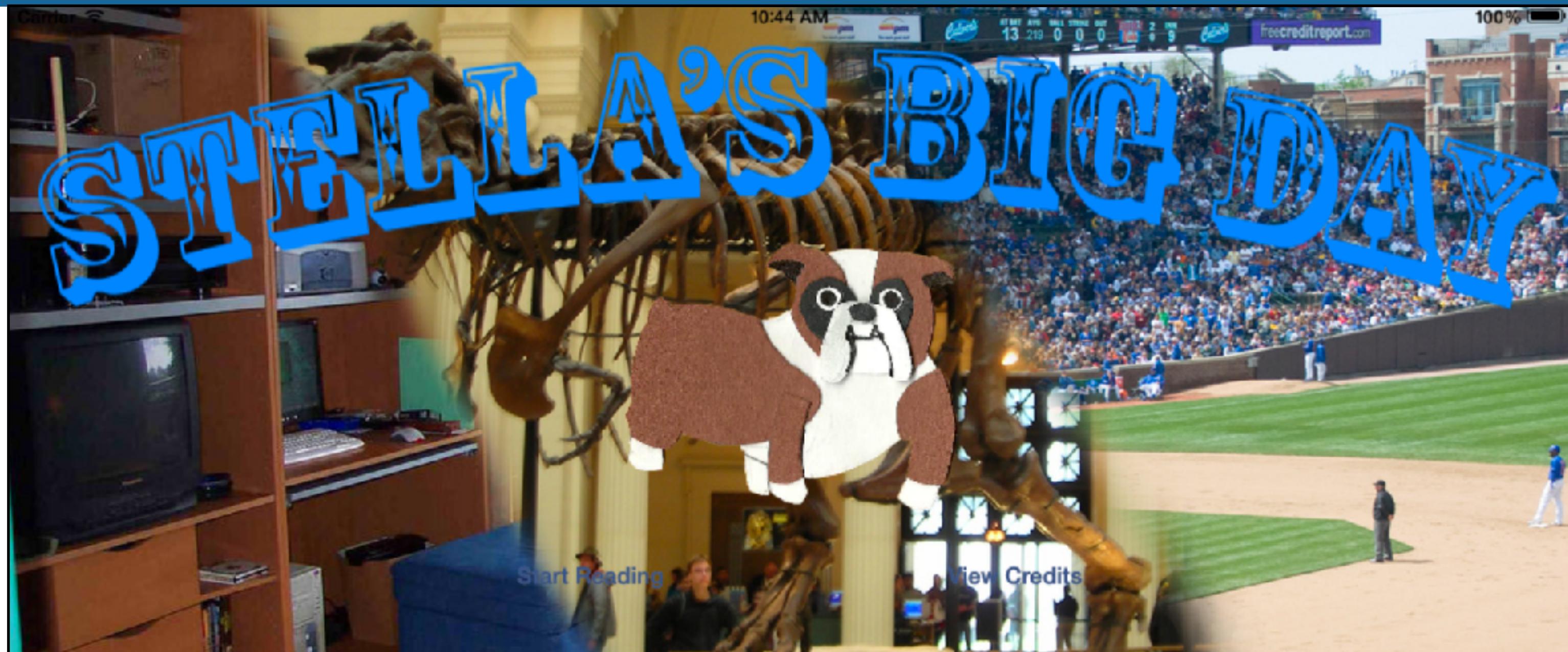
# ASSIGNMENT 1



# ASSIGNMENT 1



# ASSIGNMENT 1



Tap this button to return to the home page



Tap this button to have the book read to you

Swipe or tap the edge of the screen to move between pages

Try tapping on different objects on screen to see what happens!



# ASSIGNMENT 1



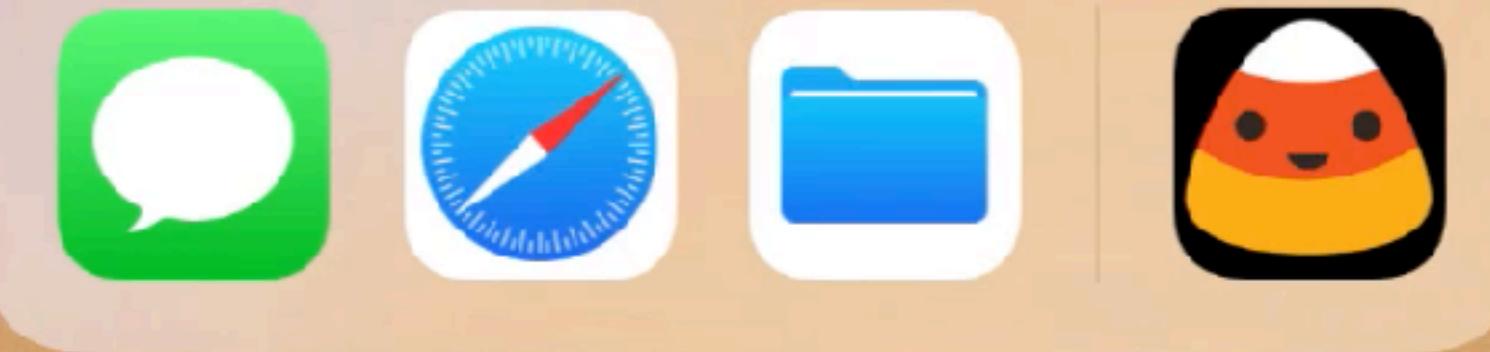
Carrier ⌘

5:23 PM

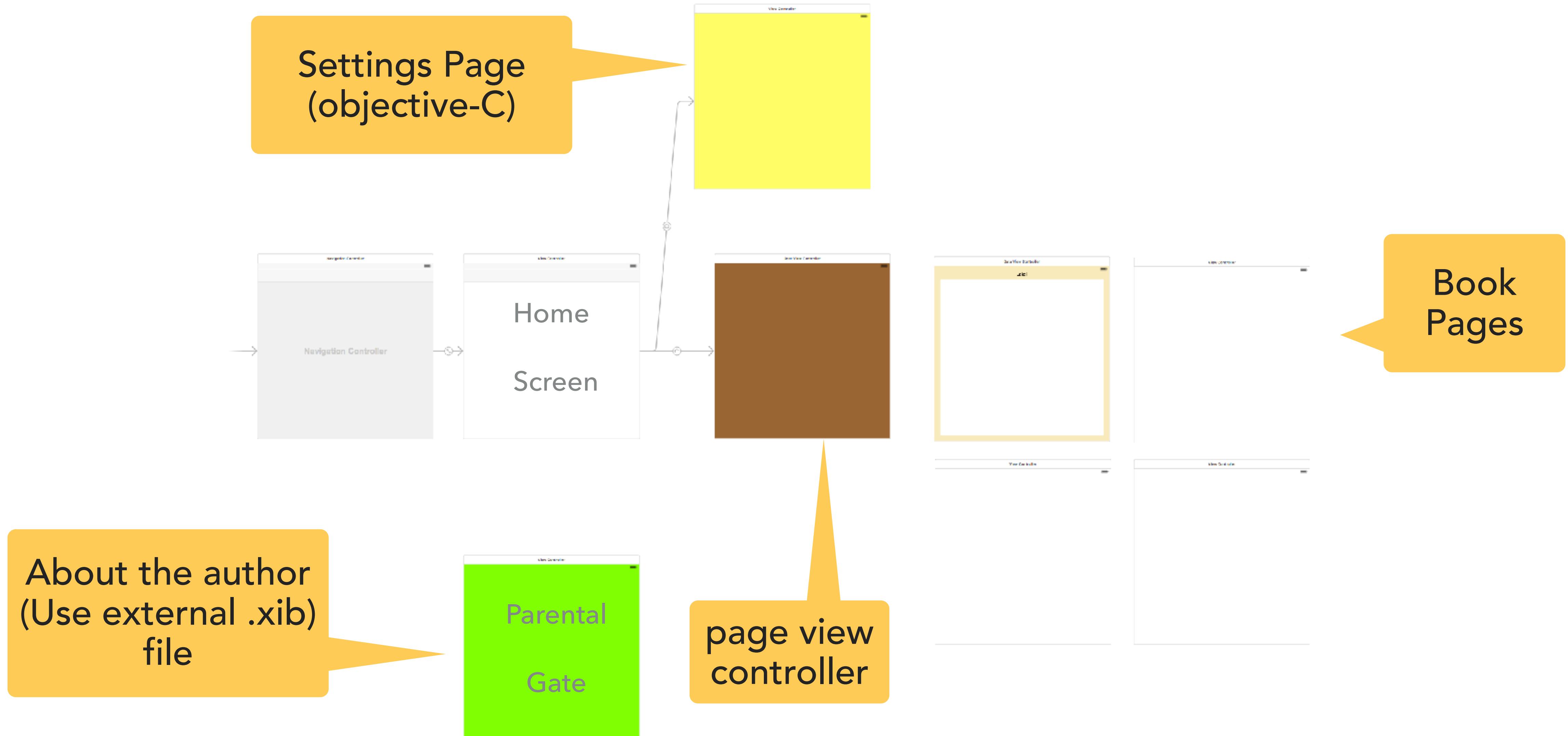
100% 



Children\_Book



# OVERALL REQUIREMENTS



# OVERALL REQUIREMENTS

- “Home Page” that represents the beginning of the book
- “Settings page” for “Read to Me” options (objective-c)
  - Automatically play on each page
  - Tap-to-play
  - Store these preferences in `UserDefaults`
- “About the Author” page protected by Parental Gate (external .xib file)
- Store currently reading page so that book opens to last view paged
- Create an App video of your book

# BOOK PAGES

- Every content page must implement:
  - “Read to me” functionality
    - Record sound using Audacity, Garage Band, iPhone and email to self (UITextKit, AVSpeechSynthesis, next week)
    - Words should highlight as they are read
    - Start automatically or by button tap (depending on user preference in settings)
  - Picture and text
    - You must use `UIDynamics` for on-screen element (next week)
    - `UIGestureRecognizer` of on-screen element
    - UIViewPropertyAnimator based animations
    - Tap to hear sound effects
  - Button to return to home page

# How Jane Met John

Read the Story

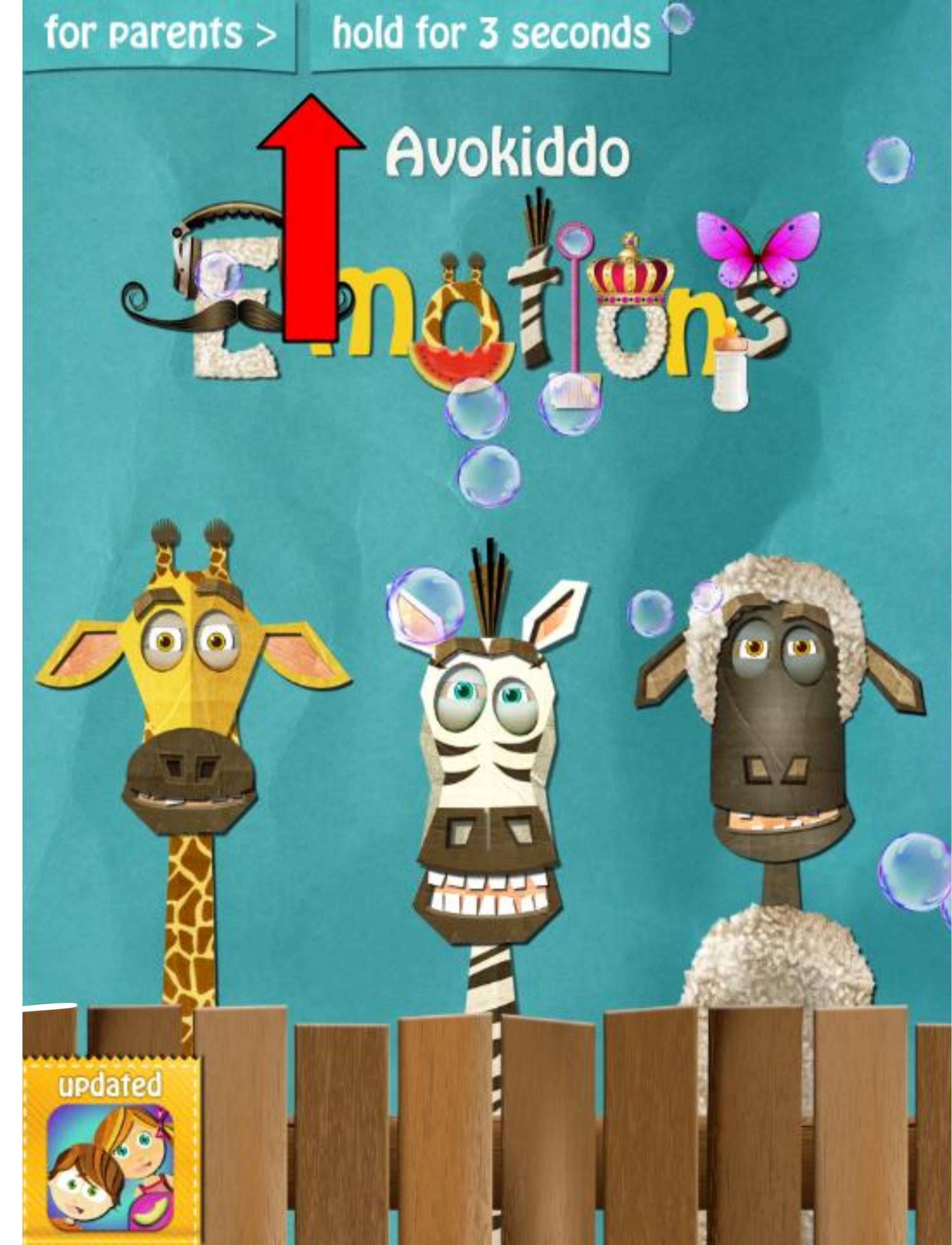
About the Author

Settings

# PARENTAL GATE

SUBTITLE

- Features are intended for parents, not kids
  - In-app purchase
  - Signup for news
  - Social Media
- Prevent kids from accessing inappropriate content



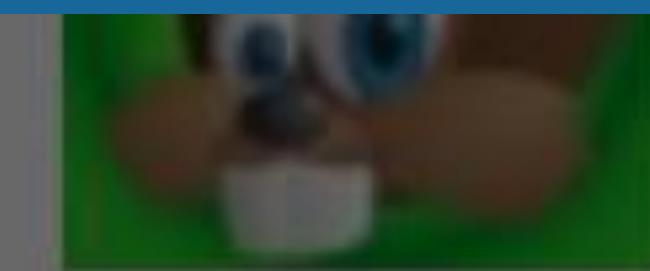


Start      About      Settings

# PARENTAL GATE

- Implementation used to be a best practice, now it is required
  - 24. Kids Apps
    - 24.3- Apps primarily intended for use by kids under 13 must get parental permission or use a parental gate before allowing the user to link out of the app or engage in commerce
- This requires gating when linking out of the app or engaging in commerce such as selling in-app items.

# PARENTAL GATE



More Fun!



ARE YOU AN ADULT?

What is **20 - 5 =**

A white rectangular input field for entering the answer to the subtraction problem.

**Submit**



# PARENTAL GATE

Touch the **triangle** for 2  
seconds



# PARENTAL GATE

## REQUIREMENTS

- Implement a Parental Gate for your book app
  - Triggered when the user tries to view the “About the Author” screen
  - In practice the credits screen would typically have a link to social media
- Custom finger drawing view for validator
- Needs to use custom error handling do-try-throw
  - Custom ErrorType

# SETTINGS PAGE

## REQUIREMENTS

- Settings page in objective-c 😳
- Allow user to select preference to automatically read the page
  - Default should be "on"



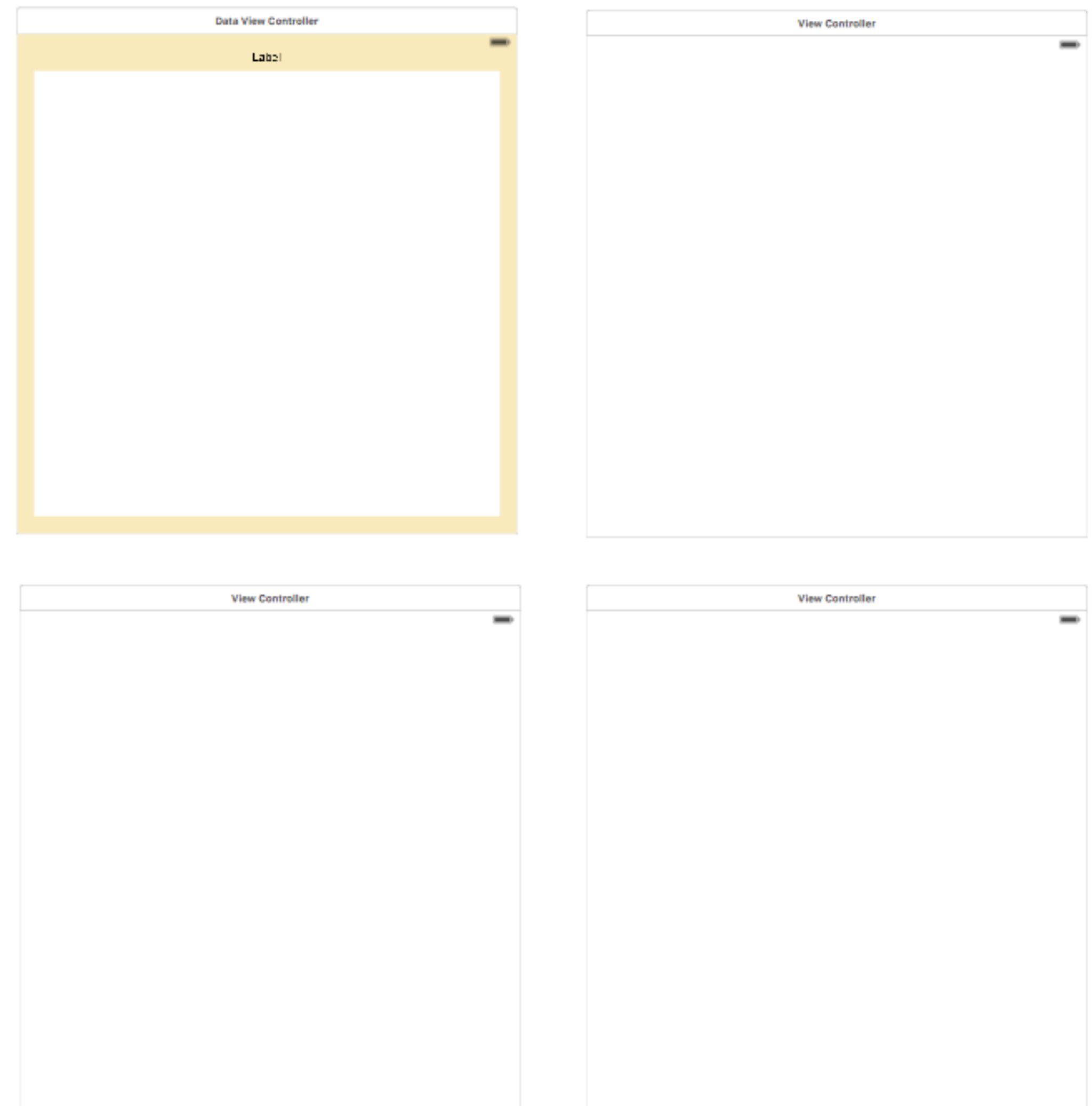
Read to me



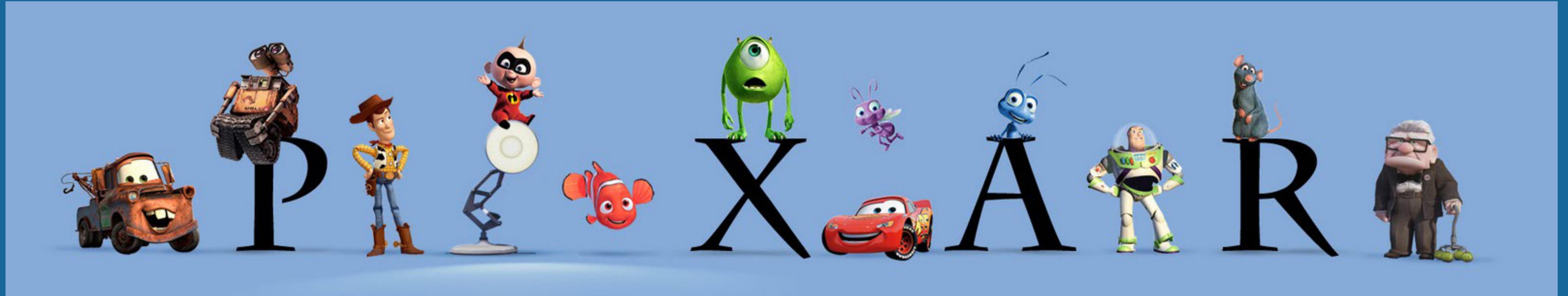
# BOOK PAGE OPTIONS

SUBTITLE

- Implementations options
  - View controller per page
  - Create view controller templates
  - Design 3 different styles of pages
  - Each page is a dictionary that includes images paths, text, sound file paths, animation start/stop frames



# DUE DATE



- Due 2 Weeks
  - No late homework accepted
- “It’s all about story...story, story, story.” ~John Lasseter

# RECOMMENDED WORK FOR THIS WEEK

- Page View Controller
  - Get basic layout
  - Back to home button
  - Save current page
  - How to handle interrupted page turns
- Book pages
  - Draw them out
  - Sound effects
  - Interactive element
  - Text-to-speech

# GOING OVER NEXT WEEK

- Settings screen
  - Obj-c first app from Apple
  - Obj-c bridging header
- Parental gate
- App Preview
- Picture and text
  - `UIDynamics` for on-screen element
  - UIViewPropertyAnimator

# SESSION 1A



# ADVANCED iOS APPLICATION DEVELOPMENT

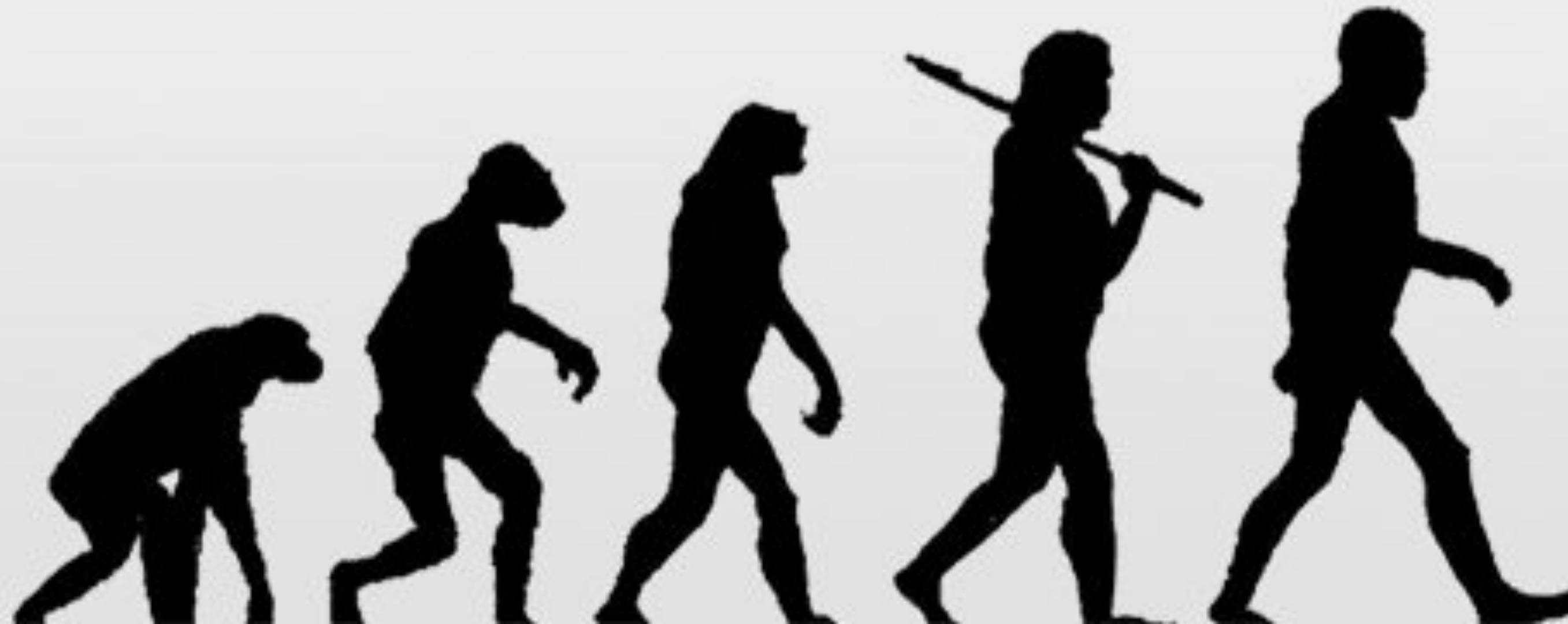
---

MPCS 51032 • SPRING 2020 • SESSION 1A

# VIEW CONTROLLER EVOLUTION

# VIEW CONTROLLER EVOLUTION

iPhone  
Evolution



# **THE VIEW CONTROLLER**

# THE VIEW CONTROLLER

IN THE BEGINNING...

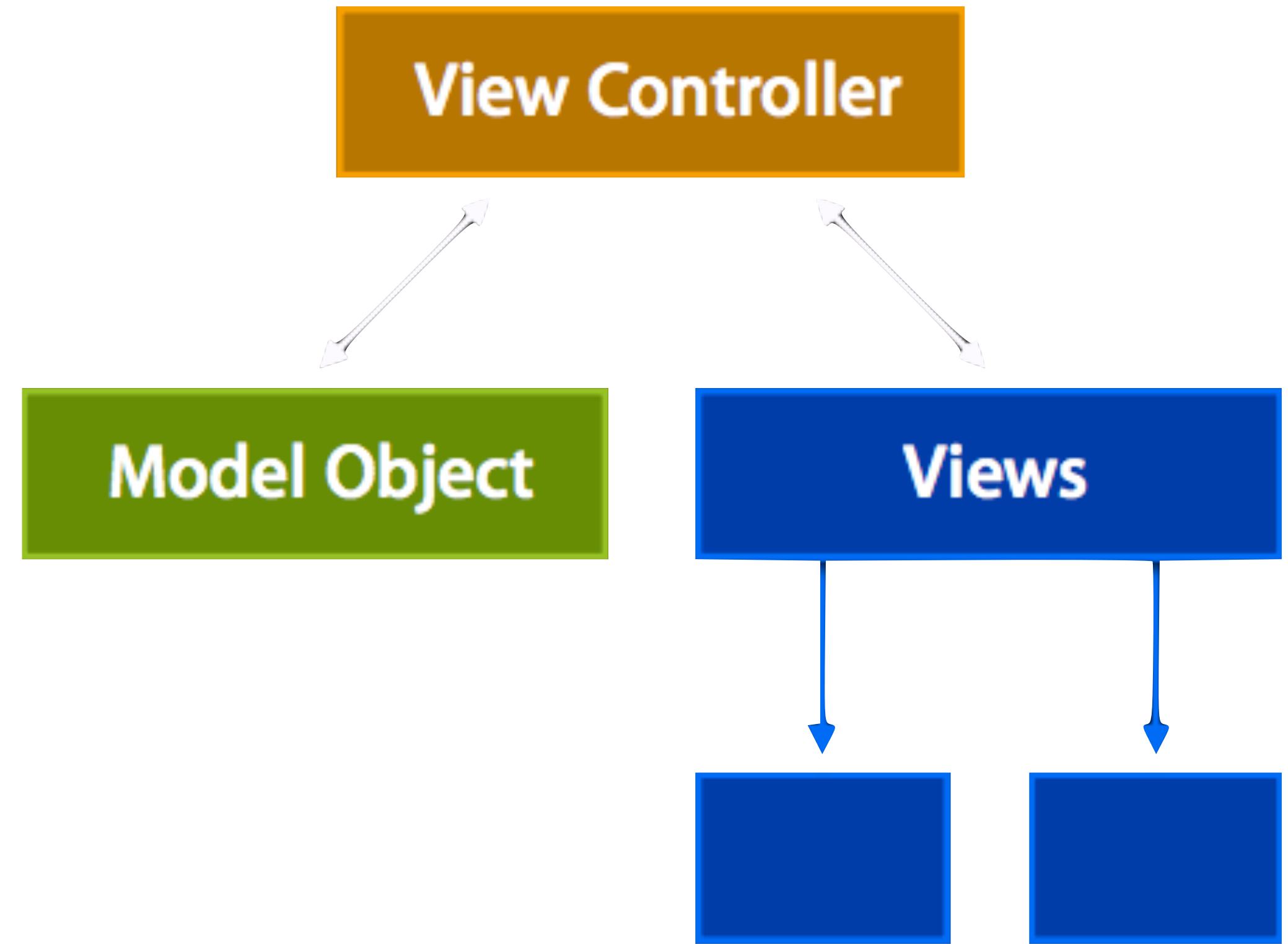
- `UIViewController`s jobs
  - Manages “screenful of objects”
  - Manage view hierarchy
  - Applications flow between view controllers



# THE VIEW CONTROLLER

CUSTOM VIEW CONTROLLERS IMPLEMENTATION

- Subclass `UIViewController`
- Associate the view controller with a view hierarchy
- Override select APIs to suit your needs
- Add your application logic



# THE VIEW CONTROLLER

## CUSTOM VIEW CONTROLLERS IMPLEMENTATION

- Commonly overridden methods
  - Loading callbacks
    - `viewDidLoad()`
    - `viewDidUnload()`
  - Appearance callbacks
    - `viewWillAppear()`
    - `viewDidAppear()`
    - `viewWillDisappear()`
    - `viewDidDisappear()`

```
// ViewController.swift
// Session1-PageViewController
//
// Created by T. Andrew Binkowski on 4/2/17.
// Copyright © 2017 T. Andrew Binkowski. All rights reserved.

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    /*
    // MARK: - Navigation

    // In a storyboard-based application, you will often want to
    override func prepare(for segue: UIStoryboardSegue, sender:
        // Get the new view controller using segue.destinationVi
        // Pass the selected object to the new view controller.
    }
    */
}
```

# THE VIEW CONTROLLER

## CUSTOM VIEW CONTROLLERS IMPLEMENTATION

Other than  
segues

- How do view controllers get on/off the screen?

- By container controllers
- By presentation

```
presentViewController(gvc, animated: true, completion: nil)
```

- By dismissal

```
presentingViewController?.dismissViewControllerAnimated(true, completion: nil)
```

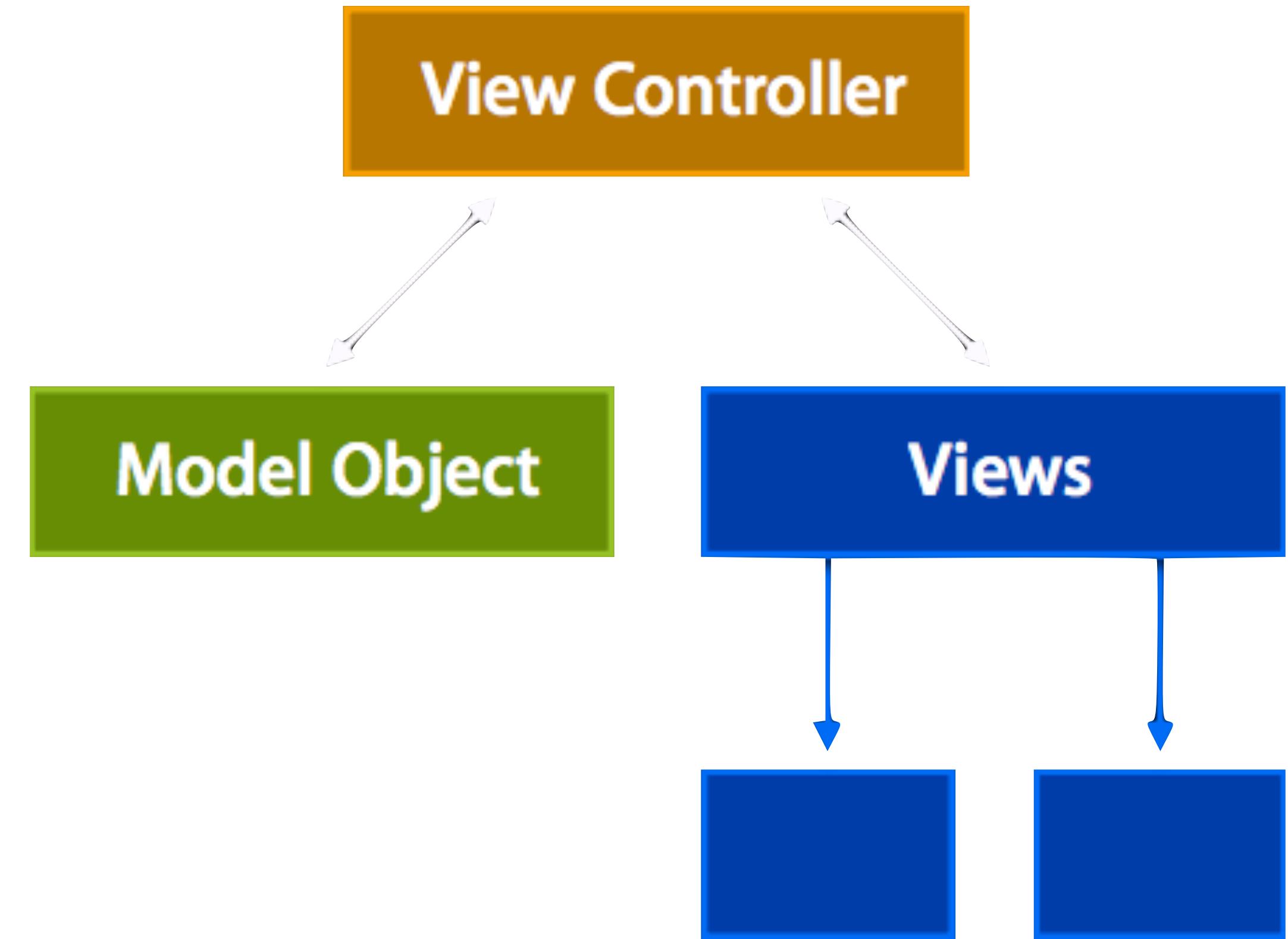
- By direct view manipulation

```
window.rootViewController = rootViewController;
```

# THE VIEW CONTROLLER

SUBTITLE

- `UIViewController` advantages
  - Consistent with Apple apps
  - Easily reusable
  - Simplify many of the tasks for managing views and models



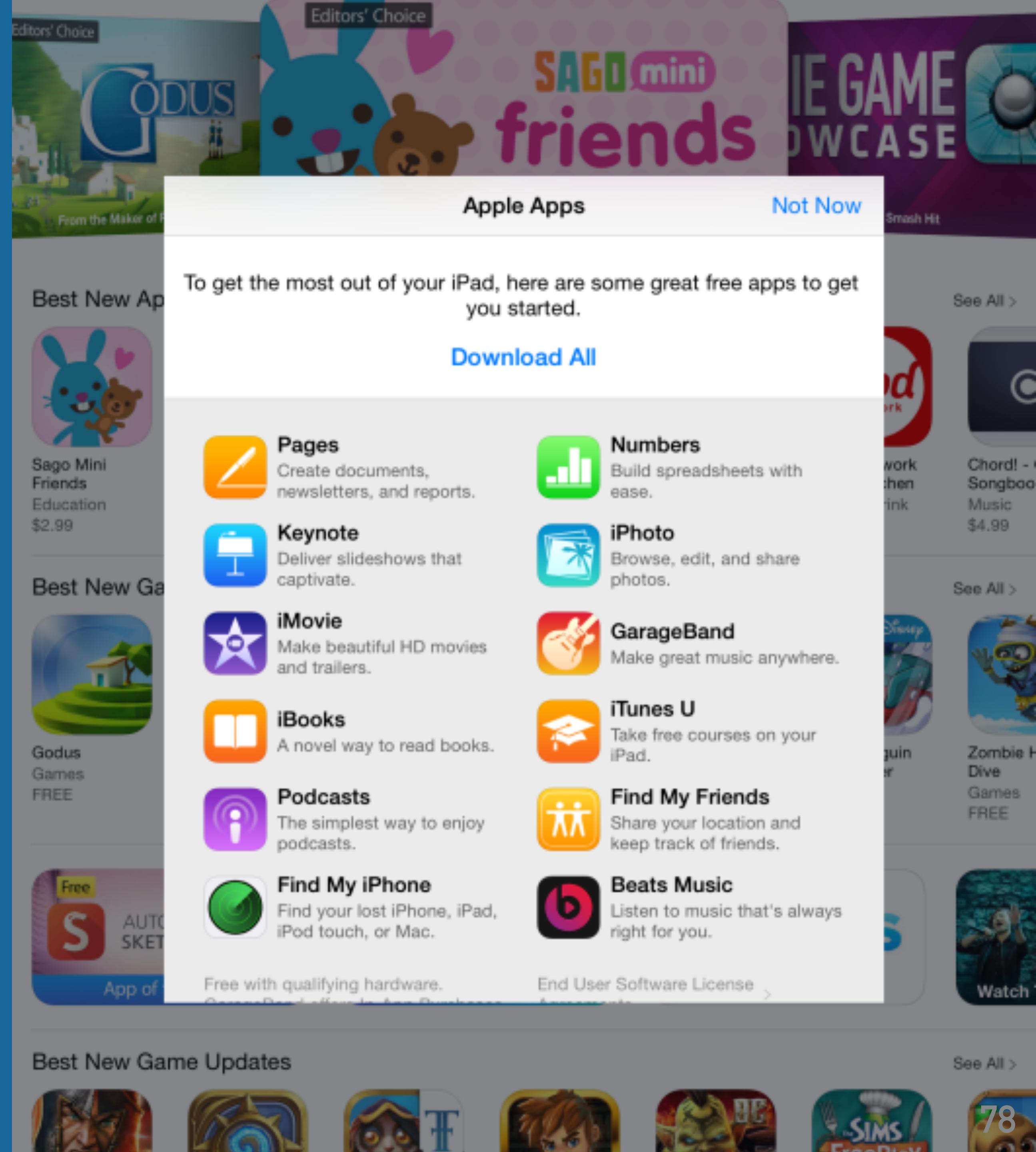
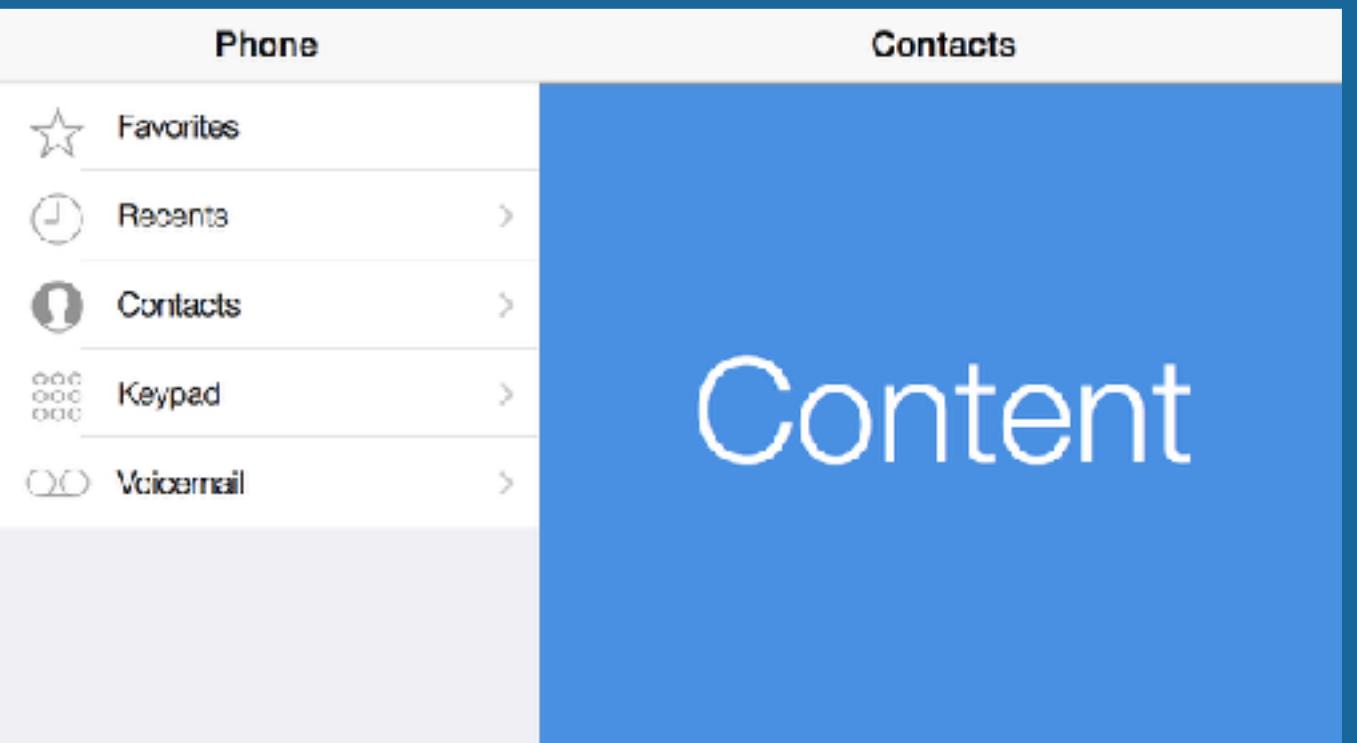
# VIEW CONTROLLERS

- For many interfaces it "makes sense"
- Screenful of content

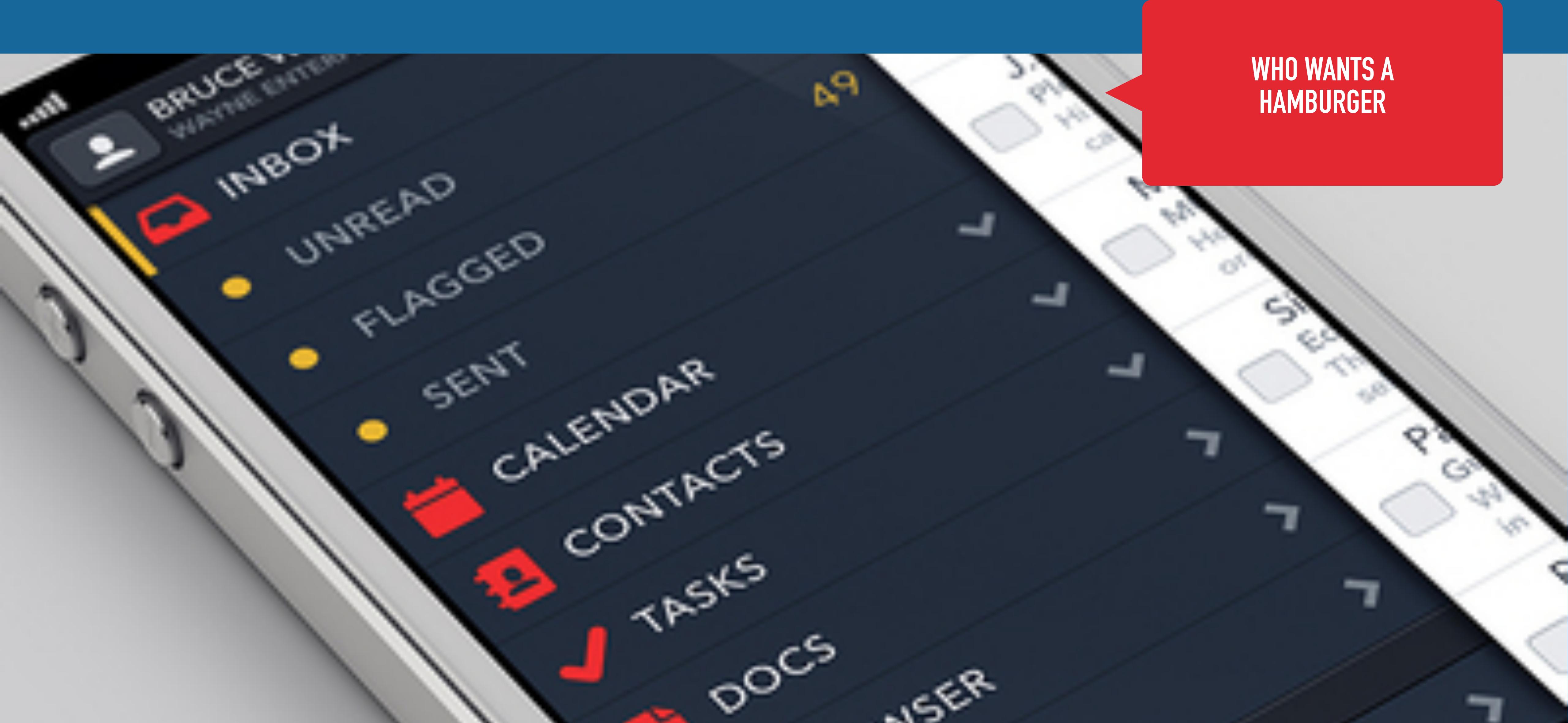


# VIEW CONTROLLERS

- “Screenful of content” getting harder to define



# VIEW CONTROLLERS



WHO WANTS A  
HAMBURGER

# VIEW CONTROLLERS

- Special controllers manage “unit of content” not “pageful of content”
  - UISplitViewController
  - UITabBarController

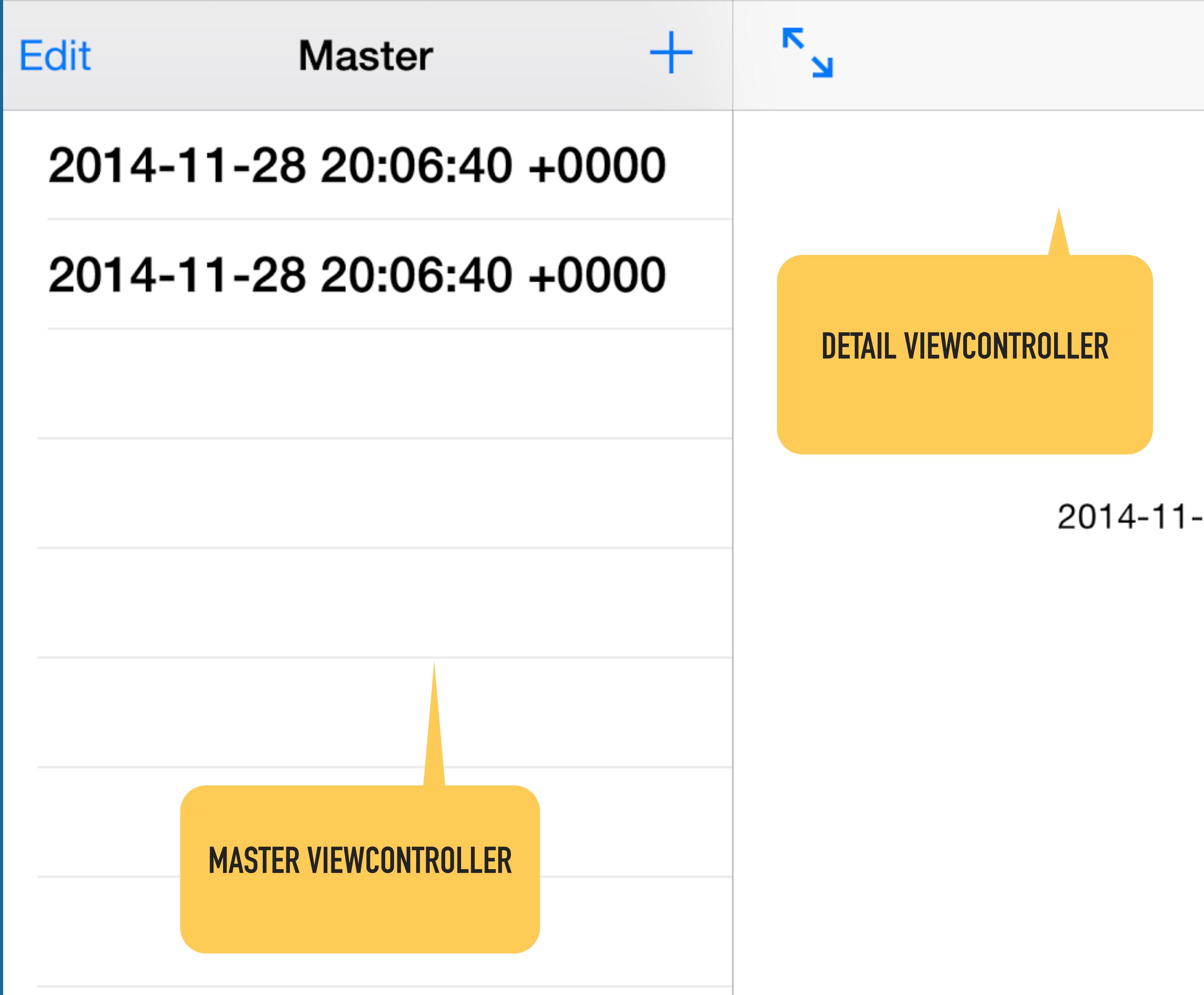
The image shows a Twitter feed with three visible tweets:

- Wall Street Journal @WSJ** 44s ago  
How one of the most controversial doctors in history steered U.S. toward lobotomizing thousands of veterans: [on.wsj.com/1h3iR4E](http://on.wsj.com/1h3iR4E)  
Retweet Retweet Star
- Atlassian HipChat @Hip...** 12/6/13  
Check out what's new in HipChat -- New Mentions, Commands & More! [ow.ly/rwE7d](http://ow.ly/rwE7d)  
Promoted by Atlassian HipChat Retweet Retweet Star
- Johnnie Manzari @johnnie** 1m ago  
The best thing about Silicon Valley is its brilliant unpredictability; e.g. one of the leading car makers is now headquartered in Palo Alto.  
Retweet Retweet Star

At the bottom, it says "John Herrman retweeted". The navigation bar at the bottom includes "Timelines", "Notifications" (with 20+ notifications), "Messages" (with 20+ messages), and "Me".

# AND NOW...

- The root view controller does manage a “screenful of content”
- `window.rootViewController = aViewController`



# VIEW CONTROLLERS

## SUMMARY

- View controllers are just controllers—the “C” in MVC
- View controllers manage a view hierarchy—not a single view!
- View controllers are typically self contained (reusable)
- View controllers connect and support common iOS application flows
- More complex aesthetics or application flows need more flexibility

**CONTENT VS.  
CONTAINER  
VIEW CONTROLLERS**

# VIEW CONTROLLER CONTAINERS

- And now...
  - Content view controller
  - Your custom view
  - Container view controllers
    - Split view controller
    - Tab bar controllers

Wall Street Journal @WSJ 44s  
How one of the most controversial doctors in history steered U.S. toward lobotomizing thousands of veterans: [on.wsj.com/1h3iR4E](http://on.wsj.com/1h3iR4E)

Atlassian HipChat @Hip... 12/6/13  
Check out what's new in HipChat -- New Mentions, Commands & More! [ow.ly/rwE7d](http://ow.ly/rwE7d)

Promoted by Atlassian HipChat

Johnnie Manzari @johnnie 1m  
The best thing about Silicon Valley is its brilliant unpredictability; e.g. one of the leading car makers is now headquartered in Palo Alto.

John Herrman retweeted

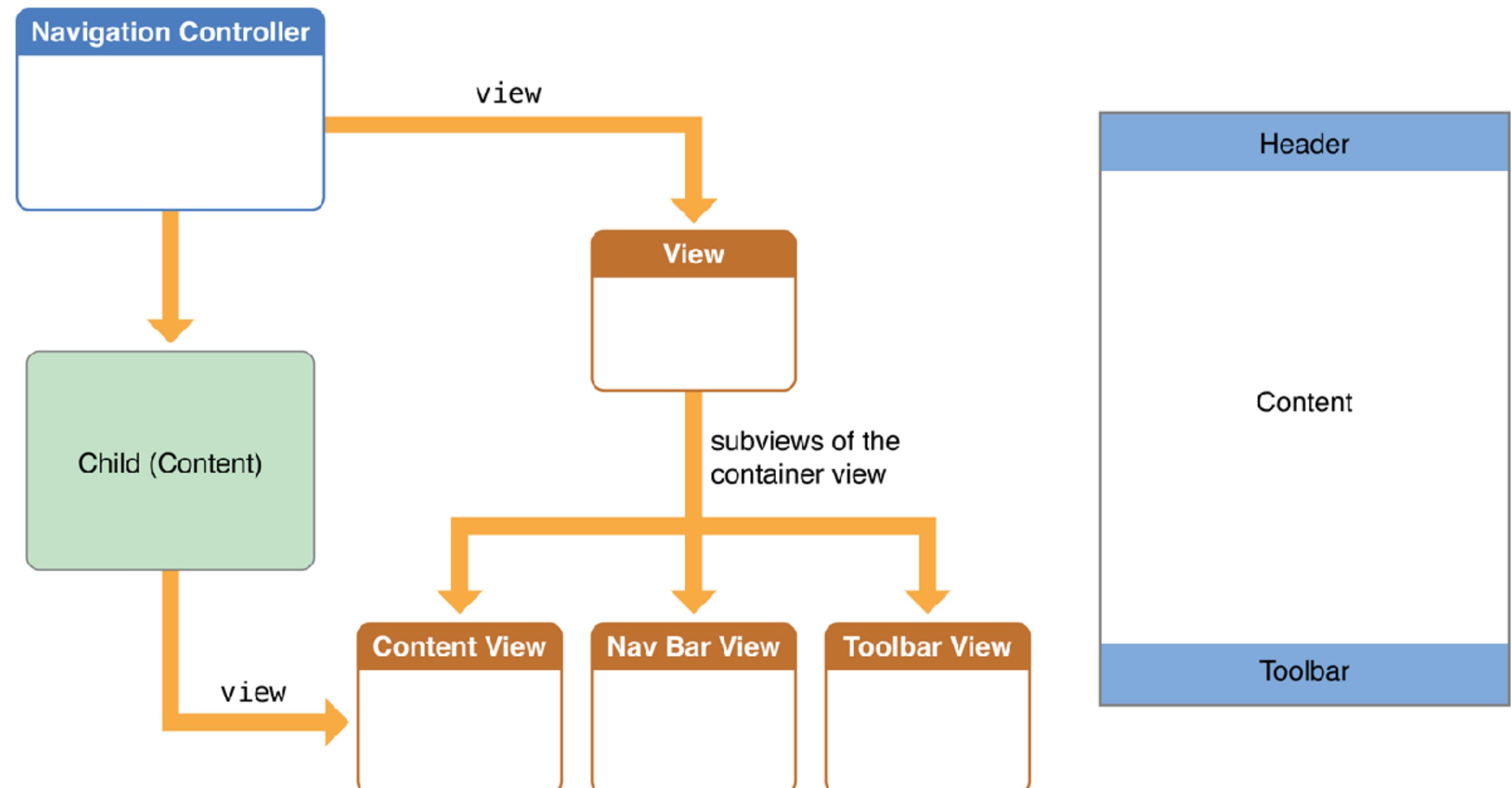
Timelines 20+ Notifications 20+ Messages Me

# VIEW CONTROLLER CONTAINERS

- Container Controllers
  - Manage a hierarchy of child views
  - UITabBarController
  - UINavigationController
  - UISplitViewController
- Content Controllers
  - Manage individual screens
  - Can be more than one view controller per screen (as seen in a UISplitViewController)

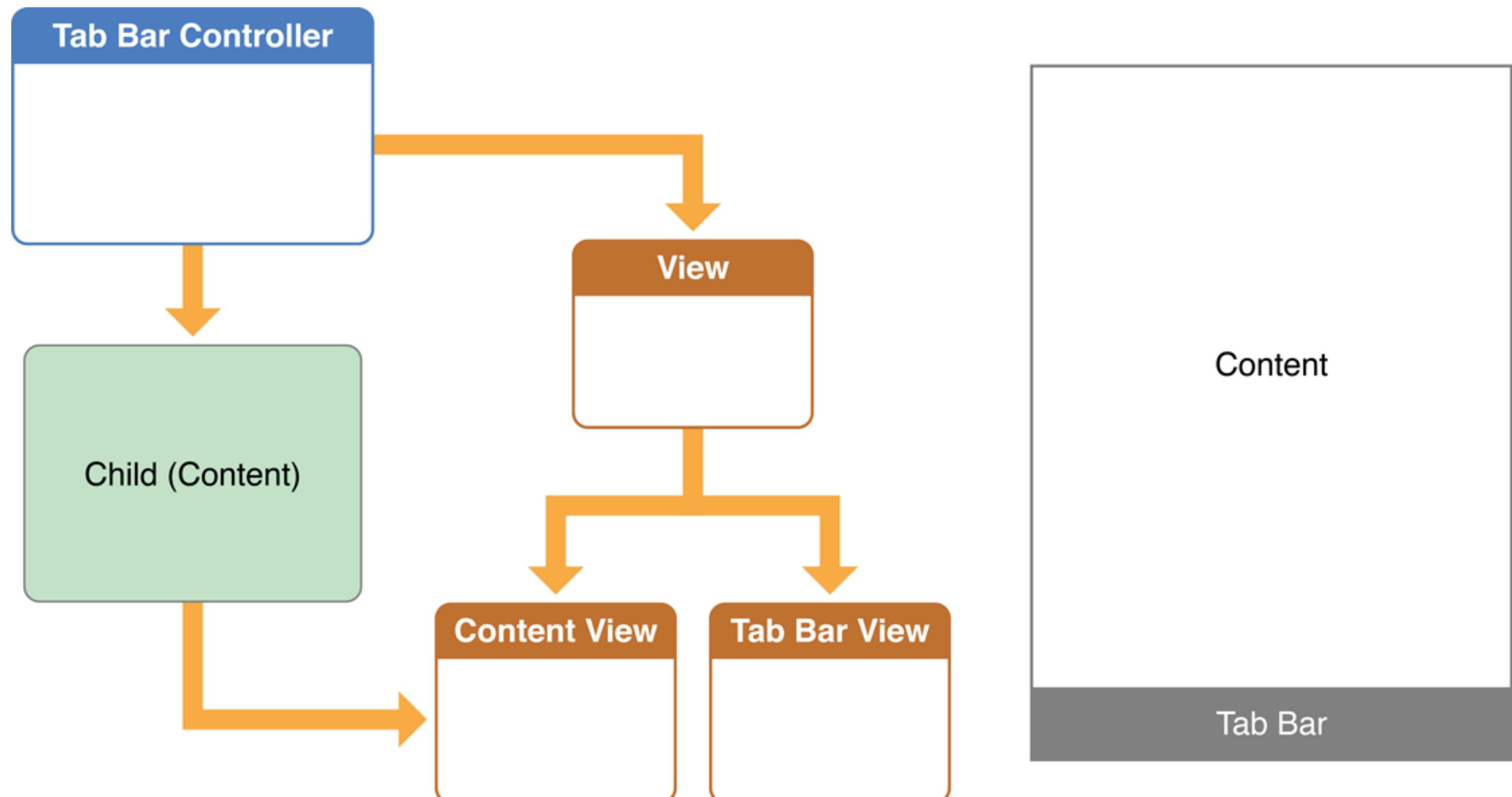
# VIEW CONTROLLER CONTAINERS

- UINavigationController
  - View controllers on a stack that you push/pop them off to view
  - Transition animation where you may see multiple view controllers
  - Contain additional content views
    - Navigation bar
    - Tool bar



# VIEW CONTROLLER CONTAINERS

- UITabBarController
  - Array of view controllers
  - Only one shown at a time



# VIEW CONTROLLER CONTAINERS

lockergnome.net/questions//5299/apple-screwed-up

**keithdsouza** 8 mins  
WTF Air India put Chennai passengers in Hyderabad plane, are you fucking kidding me :|

**BlogsDNA** 9 mins  
Twitter for iPad is here with Gestures and New UI <http://bit.ly/bycqpu>

**Techmeme**  
Skype Mobile for Android and Wifi (but still Verizon) (@philnickinson /... <http://j.mp/cuL4WM> <http://techme.me/A0F4>)

**SEOMoz** 11 mins  
A few shots from the #mozinar ( <http://bit.ly/ctkR0u> ) /via @seomoz

A red arrow points from the "Techmeme" tweet towards a red callout box.

CUSTOM VIEW CONTROLLER

**Chris Pirillo**  
@ChrisPirillo  
#756278

**Unfollow**

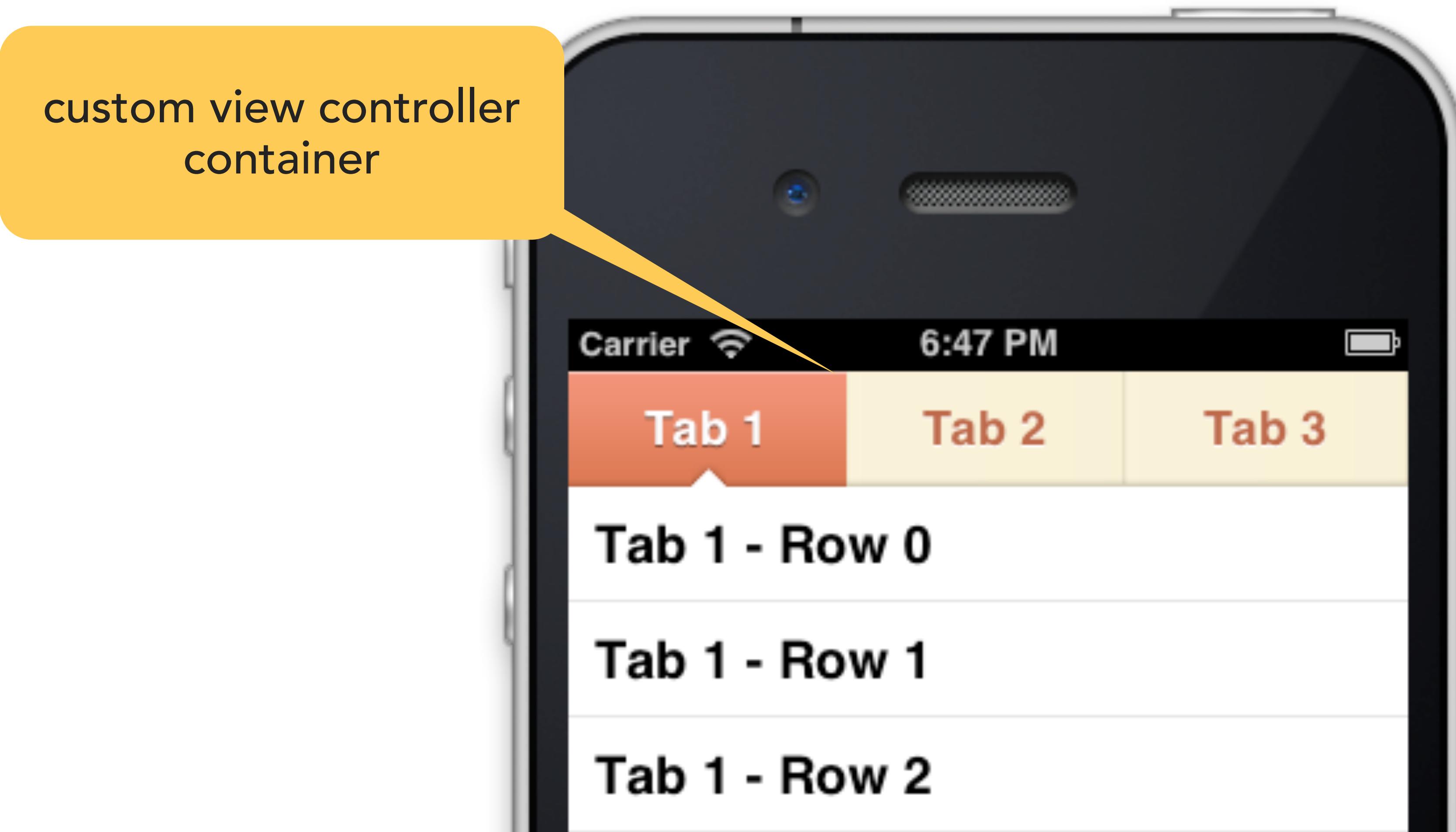
**bio** I've been making things happen online since 1992 - a media-friendly geek who produces content and catalyzes communities.

**location** Seattle, WA

**web** [chris.pirillo.com/](http://chris.pirillo.com/)

**Tweets** **@Mentions** **Favorites**

# VIEW CONTROLLER CONTAINERS



<https://github.com/hollance/MHTabBarController>

# VIEW CONTROLLER CONTAINERS

The image shows a Twitter application interface. On the left is a dark grey sidebar with white text and icons for navigation: @ Mentions, Messages, Favorites, Search, Profile, Lists, Retweets, Mute Filters, and Settings. On the right is the main content area, which displays a list of tweets. The first tweet is from David Kaneda (@flyosity) with the text "Ordered it yesterday :)". The second tweet is from Dane Baker (@flyosity) with the text "Totally already preord". The third tweet is from Mike Rundle with the text "Beautiful new iPad stand uses". At the bottom of the screen are five action buttons: Reply, Retweet, Favorite (which is highlighted with a yellow star), Actions, and Details.

UISplitViewController or  
custom container view?

# **VIEW CONTROLLER CONTAINMENT**

# VIEW CONTROLLER CONTAINERS

**Blue**  
View color



**Blue Arrow**  
Subview relationship



**Gold**  
View controller color

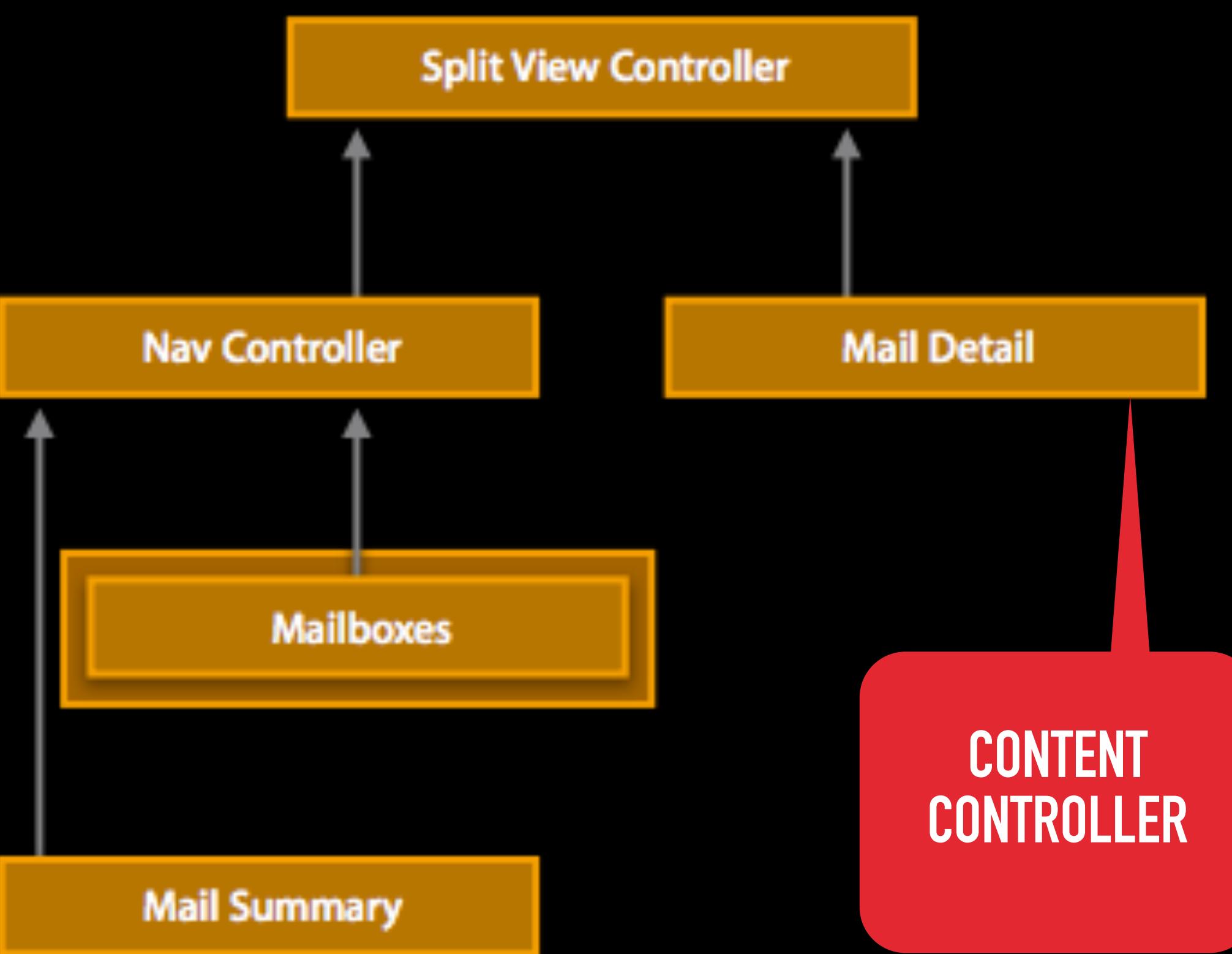
Split View Controller

**Gray Arrow**  
Parent view controller  
relationship



# VIEW CONTROLLER CONTAINERS

CONTENT CONTROLLER



CONTENT  
CONTROLLER

Image via Apple

# VIEW CONTROLLER CONTAINERS

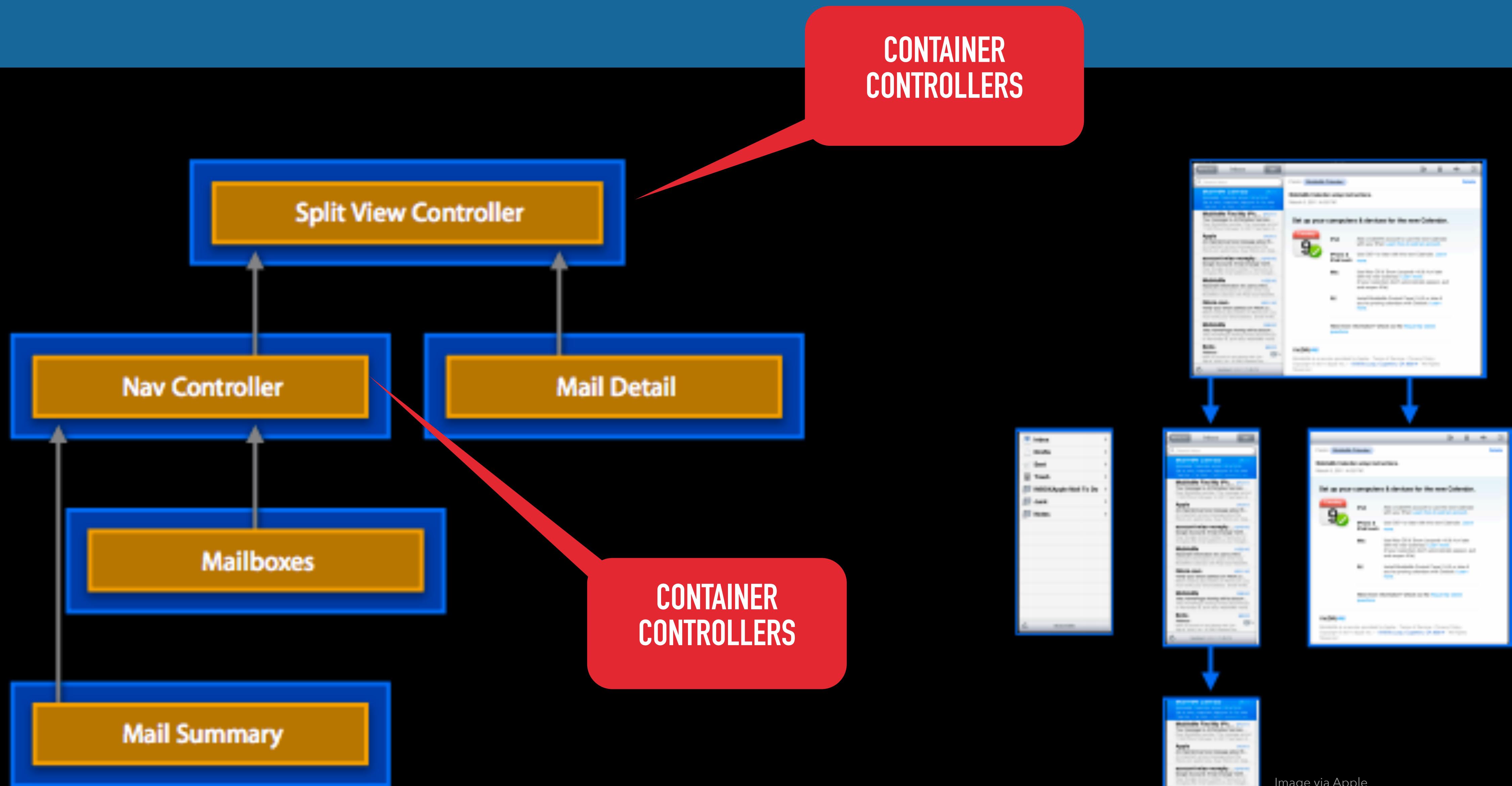
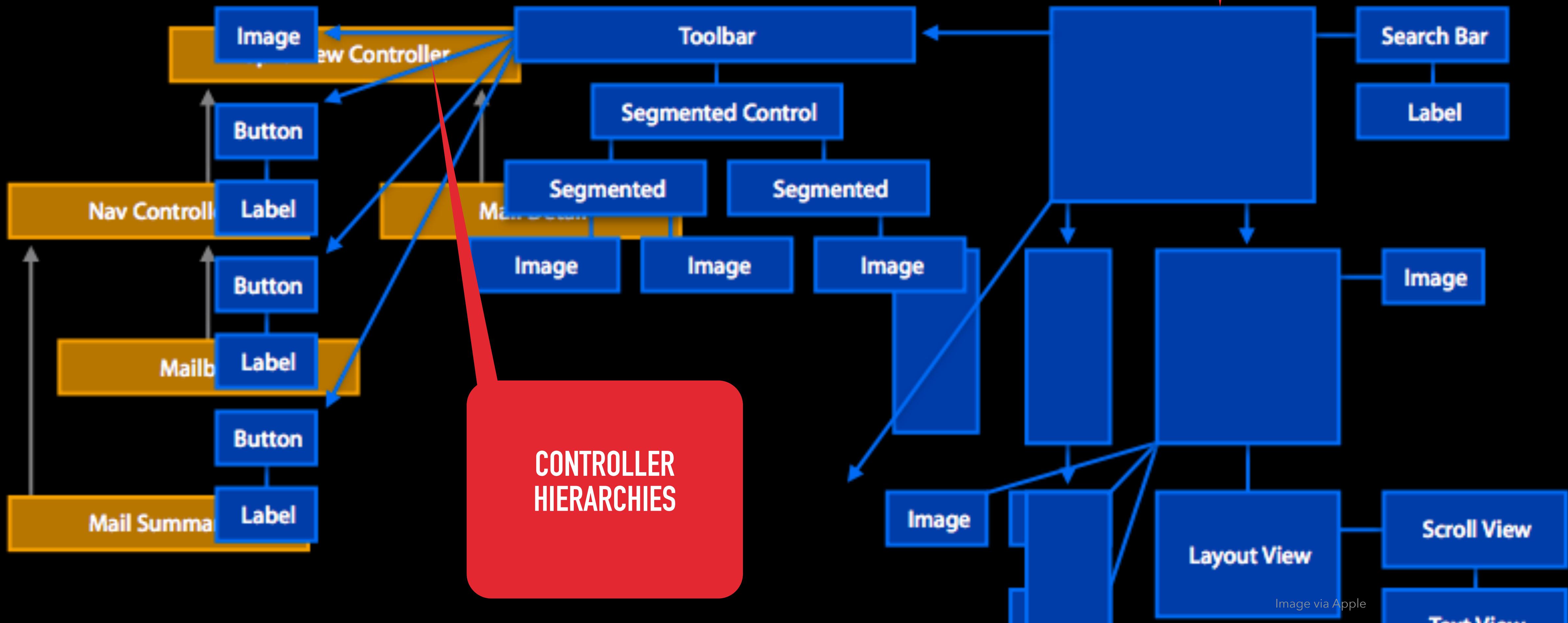


Image via Apple

# VIEW CONTROLLER CONTAINERS



VIEW HIERARCHIES

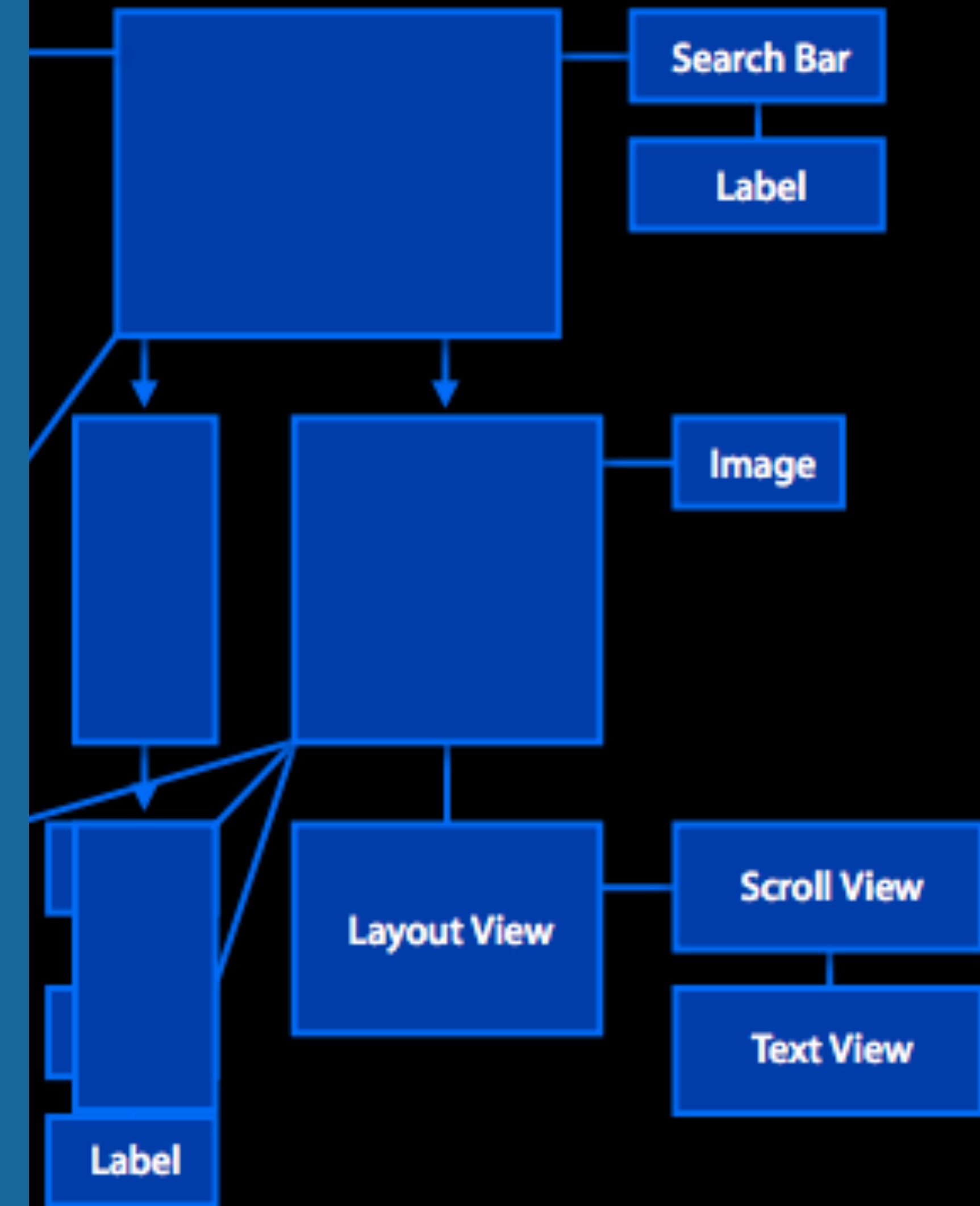


CONTROLLER  
HIERARCHIES

Image via Apple

# VIEW CONTROLLER CONTAINERS

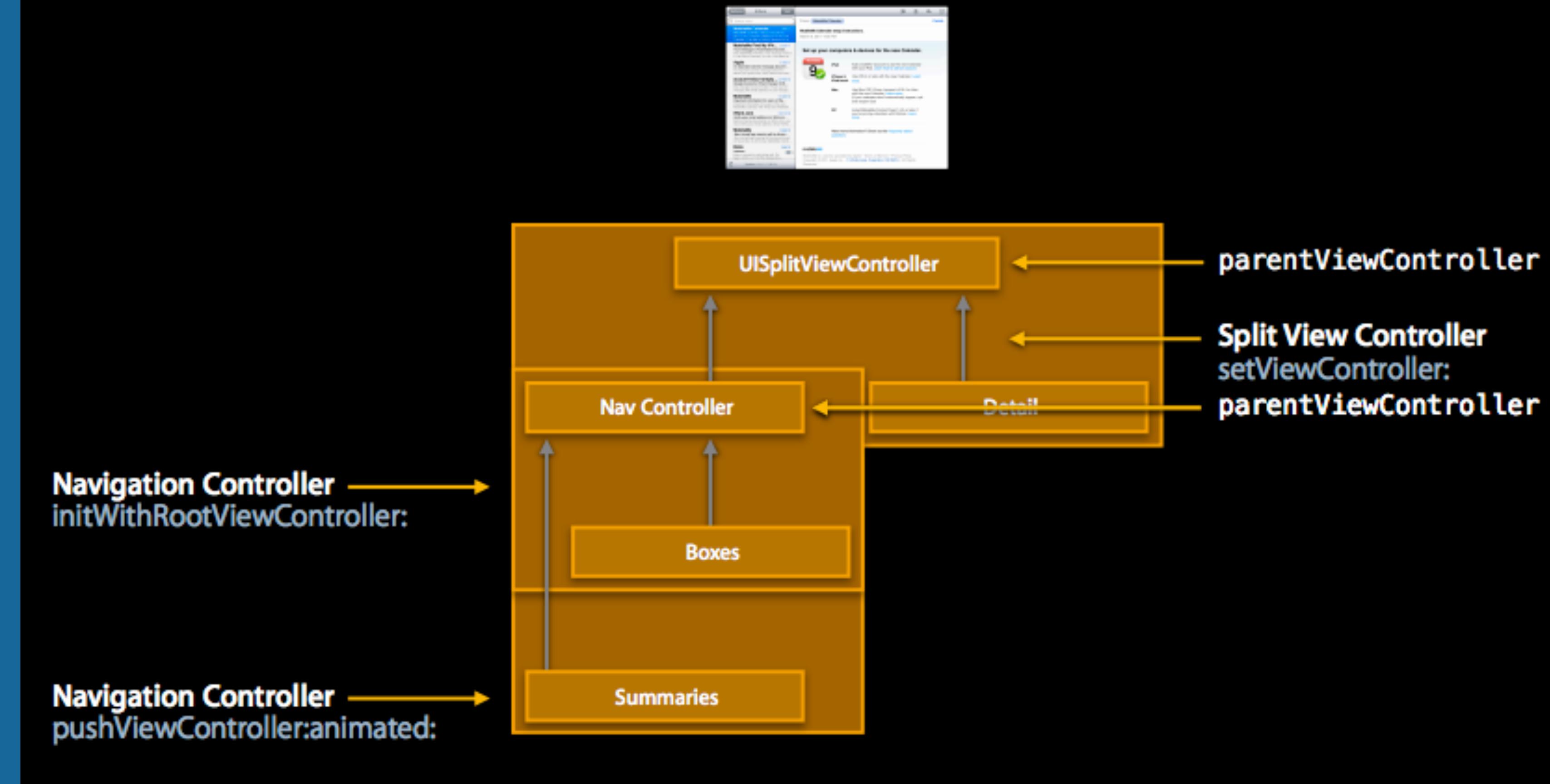
- Container views allow a view to have child views which are in turn managed by their own view controllers



# VIEW CONTROLLER CONTAINERS

- Container Controllers are responsible for child/parent relationships between view controllers
  - Pushing view controllers
  - Remove view controllers

## API and the controller hierarchy



# VIEW CONTROLLER CONTAINERS

## Tasks

Configuring a View Controller  
Using Nib Files

Interacting with Storyboards  
and Segues

Managing the View

Presenting View Controllers

Supporting Custom Transitions  
and Presentations

Responding to View Events

Configuring the View's Layout  
Behavior

Testing for Specific Kinds of  
View Transitions

Configuring the View Rotation  
Settings

Adapting to Environment  
Changes

Managing Child View  
Controllers in a Custom  
Container

Language: [Objective-C](#) [Swift](#) [Both](#) [On This Page](#) [Options](#)

## Managing Child View Controllers in a Custom Container

```
childViewControllers  
addChildViewController(_:)  
removeFromParentViewController()  
transitionFromViewController(_:toViewController:duration:options:animations:completion:)  
shouldAutomaticallyForwardAppearanceMethods()  
beginAppearanceTransition(_:animated:)  
endAppearanceTransition()  
setOverrideTraitCollection(_:forChildViewController:)  
overrideTraitCollectionForChildViewController(_:)
```

## Responding to Containment Events

```
willMoveToParentViewController(_:)
```

# CONNECTING VIEW CONTROLLERS

## SUBTITLE

- Add a view controllers view to containers view hierarchy
  - 1. Calls the container's `addChildViewController:` method to add the child .
    - This calls the child's `willMoveToParentViewController:` method automatically
  - 2. It accesses the child's view property to retrieve the view and adds it to its own view hierarchy.
    - Container sets the child's size and position before adding the view; order matters
  - 3. Call the child's `didMoveToParentViewController:` method to signal that the operation is complete

```
// Create instance of view controller and color it.
let gvc = ColorViewController()
gvc.view.backgroundColor = UIColor.redColor()

// 1.
addChildViewController(gvc)

// 2.
gvc.view.frame = CGRectMake(50, 50, 200, 200)
view.addSubview(gvc.view)

// 3.
gvc.didMoveToParentViewController(self)
```

# CONNECTING VIEW CONTROLLERS

## SUBTITLE

- Add a view controllers view to containers view hierarchy
  - 1. Calls the container's addChildViewController: method to add the child .
    - This calls the child's `willMoveToParentViewController:` method automatically
  - 2. It accesses the child's view property to retrieve the view and adds it to its own view hierarchy.
    - Container sets the child's size and position before adding the view; order matters
  - 3. Call the child's didMoveToParentViewController: method to signal that the operation is complete

```
// Create instance of view controller and color it.
let gvc = ColorViewController()
gvc.view.backgroundColor = UIColor.redColor()

// 1.
addChildViewController(gvc)

// 2.
gvc.view.frame = CGRectMake(50, 50, 200, 200)
view.addSubview(gvc.view)

// 3.
gvc.didMoveToParentViewController(self)
```

# CONNECTING VIEW CONTROLLERS

SUBTITLE

- Add a view controllers view to containers view hierarchy
  - 1. Calls the container's addChildViewController: method to add the child .
    - This calls the child's `willMoveToParentViewController:` method automatically
  - 2. It accesses the child's view property to retrieve the view and adds it to its own view hierarchy.
    - Container sets the child's size and position before adding the view; order matters
  - 3. Call the child's didMoveToParentViewController: method to signal that the operation is complete

```
// Create instance of view controller and color it.
let gvc = ColorViewController()
gvc.view.backgroundColor = UIColor.redColor()

// 1.
addChildViewController(gvc)

// 2.
gvc.view.frame = CGRectMake(50, 50, 200, 200)
view.addSubview(gvc.view)

// 3.
gvc.didMoveToParentViewController(self)
```

# CONNECTING VIEW CONTROLLERS

ADD A CHILD VIEW CONTROLLER

```
// Create instance of view controller and color it.  
let gvc = ColorViewController()  
gvc.view.backgroundColor = UIColor.redColor()  
  
// 1.  
addChildViewController(gvc)  
  
// 2.  
gvc.view.frame = CGRectMake(50, 50, 200, 200)  
view.addSubview(gvc.view)  
  
// 3.  
gvc.didMoveToParentViewController(self)
```

Show Green

Add Container



# CONNECTING VIEW CONTROLLERS

## REMOVING A VIEW CONTROLLER FROM A CONTAINER

```
// 1. Calls the child's willMoveToParentViewController: method with a  
// parameter of nil to tell the child that it is being removed.  
sender.willMoveToParentViewController(nil)  
  
// 2. Clean up the view hierarchy  
sender.view.removeFromSuperview()  
  
// 3. Calls the child's removeFromParentViewController method to remove it  
// from the container.  
sender.removeFromParentViewController()
```

# CONNECTING VIEW CONTROLLERS

## REMOVING A VIEW CONTROLLER FROM A CONTAINER

- Remove a view controllers view to containers view hierarchy

- 1. Calls the child's `willMoveToParentViewController:` method with a parameter of nil to tell the child that it is being removed.
- 3. Calling the `removeFromParentViewController` method automatically calls the child's `didMoveToParentViewController:` method

```
// 1. Calls the child's willMoveToParentViewController: method  
// parameter of nil to tell the child that it is being removed  
sender.willMoveToParentViewController(nil)  
  
// 2. Clean up the view hierarchy  
sender.view.removeFromSuperview()  
  
// 3. Calls the child's removeFromParentViewController method  
// from the container.  
sender.removeFromParentViewController()
```

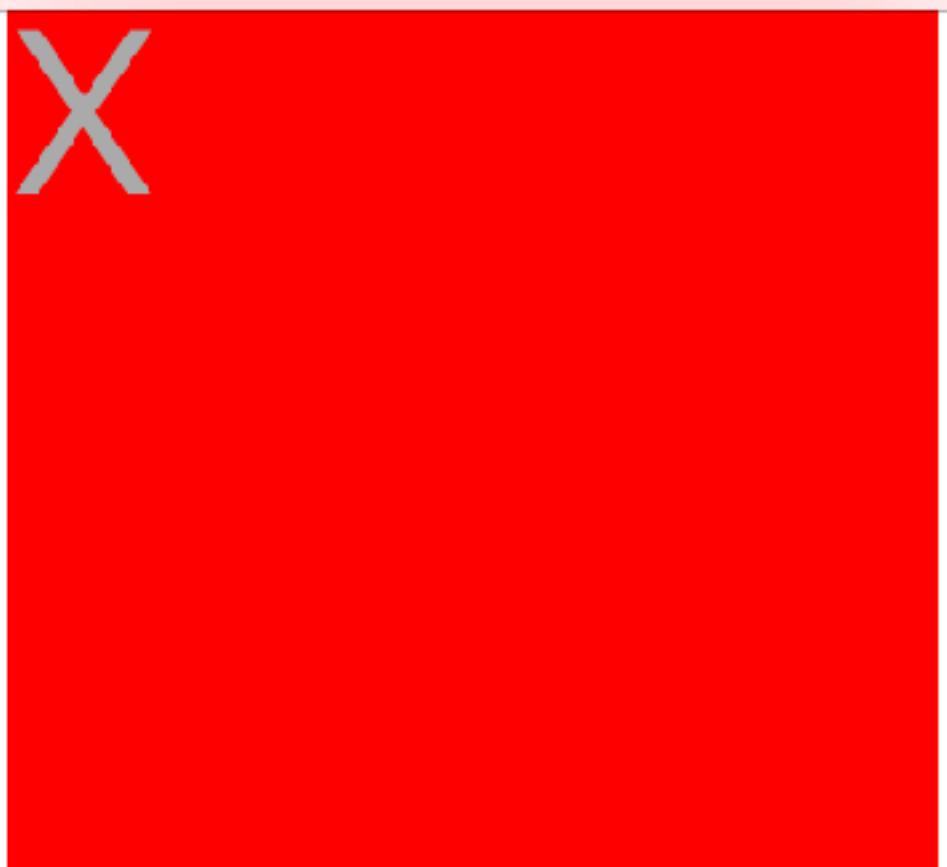
# CONNECTING VIEW CONTROLLERS

SUBTITLE

- But do they know about each other?
  - Not if you're practicing good MVC
- Possible ways to pass messages
  - NSNotifications
  - KVO
  - Delegation

Show Green

Add Container



# CONNECTING VIEW CONTROLLERS

- Create a protocol to allow communication between parent and child

child view controller

```
import UIKit

/// Protocol that allows the remove child view controller cycle to be called by the parent view controller
protocol ColorViewControllerDelegate: class {
    func removeFromContainerViewController(sender: ColorViewController)
}
```

```
/// View controller that can either dismiss itself or tell a parent view controller to remove it
class ColorViewController: UIViewController {
```

```
    /// Keep a reference to the parent view controller
    weak var delegate: ColorViewControllerDelegate?
```

```
    // MARK: - IBActions
    //
```

```
    @IBAction func tapCloseButton(sender: UIButton) {
        if parentViewController != nil {
            delegate?.removeFromContainerViewController(self)
        }
    }
```

```
}
```

```
// MARK: - Lite
//
```

# CONNECTING VIEW CONTROLLERS

- Tapping on the button in the child will remove it from the parent

```
protocol ColorViewControllerDelegate: class { ... }

/// View controller that can either dismiss itself or tell a parent view controller to remove it
///

class ColorViewController: UIViewController {

    /// Keep a reference to the parent view controller
    weak var delegate: ColorViewControllerDelegate?

    // MARK: - IBActions
    //

    @IBAction func tapCloseButton(sender: UIButton) {
        if parentViewController != nil {
            delegate?.removeFromContainerViewController(self)
        }
    }

    // MARK: - Life
    //

    override func viewDidLoad() { ... }

}
```

# CONNECTING VIEW CONTROLLERS

- More versatile implementation
- Determine if under containment or content view controller

```
/// protocol ColorViewControllerDelegate: class { ... }

/// View controller that can either dismiss itself or tell a parent view
/// controller to remove it
///

class ColorViewController: UIViewController {

    /// Keep a reference to the parent view controller
    weak var delegate: ColorViewControllerDelegate?

    //
    // MARK: - IBActions
    //

    @IBAction func tapCloseButton(sender: UIButton) {

        if parentViewController != nil {
            delegate?.removeFromContainerViewController(self)
        } else {
            presentingViewController?.dismissViewControllerAnimated(true, completion: nil)
        }
    }

    //
    // MARK: - Life
    //

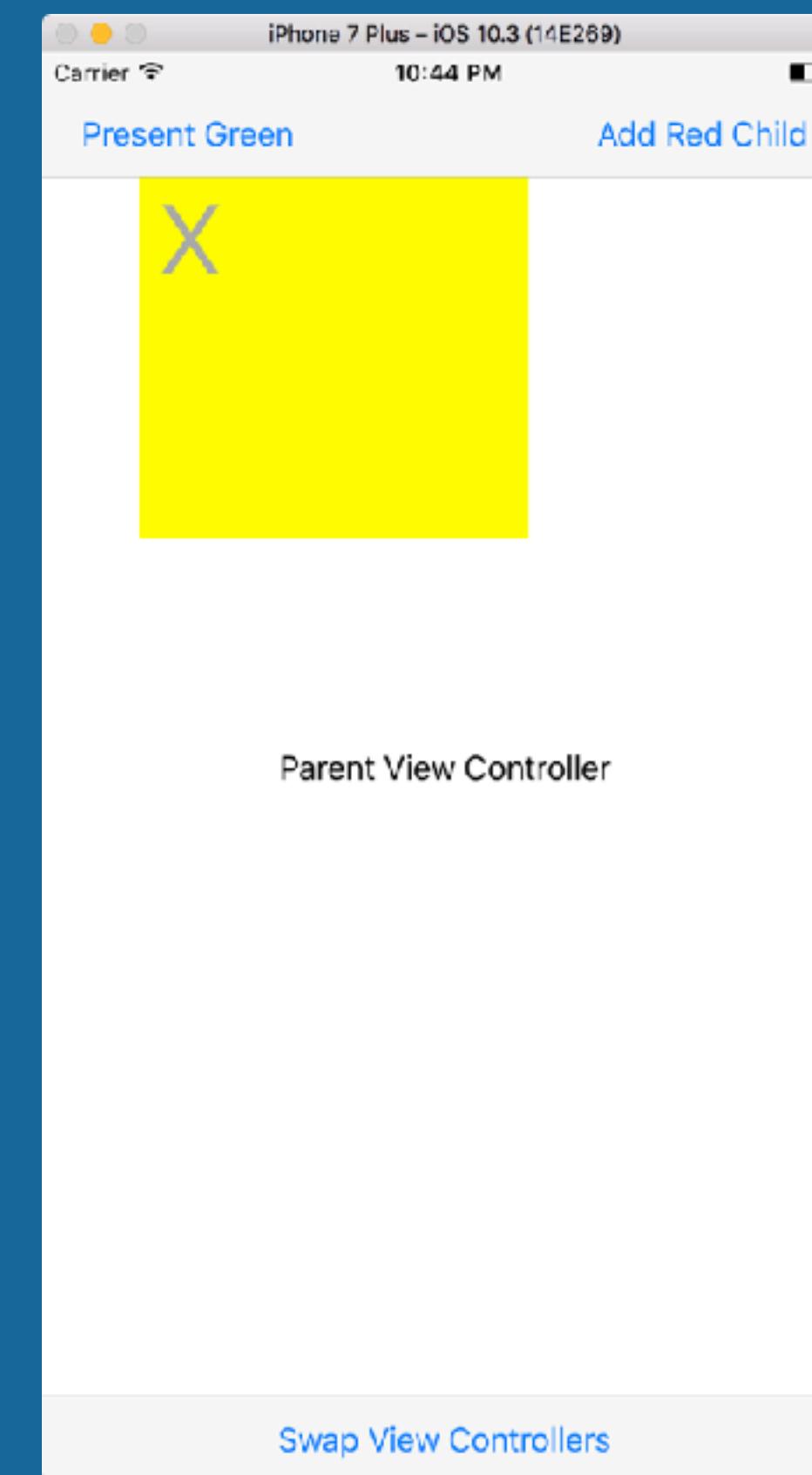
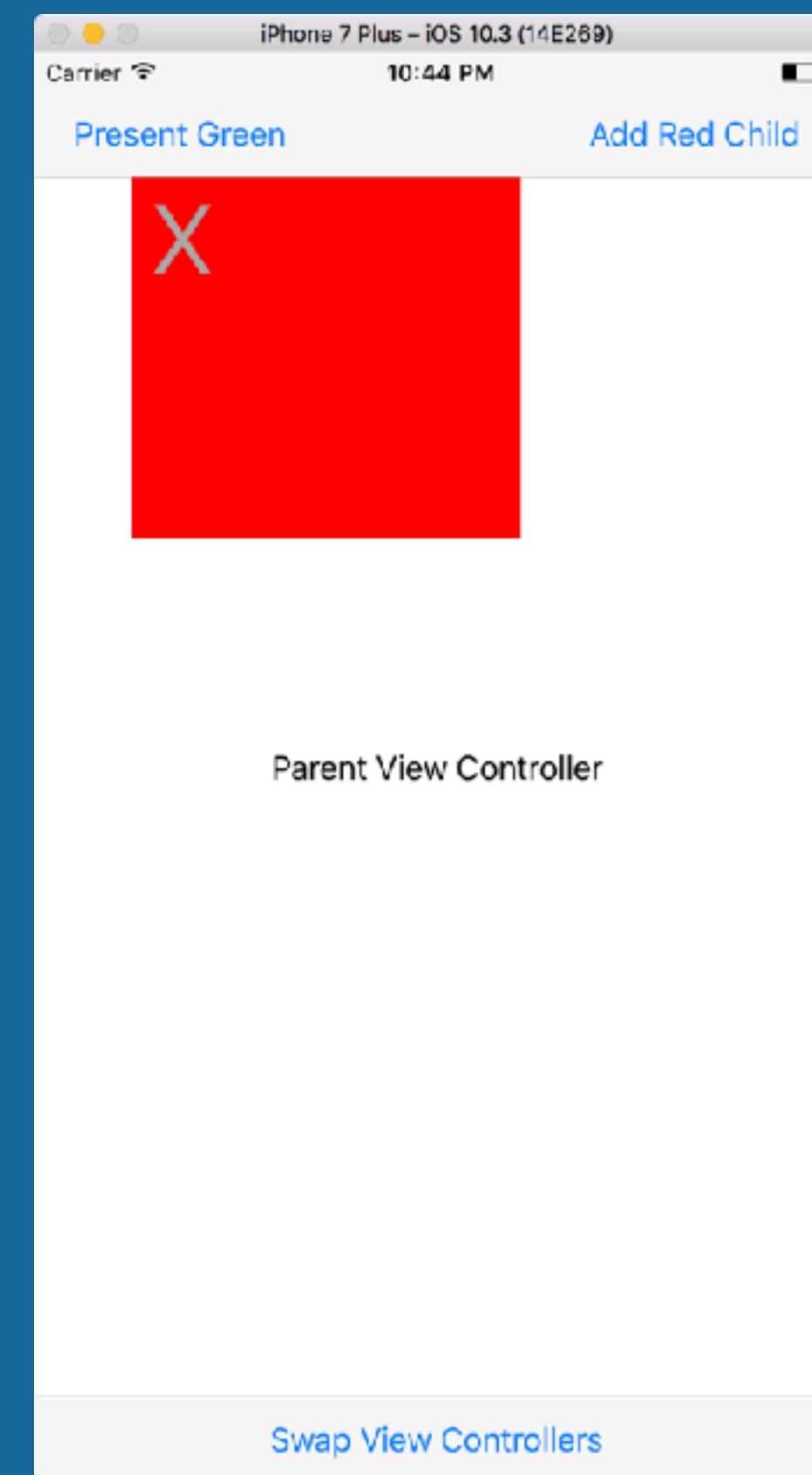
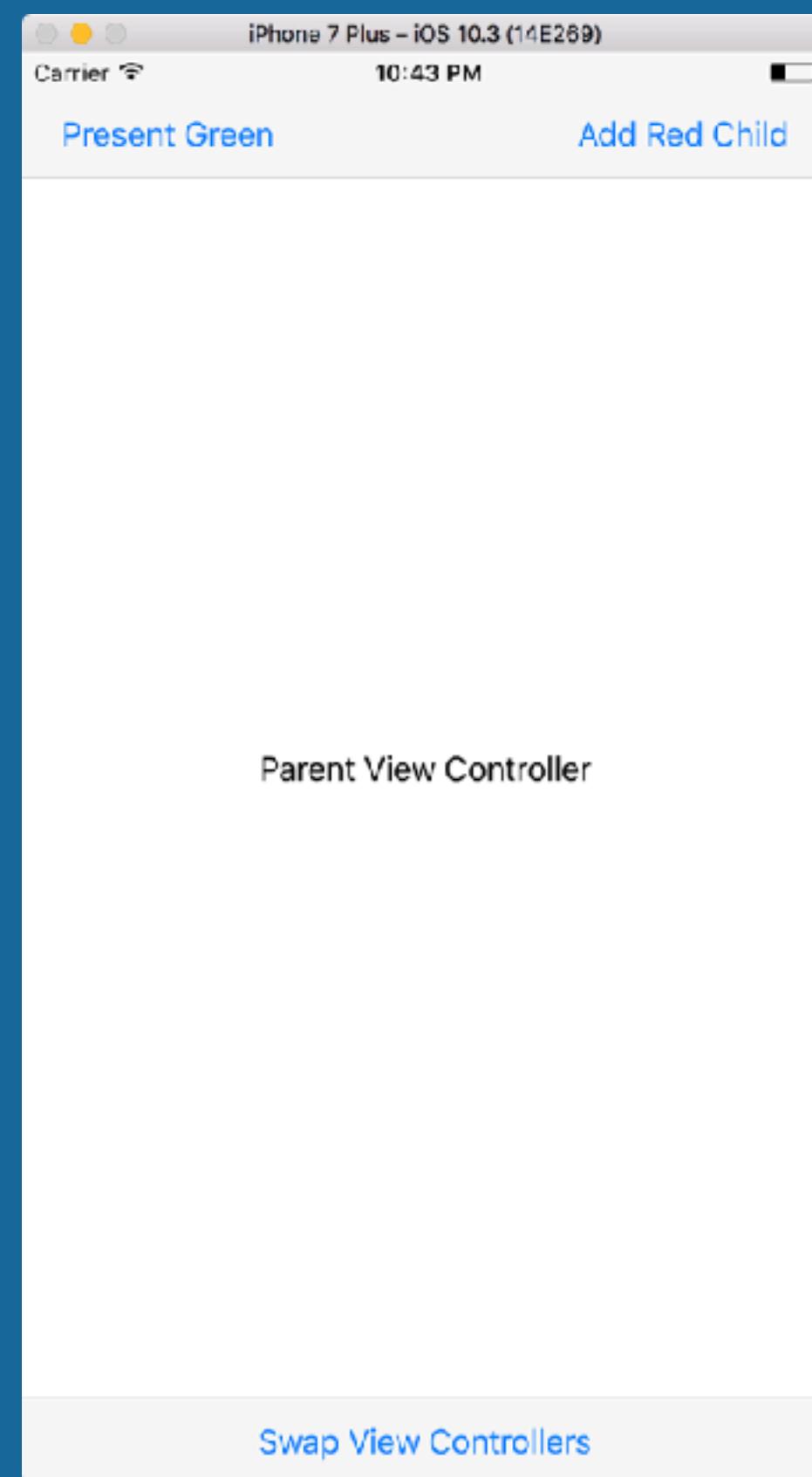
    override func viewDidLoad() { ... }

}
```

**CHILD VIEW**  
**CONTROLLER VIEW**  
**TRANSITIONS**

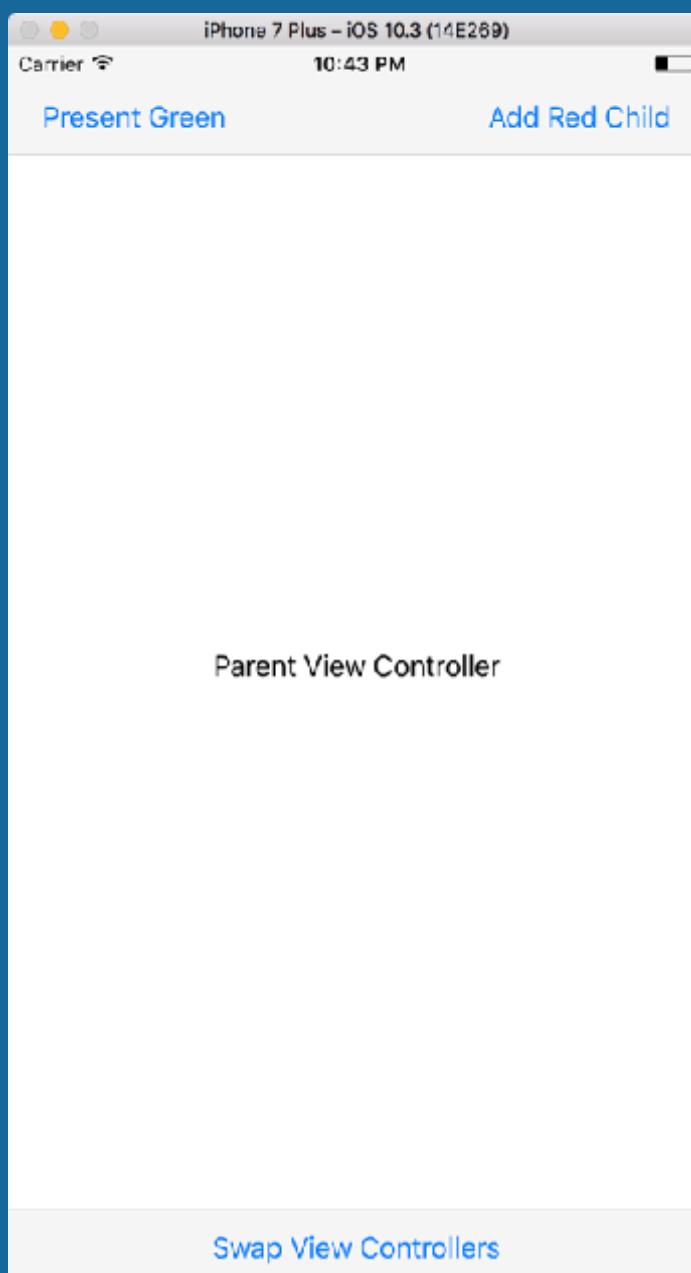
# VIEW CONTROLLER CONTAINERS

- Transitioning between children view controllers



# VIEW CONTROLLER CONTAINERS

- Transitioning between children view controllers

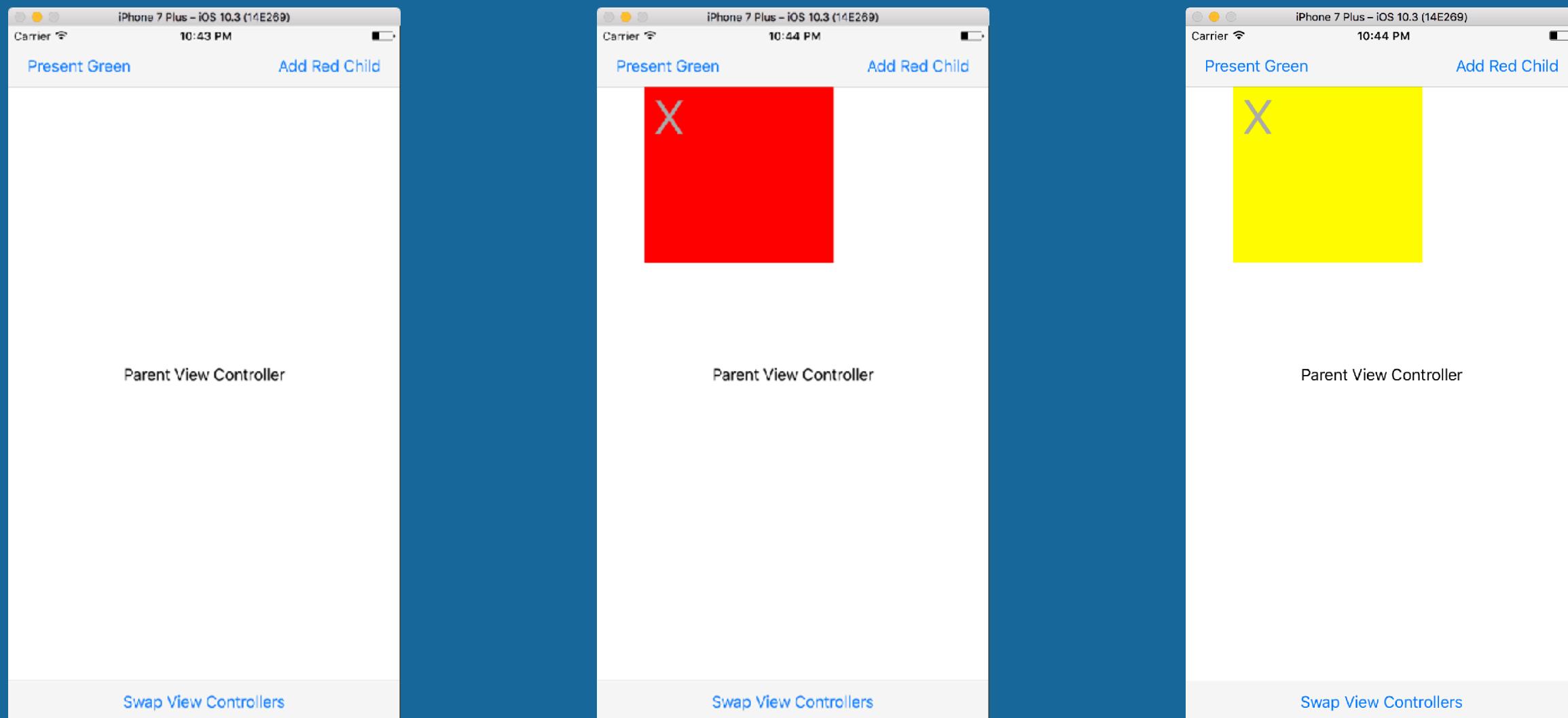


```
@IBAction func tapSwapViewControllers(_ sender: UIBarButtonItem) {  
    // Create a new view controller (yellow color)  
    let gvc = ColorViewController(nibName: "GreenViewController", bundle: nil)  
    gvc.delegate = self  
    gvc.view.backgroundColor = UIColor.yellow  
    gvc.view.frame = CGRect(x: 50, y: 50, width: 200, height: 200)  
    gvc.view.alpha = 0.0  
  
    // The view controllers have to have the same parent...but  
    // do not add it to the view hierarchy  
    addChildViewController(gvc)  
  
    // transition between them; fade out/ fade in  
    self.transition(from: gvc2!, to: gvc, duration: 2.0,  
                    options: UIViewAnimationOptions.curveEaseInOut,  
                    animations: {  
            self.gvc2?.view.alpha = 0.0  
            gvc.view.alpha = 1.0  
  
        }) { (animated) in  
            print("done")  
        }  
}
```

# VIEW CONTROLLER CONTAINERS

CONTAINER  
VIEW

- Make your own tab bar



BUTTONS

HOME.

USERS.

SETTINGS.

**VIEW CONTROLLER  
CONTAINMENT USING  
INTERFACE BUILDER**

# VIEW CONTROLLER CONTAINMENT USING INTERFACE BUILDER

SUBTITLE

- Add Container Views to a View Controller in Interface Builder
- Automatically performs the parent-child dance for you

The screenshot shows the Interface Builder library with several items listed:

- Tab Bar Item** - Represents an item on a UITabBar object.
- Search Bar** - Displays an editable search bar, containing the search icon, that sends an action message ...
- Search Bar and Search Display Controller** - Displays an editable search bar connected to a search dis...
- Fixed Space Bar Button Item** - Represents a fixed space item on a UIToolbar object.
- Flexible Space Bar Button Item** - Represents a flexible space item on a UIToolbar object.
- View** - Represents a rectangular region in which it draws and receives events.
- Container View** - Defines a region of a view controller that can include a child view controller.

At the bottom of the library, there are several small icons: a magnifying glass, a document, a square, a triangle, a circle, and a rectangle. To the right of these icons is a "Filter" button with a magnifying glass icon.

# VIEW CONTROLLER CONTAINMENT USING INTERFACE BUILDER

The screenshot shows the Xcode interface with the Project Navigator on the left and the Interface Builder canvas on the right.

**Project Navigator:**

- ViewControllerIB (marked M)
- ViewControllerIB (marked A)
- Main.storyboard (marked M)
- GreenViewController.swift (marked A)
- Assets.xcassets
- LaunchScreen.storyboard
- Info.plist
- Products

**Interface Builder (View Controller Scene):**

- View Controller
  - Top Layout Guide
  - Bottom Layout G...
  - View
    - Container View
    - Constraints
  - First Responder
  - Exit
  - Storyboard Entry Poi...
  - Embed segue to "Vie..."
- Green View Controller...
  - Green View Controller
    - View
      - I'm my own vi...
      - Constraints
    - First Responder
    - Exit

**Interface Builder Canvas:**

The canvas displays two view controllers. The first view controller, titled "View Controller", contains a blue "UIView" subview. The second view controller, titled "Green View Controller", contains a white "View" subview with the text "I'm my own view controller!".

A segue connection is visible between the two view controllers, indicating the containment relationship.

# VIEW CONTROLLER CONTAINMENT USING INTERFACE BUILDER

The screenshot illustrates the setup and execution of View Controller Containment in Xcode.

**Left Panel (File Navigator):**

- ViewController.swift
- Main.storyboard **M**
- GreenViewController.swift **A**
- Assets.xcassets
- LaunchScreen.storyboard
- Info.plist
- Products

**Center Panel (Interface Builder):**

The Main.storyboard file is open, showing the storyboard structure:

- Bottom Layout G...**: A container view with a child view and constraints.
- View**: A standard view.
- Container View**: A container view used for containment.
- Constraints**: Constraints for the views.
- First Responder**: Standard responder chain.
- Exit**: Standard exit node.
- Storyboard Entry Poi...**: Storyboard entry point.
- Embed segue to "Vie..."**: An embed segue connecting the main view to a child view.

**Green View Controller...**: A section containing:

- Green View Controller**: A green view controller instance.
- View**: A view within the green view controller.
- I'm my own vi...**: A label within the green view controller's view.
- Constraints**: Constraints for the green view controller's view.
- First Responder**: Standard responder chain.
- Exit**: Standard exit node.

**Right Panel (Simulator Preview):**

The iPhone 7 Plus - iOS 10.3 (14E269) simulator shows the result of the containment setup. The main view contains a green view controller, which displays the text "I'm my own view controller!".

**Bottom Bar:** Segmented Control - Displays

**SOME IMPORTANT  
THINGS TO REMEMBER  
ABOUT CONTAINMENT**

# WHAT YOU NEED TO KNOW ABOUT HIERARCHIES

- Container controllers are responsible for child/parent relationships
- There are consistent and inconsistent hierarchies
  - `UIViewControllerHierarchyInconsistencyException`
    - The view associated with a view controller was added directly to a view hierarchy
    - Pointing to view controller that are not in that view hierarchy
- Appearance callbacks are different than content view controllers

# WHAT YOU NEED TO KNOW ABOUT HIERARCHIES

- Appearance callbacks
  - `addChildViewController` does not trigger appearance callbacks (e.g. viewDid/WillAppear)
  - This view controller is a child (nothing to do with appearance)
- Appearance callbacks called when they move in/out of the window hierarchy
  - Not necessarily when you “see” it
  - Add a view “behind” a view and you do not see
  - Think of it as “Made it into the window’s view hierarchy”

# WHAT YOU NEED TO KNOW ABOUT HIERARCHIES

## APPEARANCE CALLBACKS

- `viewWillAppear:`
  - ■ Called before the view is added to the windows' view hierarchy
  - ■ Called before `view.layoutSubviews()` (if necessary)
- `viewDidAppear:`
  - Called after the view is added to the view hierarchy
  - Called after `view.layoutSubviews()` (if necessary)
- `viewWillDisappear:`
  - Called before the view is removed from the windows' view hierarchy
- `viewDidDisappear:`
  - Called after the view is removed from the windows' view hierarchy

# WHAT YOU NEED TO KNOW ABOUT HIERARCHIES

- When should you create a custom view controller?
  - Never, use Apple's built in containers
  - Design, you want custom aesthetic
  - Function, new application flow

# WHAT YOU NEED TO KNOW ABOUT HIERARCHIES

- Apple is updating container controllers
  - iOS5 UISplitViewController always shows master view controller
  - iOS7 Custom ViewController transitions (modal vc on iPhone)
  - iOS8 Custom presentation controllers
  - ...
  - iOS14 UIHamburger??
  - Not in SwiftUI (yet)

# WHAT YOU NEED TO KNOW ABOUT HIERARCHIES

- Practical suggestions for building a container view controller (from Apple)
  - Design as content view controller first
  - Never access any view other than the top-level view of the child view controller (i.e. don't expose unnecessary details to children)
  - The container should use protocols to declare methods and properties



# ADVANCED iOS APPLICATION DEVELOPMENT

---

MPCS 51032 • SPRING 2020 • SESSION 1A

# SESSION 1B



# ADVANCED iOS APPLICATION DEVELOPMENT

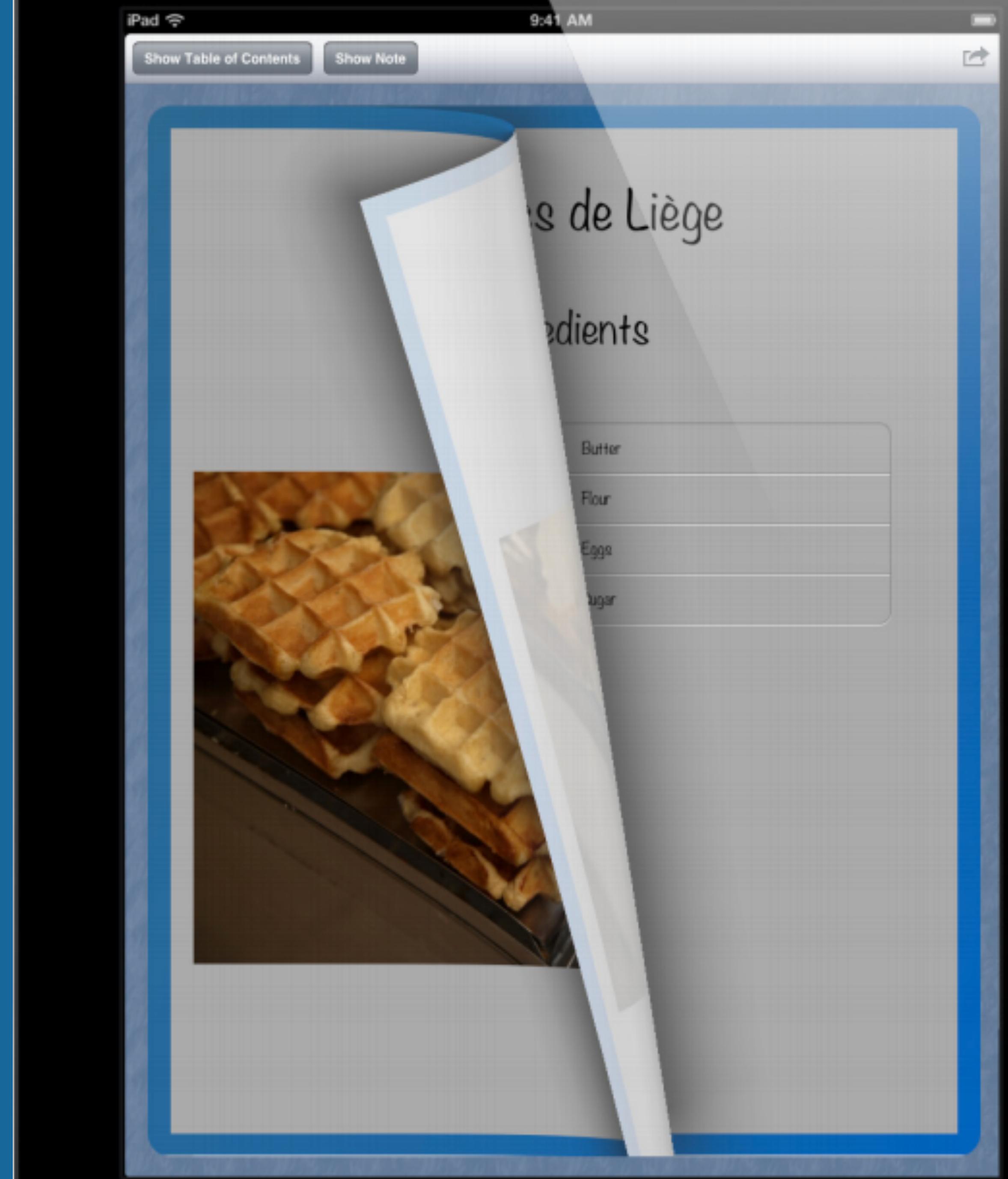
---

MPCS 51032 • SPRING 2020 • SESSION 1A

# PAGE VIEW CONTROLLER

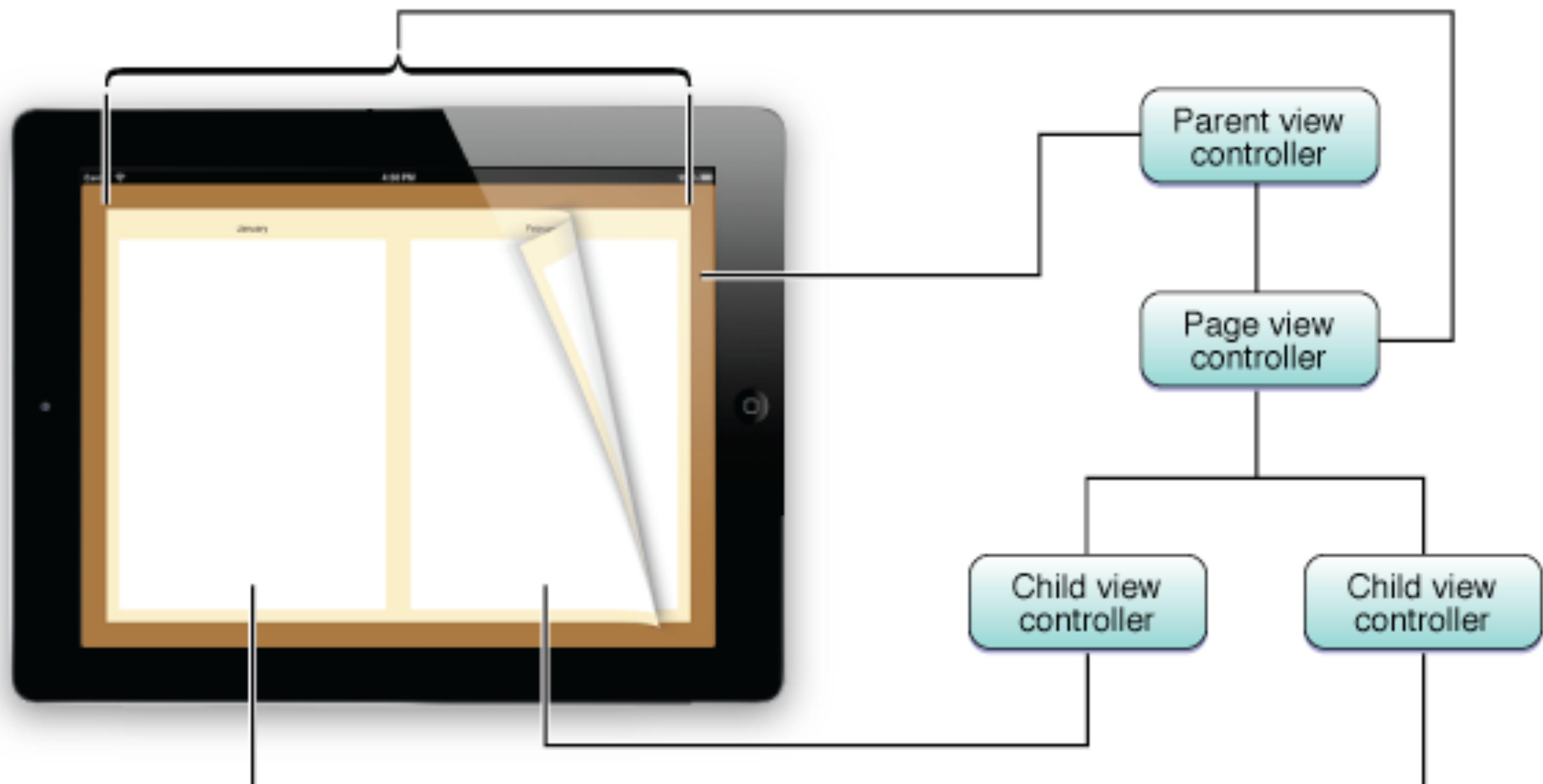
# PAGE VIEW CONTROLLER

- Container view controller released in iOS5
- Present content in a page-by-page manner
  - Navigate among views with a page curl transition
  - Manages a self-contained view hierarchy
  - Manages child view controllers that present the content



# PAGE VIEW CONTROLLER

- Parent view of this hierarchy is managed by the page view controller
- Child views are managed by the content view controllers that you provide



# PAGE VIEW CONTROLLER

- Navigation can be controlled
  - Programmatically by your app
  - Directly by the user using gestures
- When navigating from page to page uses the transition that you specify to animate the change



# PAGE VIEW CONTROLLER

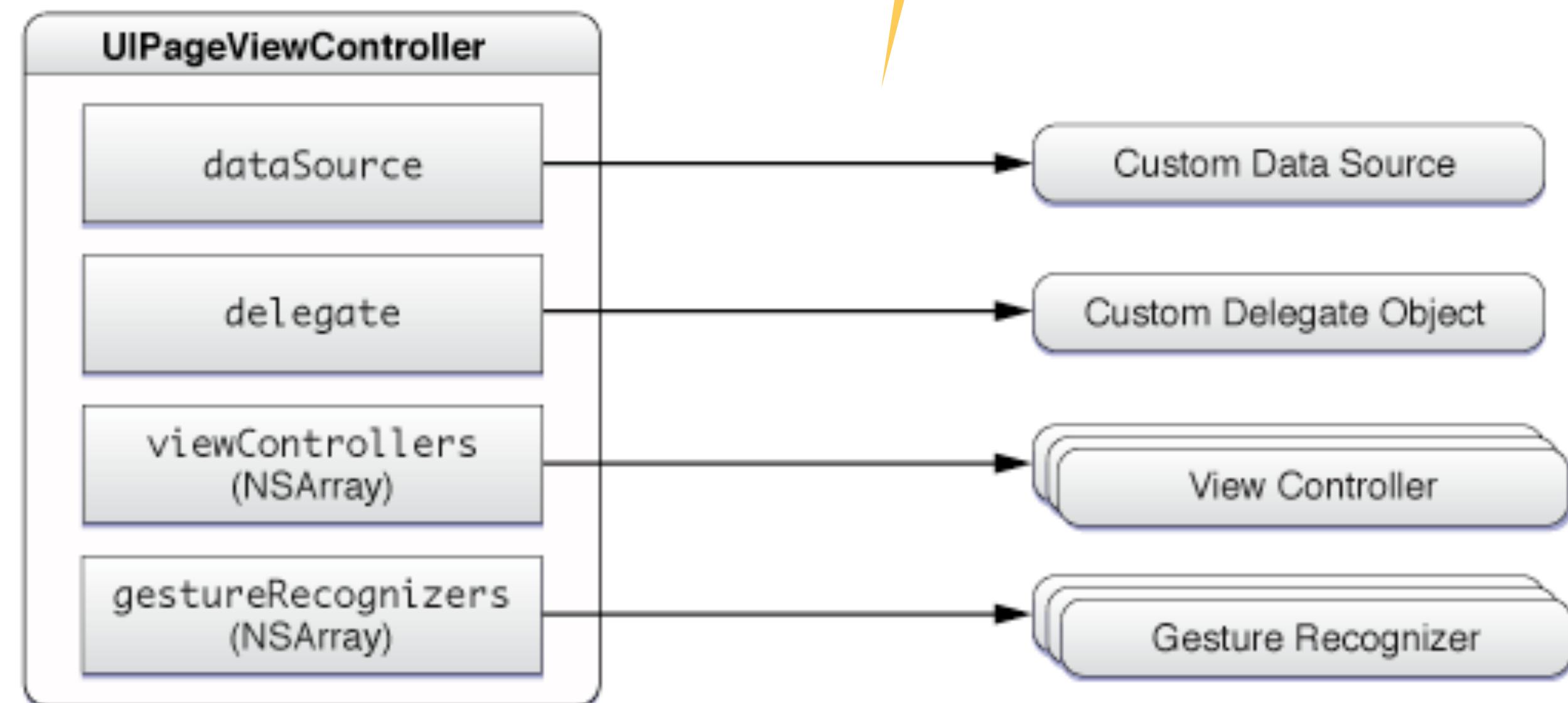
- Provide the content view controllers
  - One at a time
  - Two at a time
  - As-needed using a data source
- When providing content view controllers one at a time use the `setViewControllers:direction:animated:completion:` method to set the current content view controllers.
- To support gesture-based navigation, you must provide your view controllers using a data source object



# PAGE VIEW CONTROLLER

UIPAGEVIEWCONTROLLER API

- A page view interface consists of the following objects
  - An optional delegate
  - An optional data source
  - An array of the current view controllers
  - An array of gesture recognizers

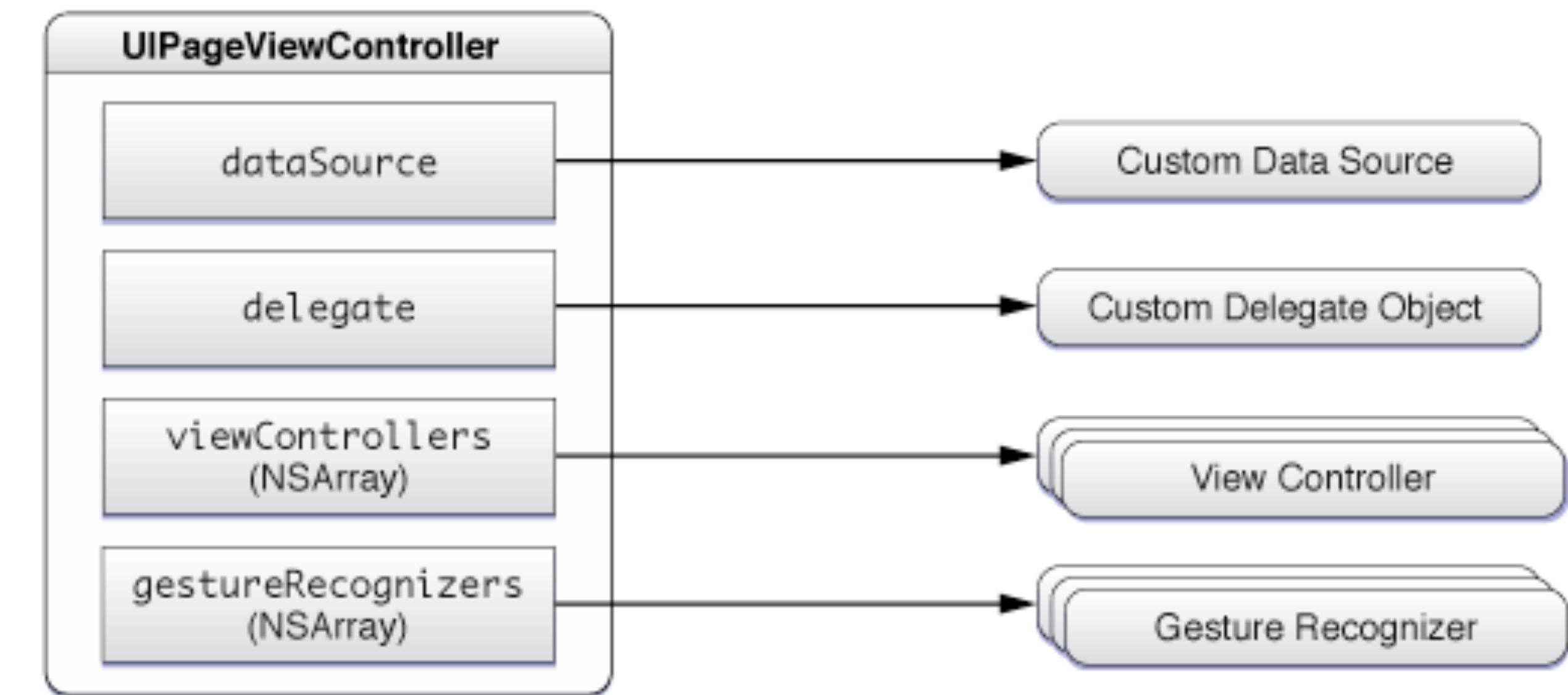


Just like a table

# PAGE VIEW CONTROLLER

## UIPAGEVIEWCONTROLLER API

- The data source is responsible for providing the content view controllers on demand
  - Must conform to the `UIPageViewControllerDataSource` protocol

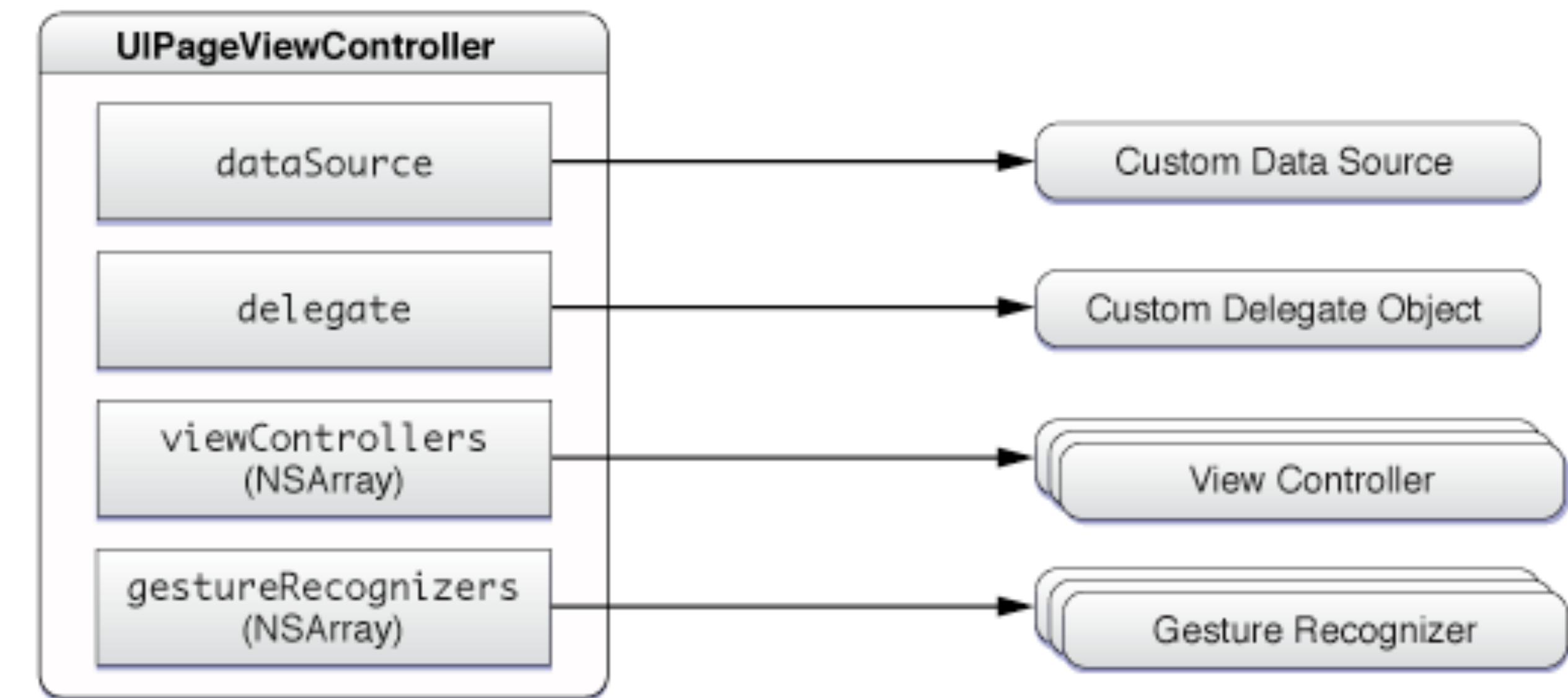


Reusing view controllers  
similar to table cells

# PAGE VIEW CONTROLLER

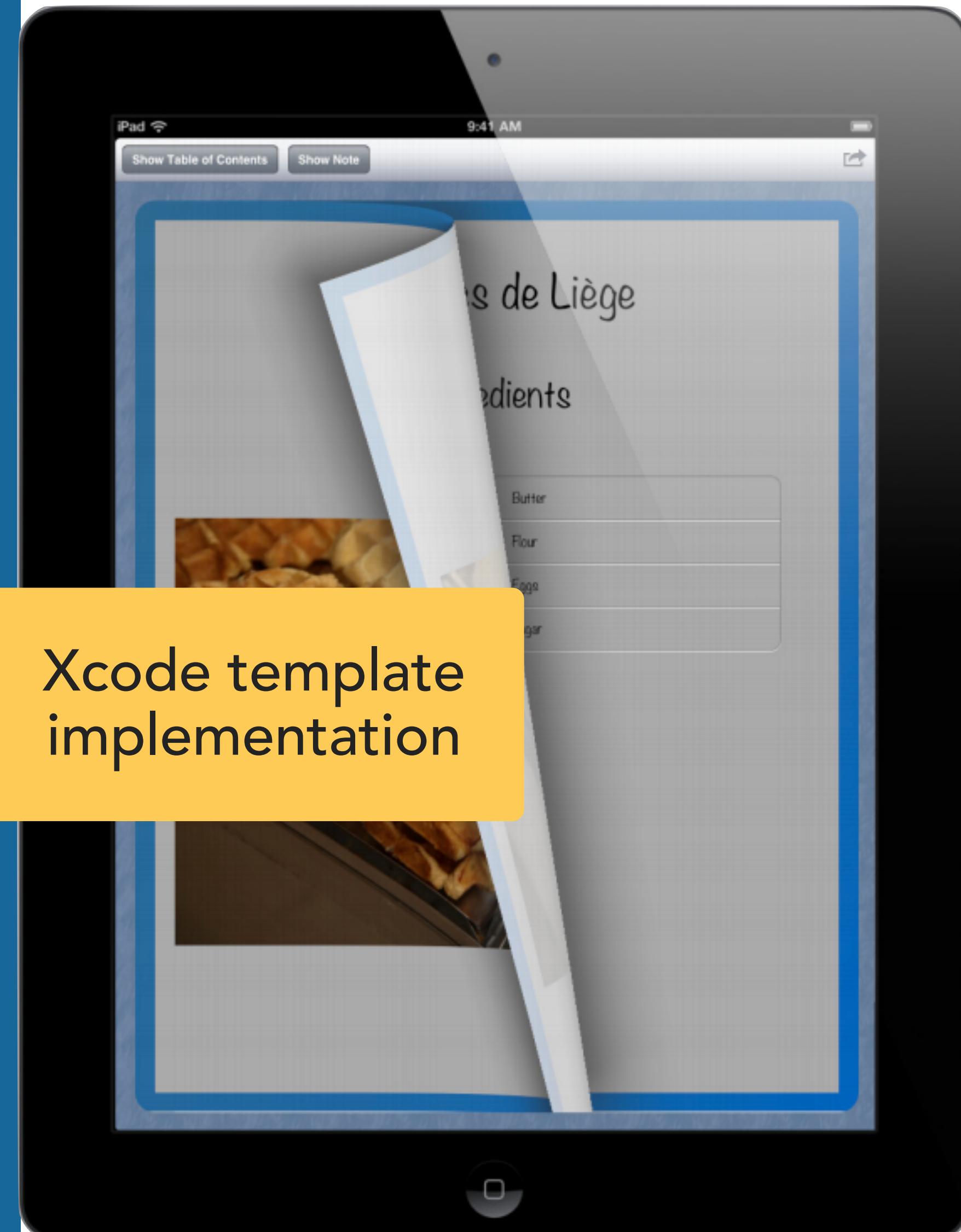
## UIPAGEVIEWCONTROLLER API

- The delegate object
  - Provides some appearance-related information
  - Provides methods called in response to orientation changes
  - Receives notifications about gesture-initiated transitions



# PAGE VIEW CONTROLLER

- Page view controller can be used in a wide variety of settings
- A page view controller's view can be resized and embedded in a view hierarchy
  - Unlike a navigation or tab bar controller
- Can be a window's root view controller



# **UIPAGEVIEWCONTROLLER**

# UIPAGEVIEWCONTROLLER

## INITIALIZATION

transition style

```
// Configure the page view controller and add it as a child view controller.  
self.pageViewController = UIPageViewController(transitionStyle: .PageCurl, navigationOrientation: .Horizontal, options: nil)  
self.pageViewController!.delegate = self
```

page flip direction

page spine

- Initialization

# UIPAGEVIEWCONTROLLER

## INITIALIZATION

```
let viewControllers = [startingViewController]
self.pageViewController!.setViewControllers(viewControllers, direction: .Forward, animated: false, completion: {done in })
```

array of view controllers

- Initial view controllers
  - Use 2 view controllers for mid-spine look

# UIPAGEVIEWCONTROLLER

## INITIALIZATION

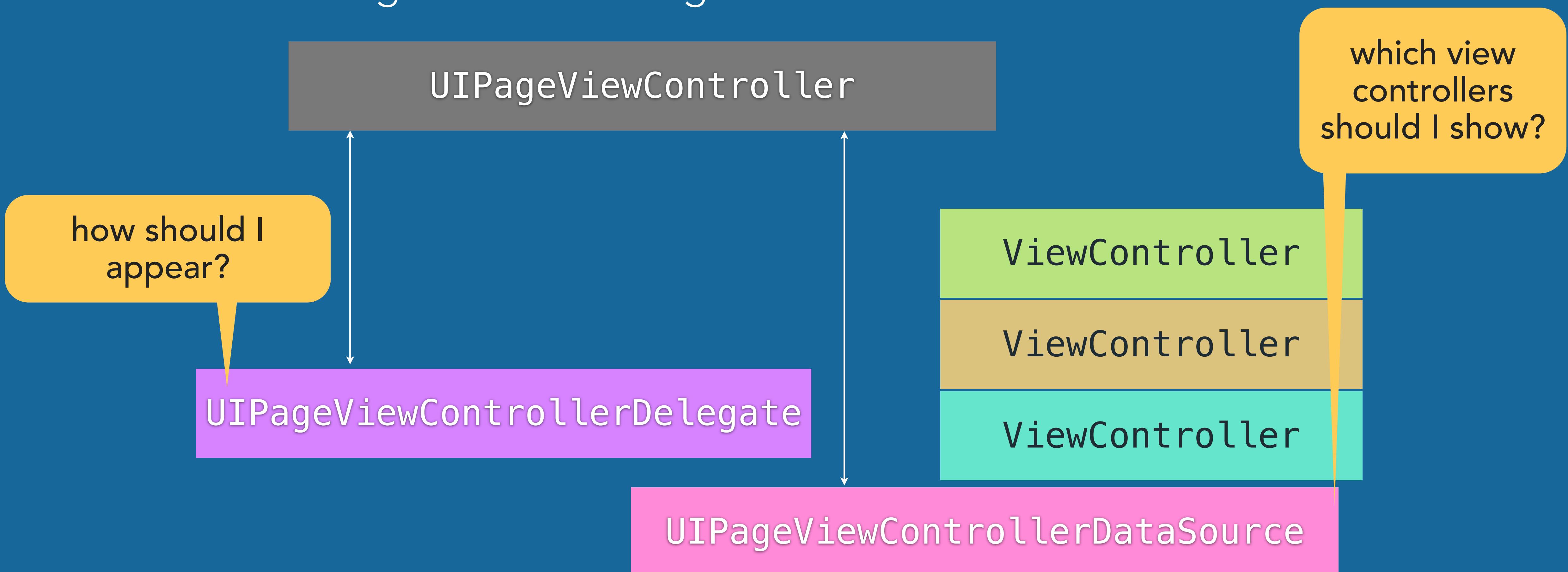
```
let viewControllers = [currentViewController]
self.pageViewController!.setViewControllers(viewControllers, direction: .Forward, animated: true, completion: {done in })
```

- Can also call `setViewControllers` programmatic navigation
  - Go directly to a different ViewController out of order

# UIPAGEVIEWCONTROLLER

## INITIALIZATION

- User-driven navigation uses delegate and data source



# **A SIMPLE PAGE VIEW CONTROLLER EXAMPLE**

**ONCE UPON A TIME...**

**PAGE VIEW**

**CONTROLLER**

**TEMPLATE**

# PAGE VIEW CONTROLLER TEMPLATE

iOS

Application

Framework & Library

Other

OS X

Application

Framework & Library

System Plug-in

Other

 Master-Detail Application

 Page-Based Application

 1 Single View Application

 Tabbed Application

 Game

**Page-Based Application**

This template provides a starting point for a page-based application that uses a page view controller to manage multiple pages of content.

**Page-Based**

Cancel Previous Next

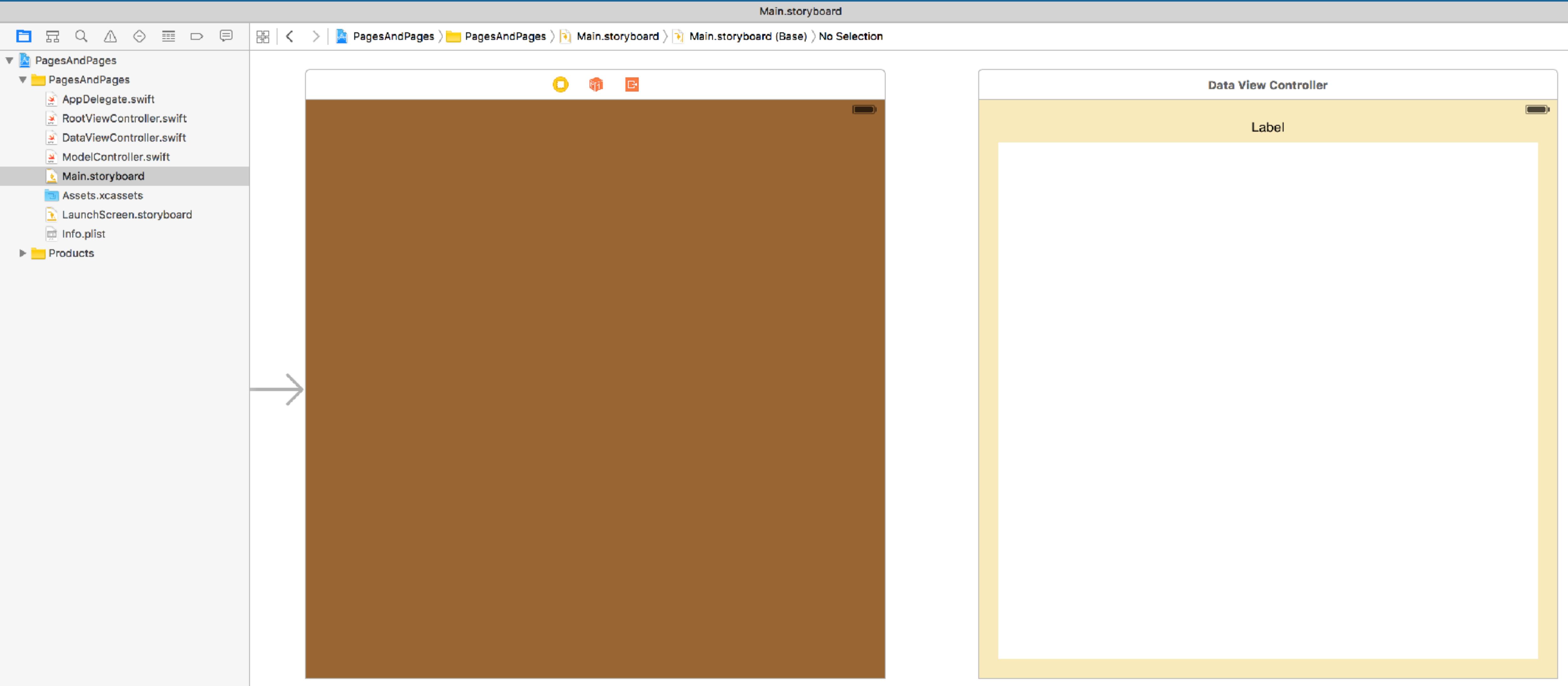
ction



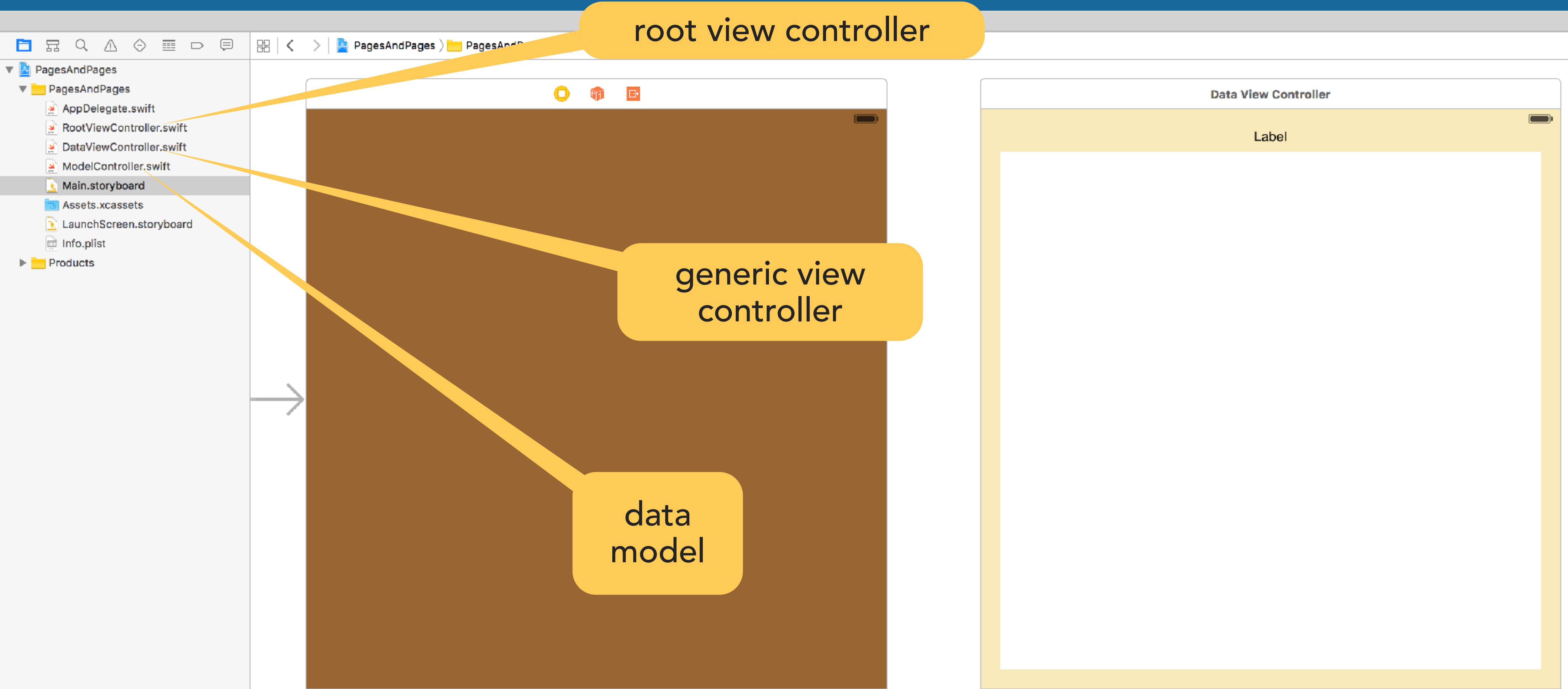
f - Define a block as

**as Variable** - Save  
able to allow reuse or  
argument.

# PAGE VIEW CONTROLLER TEMPLATE

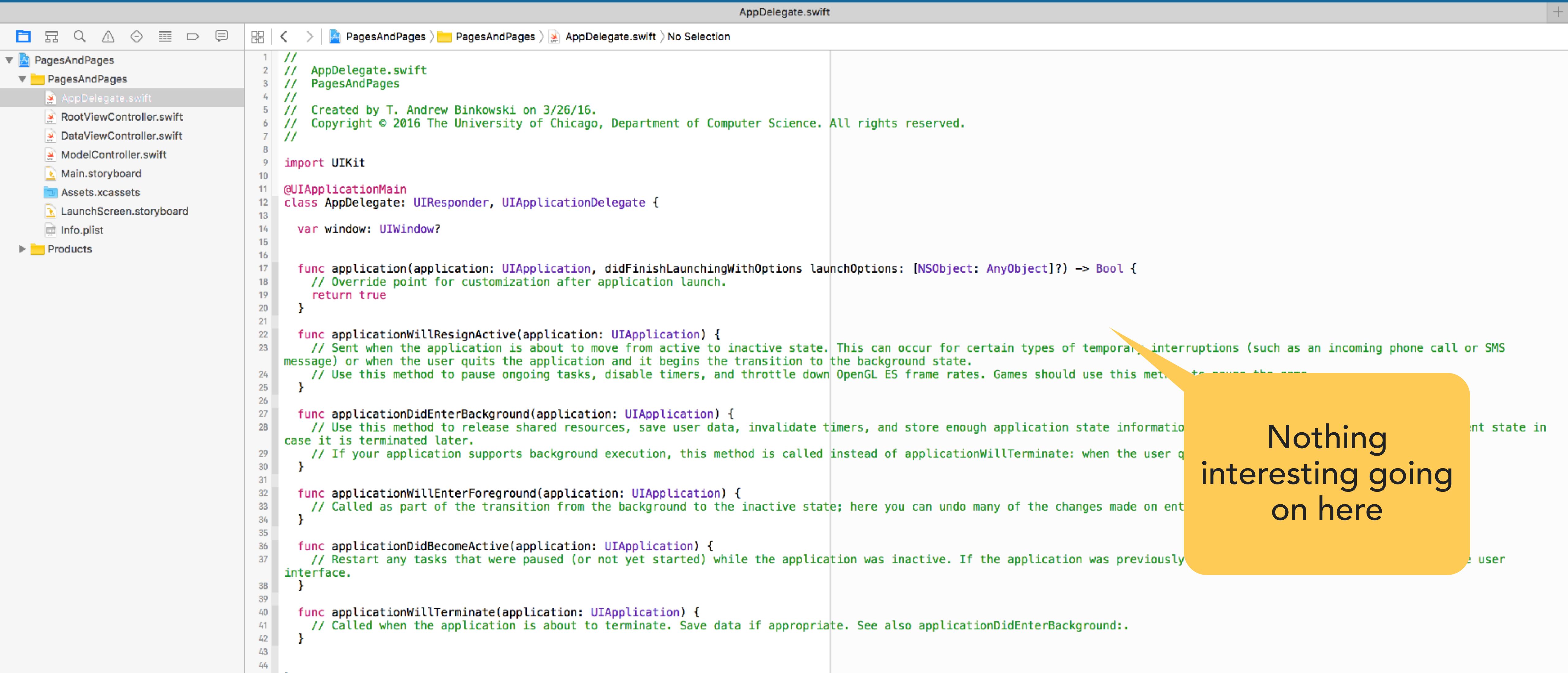


# PAGE VIEW CONTROLLER TEMPLATE



# PAGE VIEW CONTROLLER TEMPLATE

## ROOT VIEW CONTROLLER



The screenshot shows the Xcode interface with the file `AppDelegate.swift` open. The left sidebar shows the project structure with files like `RootViewController.swift`, `DataViewController.swift`, and `ModelController.swift`. The main editor area contains the following Swift code:

```
// AppDelegate.swift
// PagesAndPages
// Created by T. Andrew Binkowski on 3/26/16.
// Copyright © 2016 The University of Chicago, Department of Computer Science. All rights reserved.

import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // 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.
    }

    func applicationDidEnterBackground(application: UIApplication) {
        // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information
        // case it is terminated later.
        // If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.
    }

    func applicationWillEnterForeground(application: UIApplication) {
        // Called as part of the transition from the background to the inactive state; here you can undo many of the changes made on entering the background.
    }

    func applicationDidBecomeActive(application: UIApplication) {
        // Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously
        // in the background, during its transition to the active state, it calls instead of this method.
    }

    func applicationWillTerminate(application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:.
    }
}
```

A yellow callout bubble with the text "Nothing interesting going on here" points to the bottom of the code.

# PAGE VIEW CONTROLLER TEMPLATE

## ROOT VIEW CONTROLLER

The screenshot shows the Xcode interface with the file `RootViewController.swift` open. The code implements a `RootViewController` class that conforms to `UIViewController` and `UIPageViewControllerDelegate`. It initializes a `pageViewController` and sets its delegate to `self`. The `viewDidLoad` method configures the page view controller's data source and adds it as a child. A callout bubble points from the text "Useful for book style app" to the `UIPageViewControllerDelegate` documentation.

```
// RootViewController.swift
// PagesAndPages
// Created by T. Andrew Binkowski on 3/26/16.
// Copyright © 2016 The University of Chicago, Department of
// import UIKit
class RootViewController: UIViewController, UIPageViewControllerDelegate {
    var pageViewController: UIPageViewController?
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
        // Configure the page view controller and add it as a child.
        self.pageViewController = UIPageViewController(transitionStyle: .scroll)
        self.pageViewController!.delegate = self
        let startingViewController: DataViewController = self.modelController!.createViewController()
        let viewControllers = [startingViewController]
        self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: true, completion: nil)
        self.pageViewController!.dataSource = self.modelController!
        self.addChildViewController(self.pageViewController!)
        self.view.addSubview(self.pageViewController!.view)
        // Set the page view controller's bounds using an inset rectangle.
        var pageViewRect = self.view.bounds
        if UIDevice.currentDevice().userInterfaceIdiom == .Pad {
            pageViewRect = CGRectInset(pageViewRect, 40.0, 40.0)
        }
        self.pageViewController!.view.frame = pageViewRect
        self.pageViewController!.didMove(toParentViewController: self)
    }
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

**UIPageViewControllerDelegate**

Inherits from: None  
Conforms to: NSObject  
Framework: UIKit in iOS 5.0 and later. [More related](#)

The delegate of a page view controller must adopt the `UIPageViewControllerDelegate` protocol. This protocol defines methods that allow the delegate to receive a notification when the device orientation changes and when the user navigates to a new page. For page-curl style transitions, the delegate can provide a different spine location in response to a change in the interface orientation.

**Responding to Page View Controller Events**

- `pageViewController:willTransitionToViewControllers:`

Called before a gesture-driven transition begins.

**Declaration**

**SWIFT**

```
optional func pageViewController(_ pageViewController: UIPageViewController, willTransitionToViewControllers pendingViewControllers: [UIViewController])
```

**OBJECTIVE-C**

```
-(void)pageViewController:(UIPageViewController *)pageViewController willTransitionToViewControllers:(NSArray<UIViewController *> *)pendingViewControllers
```

[Feedback](#)

**Page view controller Delegate**

**Useful for book style app**

# PAGE VIEW CONTROLLER TEMPLATE

## ROOT VIEW CONTROLLER

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
    // Configure the page view controller and add it as a child view controller.
    self.pageViewController = UIPageViewController(transitionStyle: .pageCurl, navigationOrientation: .horizontal, options: nil)
    self.pageViewController!.delegate = self

    let startingViewController: DataViewController = self.modelController.viewControllerAtIndex(0, storyboard: self.storyboard)!)
    let viewControllers = [startingViewController]
    self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: false, completion: {done in })

    self.pageViewController!.dataSource = self.modelController

    self.addChildViewController(self.pageViewController!)
    self.view.addSubview(self.pageViewController!.view)

    // Set the page view controller's bounds using an inset rect so that self's view is visible around the edges of the pages.
    var pageViewRect = self.view.bounds
    if UIDevice.current.userInterfaceIdiom == .pad {
        pageViewRect = pageViewRect.insetBy(dx: 40.0, dy: 40.0)
    }
    self.pageViewController!.view.frame = pageViewRect

    self.pageViewController!.didMove(toParentViewController: self)
}
```

Initialize the page view controller

# PAGE VIEW CONTROLLER TEMPLATE

## ROOT VIEW CONTROLLER

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
    // Configure the page view controller and add it as a child view controller.
    self.pageViewController = UIPageViewController(transitionStyle: .pageCurl, navigationOrientation: .horizontal, options: nil)
    self.pageViewController!.delegate = self

    let startingViewController: DataViewController = self.modelController.viewControllerAtIndex(0, storyboard: self.storyboard)!)
    let viewControllers = [startingViewController]
    self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: false, completion: {done in })

    self.pageViewController!.dataSource = self.modelController

    self.addChildViewController(self.pageViewController!)
    self.view.addSubview(self.pageViewController!.view)

    // Set the page view controller's bounds using an inset rect so that self's view is visible around the edges of the pages.
    var pageViewRect = self.view.bounds
    if UIDevice.current.userInterfaceIdiom == .pad {
        pageViewRect = pageViewRect.insetBy(dx: 40.0, dy: 40.0)
    }
    self.pageViewController!.view.frame = pageViewRect

    self.pageViewController!.didMove(toParentViewController: self)
}
```

set the initial view controller

# PAGE VIEW CONTROLLER TEMPLATE

## ROOT VIEW CONTROLLER

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
    // Configure the page view controller and add it as a child view controller.
    self.pageViewController = UIPageViewController(transitionStyle: .pageCurl, navigationOrientation: .horizontal, options: nil)
    self.pageViewController!.delegate = self

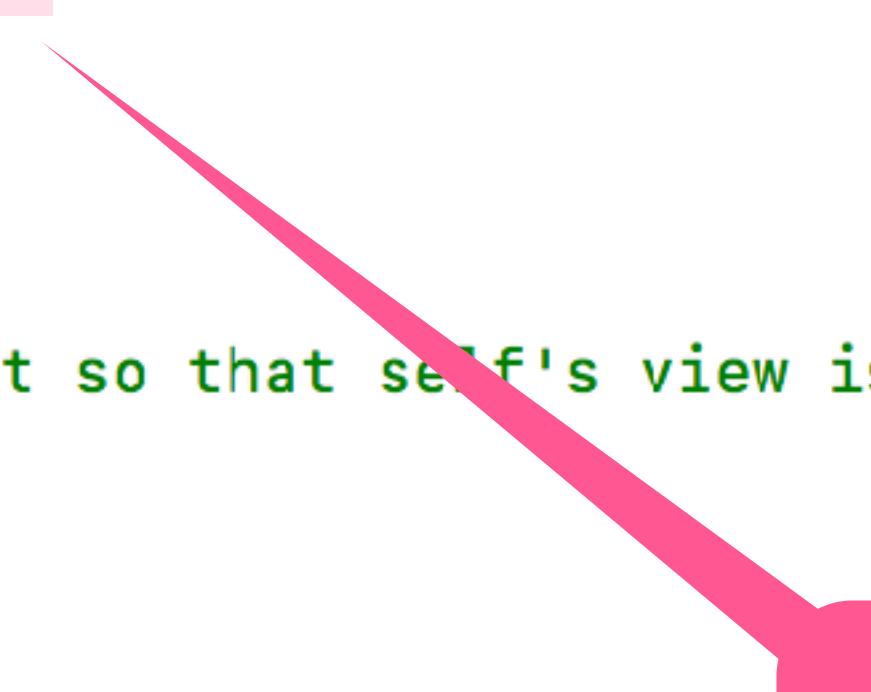
    let startingViewController: DataViewController = self.modelController.viewControllerAtIndex(0, storyboard: self.storyboard)!)
    let viewControllers = [startingViewController]
    self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: false, completion: {done in })

    self.pageViewController!.dataSource = self.modelController

    self.addChildViewController(self.pageViewController!)
    self.view.addSubview(self.pageViewController!.view)

    // Set the page view controller's bounds using an inset rect so that self's view is visible around the edges of the pages.
    var pageViewRect = self.view.bounds
    if UIDevice.current.userInterfaceIdiom == .pad {
        pageViewRect = pageViewRect.insetBy(dx: 40.0, dy: 40.0)
    }
    self.pageViewController!.view.frame = pageViewRect

    self.pageViewController!.didMove(toParentViewController: self)
}
```



data source in  
modelController

# PAGE VIEW CONTROLLER TEMPLATE

## ROOT VIEW CONTROLLER

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
    // Configure the page view controller and add it as a child view controller.
    self.pageViewController = UIPageViewController(transitionStyle: .pageCurl, navigationOrientation: .horizontal, options: nil)
    self.pageViewController!.delegate = self

    let startingViewController: DataViewController = self.modelController.viewControllerAtIndex(0, storyboard: self.storyboard)!
    let viewControllers = [startingViewController]
    self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: false, completion: {done in })

    self.pageViewController!.dataSource = self.modelController

    self.addChildViewController(self.pageViewController!)
    self.view.addSubview(self.pageViewController!.view)

    // Set the page view controller's bounds using an inset rect so that self's view is visible around the page view controller
    var pageViewRect = self.view.bounds
    if UIDevice.current.userInterfaceIdiom == .pad {
        pageViewRect = pageViewRect.insetBy(dx: 40.0, dy: 40.0)
    }
    self.pageViewController!.view.frame = pageViewRect

    self.pageViewController!.didMove(toParentViewController: self)
}
```

view controller  
containment

# PAGE VIEW CONTROLLER TEMPLATE

## ROOT VIEW CONTROLLER

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
    // Configure the page view controller and add it as a child view controller
    self.pageViewController = UIPageViewController(transitionStyle: .pageCurl,
                                                initialPage: 0,
                                                options: nil)
    self.pageViewController!.delegate = self

    let startingViewController: DataViewController = self.modelController.viewControllerAtIndex(0, storyboard: self.storyboard)!)
    let viewControllers = [startingViewController]
    self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: false, completion: {done in })

    self.pageViewController!.dataSource = self.modelController

    self.addChildViewController(self.pageViewController!)
    self.view.addSubview(self.pageViewController!.view)

    // Set the page view controller's bounds using an inset rect so that self's view is visible around the pages
    var pageViewRect = self.view.bounds
    if UIDevice.current.userInterfaceIdiom == .pad {
        pageViewRect = pageViewRect.insetBy(dx: 40.0, dy: 40.0)
    }
    self.pageViewController!.view.frame = pageViewRect

    self.pageViewController!.didMove(toParentViewController: self)
}
```

This is just one way of implementing a page view controller. Provides "chrome" around the pages.

view controller containment

# PAGE VIEW CONTROLLER TEMPLATE

## ROOT VIEW CONTROLLER

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
    // Configure the page view controller and add it as a child view controller.
    self.pageViewController = UIPageViewController(transitionStyle: .pageCurl, navigationOrientation: .horizontal)
    self.pageViewController!.delegate = self

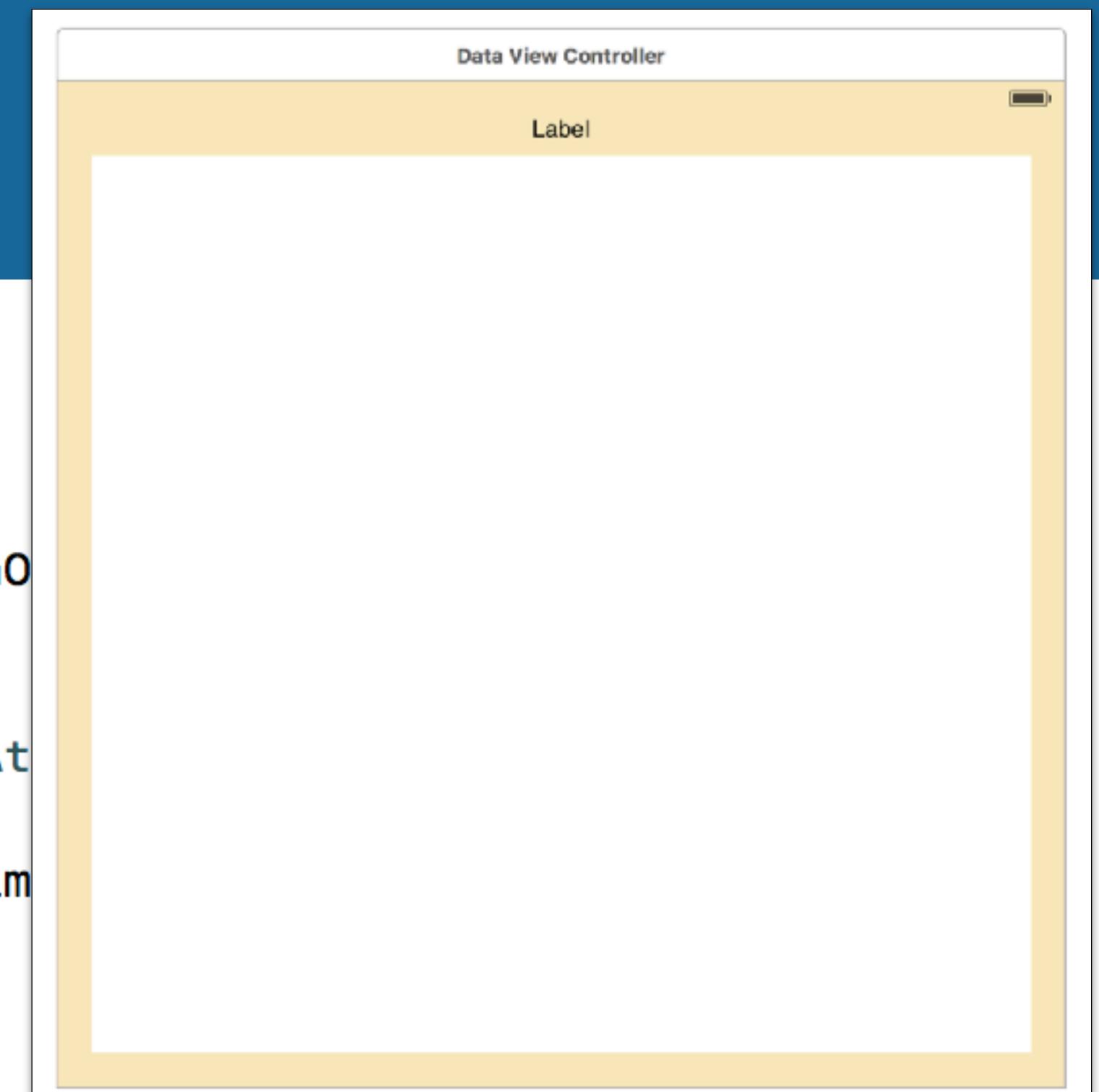
    let startingViewController: DataViewController = DataViewController()
    let viewControllers = [startingViewController]
    self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: true, completion: nil)

    self.pageViewController!.dataSource = self

    self.addChildViewController(self.pageViewController!)
    self.view.addSubview(self.pageViewController!.view)

    // Set the page view controller's bounds using an inset rect so that self's view is visible around the edges of the pages.
    var pageViewRect = self.view.bounds
    if UIDevice.current.userInterfaceIdiom == .pad {
        pageViewRect = pageViewRect.insetBy(dx: 40.0, dy: 40.0)
    }
    self.pageViewController!.view.frame = pageViewRect

    self.pageViewController!.didMove(toParentViewController: self)
}
```



# PAGE VIEW CONTROLLER TEMPLATE

## ROOT VIEW CONTROLLER

show only one page (portrait)

```
func pageViewController(pageViewController: UIPageViewController, spineLocationForInterfaceOrientation orientation: UIInterfaceOrientation) -> UIPageViewController {  
    if (orientation == .Portrait) || (orientation == .PortraitUpsideDown) || (UIDevice.currentDevice().userInterfaceOrientation == .Phone) {  
        // In portrait orientation or on iPhone: Set the spine position to "min"  
        // and the page view controller's view controllers array to contain just  
        // one view controller. Setting the spine position to  
        // 'UIPageViewControllerSpineLocationMid' in landscape orientation sets  
        // the doubleSided property to true, so set it to false here.  
        let viewController = self.pageViewController!.viewControllers![0]  
        let viewControllers = [viewController]  
        self.pageViewController!.setViewControllers(viewControllers, direction: .Forward, animated: true, completion: {done in })  
  
        self.pageViewController!.doubleSided = false  
        return .Min  
    }  
  
    // In landscape orientation: Set the spine location to "mid" and the  
    // page view controller's view controllers array to contain two view  
    // controllers. If the current page is even, set it to contain the current  
    // and next view controllers; if it is odd, set the array to contain the  
    // previous and current view controllers.  
    let viewController = self.pageViewController!.viewControllers![0] as! DataViewController  
    var viewControllers: [UIViewController]  
  
    let indexOfCurrentViewController = self.modelController.indexOfViewController(currentViewController)  
    if (indexOfCurrentViewController == 0) || (indexOfCurrentViewController % 2 == 0) {  
        let nextViewController = self.modelController.pageViewController(self.pageViewController!, viewControllerAfterViewController: currentViewController)  
        viewControllers = [currentViewController, nextViewController]  
    } else {  
        let previousViewController = self.modelController.pageViewController(self.pageViewController!, viewControllerBeforeViewController:  
        viewControllers = [previousViewController!, currentViewController]  
    }  
    self.pageViewController!.setViewControllers(viewControllers, direction: .Forward, animated: true, completion: {done in })  
  
    return .Mid
```

custom methods  
to get view  
controller array

show two  
pages



# PAGE VIEW CONTROLLER TEMPLATE

## MODEL CONTROLLER

```
import UIKit

/*
A controller object that manages a simple model — a collection of month names.

The controller serves as the data source for the page view controller; it therefore implements pageViewController:viewControllerBeforeViewController: and pageViewController:viewControllerAfterViewController:. It also implements a custom method, viewControllerAtIndex:, which is useful in the implementation of the delegate methods. Note that the initial configuration of the page view controller is done in the storyboard, so there is no need to actually create view controllers for each page in advance -- indeed doing so incurs unnecessary overhead. Instead, the controller is created on demand.
*/
class ModelController: NSObject, UIPageViewControllerDataSource {

    var pageData: [String] = []

    override init() { ... }

    func viewControllerAtIndex(index: Int, storyboard: UIStoryboard) -> DataViewController? { ... }

    func indexOfViewController(viewController: DataViewController) -> Int { ... }

    // MARK: - Page View Controller Data Source

    func pageViewController(pageViewController: UIPageViewController, viewControllerBeforeViewController viewController: UIViewController) -> UIViewController? { ... }

    func pageViewController(pageViewController: UIPageViewController, viewControllerAfterViewController viewController: UIViewController) -> UIViewController? { ... }

}
```

View controller  
for a data model  
at index

determine the  
index of a view  
controller

Data source methods

# PAGE VIEW CONTROLLER TEMPLATE

## MODEL CONTROLLER

```
class ModelController: NSObject, UIPageViewControllerDataSource {  
  
    var pageData: [String] = []  
  
    override init() {  
        super.init()  
        // Create the data model.  
        let dateFormatter = DateFormatter()  
        pageData = dateFormatter.monthSymbols  
    }  
  
    func viewControllerAtIndex(index: Int, storyboard: UIStoryboard) -> DataViewController? {  
        // Return the data view controller for the given index.  
        if (self.pageData.count == 0) || (index >= self.pageData.count) {  
            return nil  
        }  
  
        // Create a new view controller and pass suitable data.  
        let dataViewController = storyboard.instantiateViewControllerWithIdentifier("DataViewController") as! DataViewController  
        dataViewController.dataObject = self.pageData[index]  
        return dataViewController  
    }  
  
    func indexOfViewController(viewController: DataViewController) -> Int {  
        // Return the index of the given data view controller.  
        // For simplicity, this implementation uses a static array of model objects and the view controller stores the model object; you  
        return pageData.indexOf(viewController.dataObject) ?? NSNotFound  
    }  
  
    // MARK: - Page View Controller Data Source  
  
    func pageViewController(pageViewController: UIPageViewController, viewControllerBeforeViewController viewController: UIViewController) -> UIViewController? { ... }  
    func pageViewController(pageViewController: UIPageViewController, viewControllerAfterViewController viewController: UIViewController) -> UIViewController? { ... }  
}
```

View controller  
for a data model  
at index

determine the  
index of a view  
controller

# PAGE VIEW CONTROLLER TEMPLATE

## MODEL CONTROLLER

View controller  
for a data model  
at index

```
// MARK: - Page View Controller Data Source
```

```
func pageViewController(_ pageViewController: UIPageViewController, viewControllerBefore viewController: UIViewController) -> UIViewController? {
    var index = self.indexOfViewController(viewController as! DataViewController)
    if (index == 0) || (index == NSNotFound) {
        return nil
    }

    index -= 1
    return self.viewControllerAtIndex(index, storyboard: viewController.storyboard!)
}
```

```
func pageViewController(_ pageViewController: UIPageViewController, viewControllerAfter viewController: UIViewController) -> UIViewController? {
    var index = self.indexOfViewController(viewController as! DataViewController)
    if index == NSNotFound {
        return nil
    }

    index += 1
    if index == self.pageData.count {
        return nil
    }
    return self.viewControllerAtIndex(index, storyboard: viewController.storyboard!)
}
```

determine  
the index of a  
view  
controller

# PAGE VIEW CONTROLLER TEMPLATE

## MODEL CONTROLLER

```
demand.  
*/  
  
class ModelController: NSObject, UIPageViewControllerDataSource {  
  
    var pageData: [String] = []  
  
    override init() {  
        super.init()  
    }  
  
    func viewControllerAtIndex(index: Int, storyboard: UIStoryboard) -> UIViewController? {  
        let viewController = storyboard.instantiateViewControllerWithIdentifier("DataViewController")  
        return viewController  
    }  
  
    func indexOfViewController(viewController: DataViewController) -> Int {  
        let index = self.pageData.indexOf(viewController.pageData)  
        if index == NSNotFound {  
            return -1  
        }  
        return index  
    }  
  
    // MARK: - Page View Controller Data Source  
  
    func pageViewController(pageViewController: UIPageViewController, viewControllerBeforeViewController viewController: UIViewController) -> UIViewController? {  
        var index = self.indexOfViewController(viewController as! DataViewController)  
        if (index == 0) || (index == NSNotFound) {  
            return nil  
        }  
  
        index--  
        return self.viewControllerAtIndex(index, storyboard: viewController.storyboard!)  
    }  
  
    func pageViewController(pageViewController: UIPageViewController, viewControllerAfterViewController viewController: UIViewController) -> UIViewController? {  
        var index = self.indexOfViewController(viewController as! DataViewController)  
        if index == NSNotFound {  
            return nil  
        }  
  
        index++  
        if index == self.pageData.count {  
            return nil  
        }  
        return self.viewControllerAtIndex(index, storyboard: viewController.storyboard!)  
    }  
}
```



The diagram illustrates the mapping between external names and internal names. Two yellow callout boxes point to the storyboard parameter in the pageViewControllers methods. The left box is labeled "EXTERNAL NAME" and the right box is labeled "INTERNAL NAME".

# PAGE VIEW CONTROLLER TEMPLATE

## DATA VIEW CONTROLLER

```
import UIKit

class DataViewController: UIViewController {

    @IBOutlet weak var dataLabel: UILabel!
    var dataObject: String = ""

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

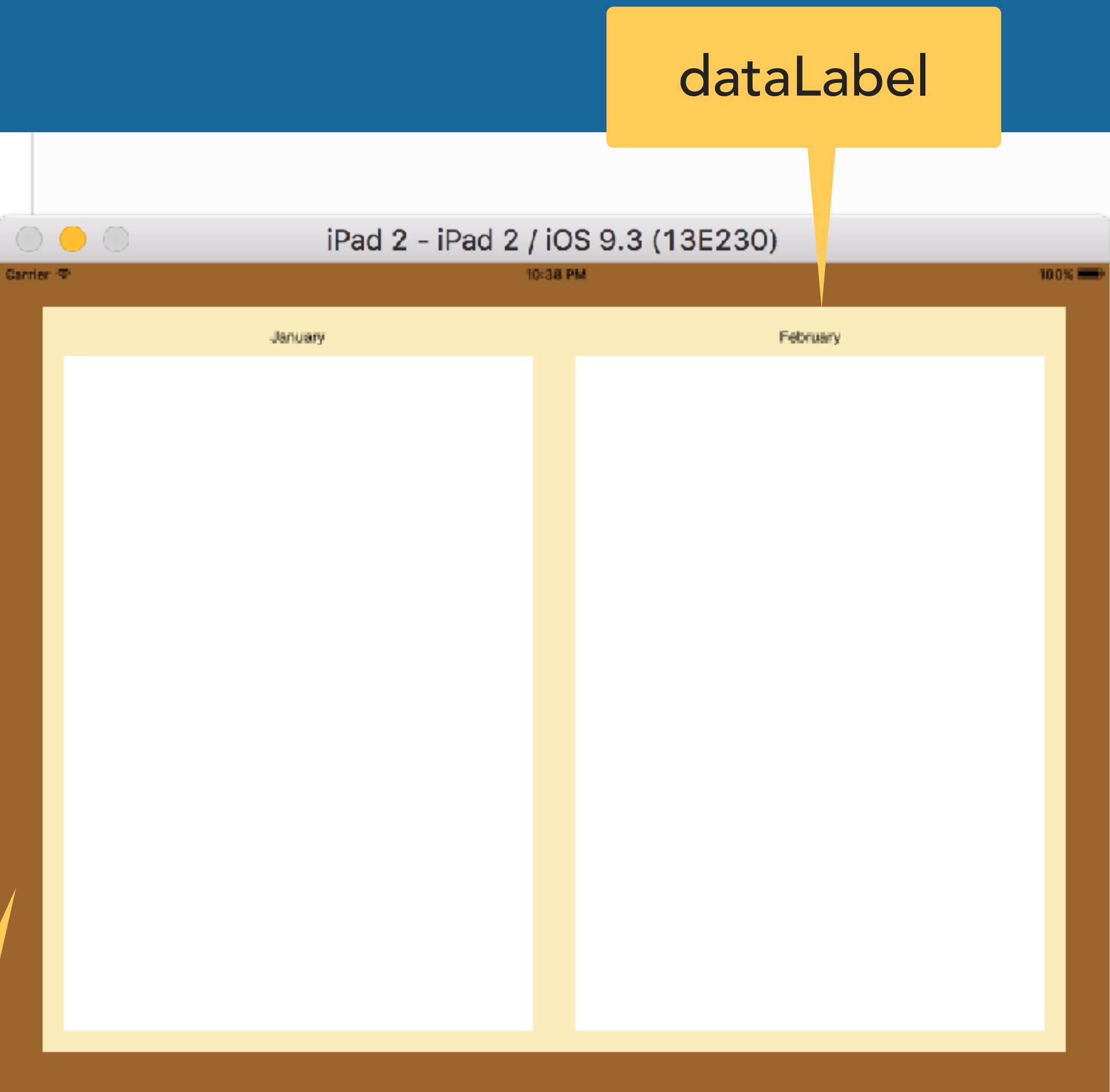
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    override func viewWillAppear(animated: Bool) {
        super.viewWillAppear(animated)
        self.dataLabel!.text = dataObject
    }
}
```

When is VC loaded?

Inset

dataLabel



**LET'S BE LAZY**

# LAZY

```
var modelController: ModelController {  
    // Return the model controller object, creating it if necessary.  
    // In more complex implementations, the model controller may be  
    // passed to the view controller.  
    if _modelController == nil {  
        _modelController = ModelController()  
    }  
    return _modelController!  
}  
  
var _modelController: ModelController? = nil
```

# LAZY

- Lazy initialization is a technique for delaying the creation of an object until needed
- Swift `lazy` attribute
- Examples of best practices
  - Initial value for a property is not known until after the object is initialized
  - Initial value for a property is computationally intensive

```
var modelController: ModelController {  
    // Return the model controller object, creating it if necessary.  
    // In more complex implementations, the model controller may be  
    // passed to the view controller.  
    if _modelController == nil {  
        _modelController = ModelController()  
    }  
    return _modelController!  
}  
  
var _modelController: ModelController? = nil
```

# LAZY

Objective-C style pattern  
of lazy initialization

```
var modelController: ModelController {  
    // Return the model controller object, creating it if necessary.  
    // In more complex implementations, the model controller may be  
    // passed to the view controller.  
    if _modelController == nil {  
        _modelController = ModelController()  
    }  
    return _modelController!  
}  
  
var _modelController: ModelController? = nil
```

# LAZY

```
import UIKit

class RootViewController: UIViewController, UIPageViewControllerDelegate {

    var pageViewController: UIPageViewController?

    override func viewDidLoad() { ... }

    override func didReceiveMemoryWarning() { ... }

    lazy var modelController: ModelController = ModelController()

    // MARK: - UIPageViewController delegate methods

    func pageViewController(pageViewController: UIPageViewController, spineLocationForInterfaceOrientation
    )
```

# LAZY

## Objective-C style pattern of lazy initialization

```
1 // RootViewController.swift
2 // PagesAndPages
3 // Created by T. Andrew Binkowski on 3/26/16.
4 // Copyright © 2016 The University of Chicago, Department of Computer Science. All rights reserved.
5
6 import UIKit
7
8 class RootViewController: UIViewController, UIPageViewControllerDelegate {
9     var pageViewController: UIPageViewController?
10
11     override func viewDidLoad() { ... }
12
13     override func didReceiveMemoryWarning() { ... }
14
15
16     var modelController: ModelController {
17         // Return the model controller object, creating it if necessary.
18         // In more complex implementations, the model controller may be passed to the view controller.
19         if _modelController == nil {
20             _modelController = ModelController()
21         }
22         return _modelController!
23     }
24
25     var _modelController: ModelController? = nil
26
27
28     // MARK: - UIPageViewController delegate methods
29
30
31     func pageViewController(pageViewController: UIPageViewController, spineLocationForInterfaceOrientation: UIInterfaceOrientation) -> CGFloat {
32         return 0.5
33     }
34
35
36     // MARK: - UIScrollViewDelegate methods
37
38
39     func scrollViewDidScroll(scrollView: UIScrollView) {
40         let offset = scrollView.contentOffset.y
41         let page = Int(offset / 100)
42         self.pageViewController?.setViewControllers([self.modelController!.pageAtIndex(page)], direction: .Right, animated: true)
43     }
44
45
46     func scrollViewWillEndDragging(scrollView: UIScrollView, withVelocity velocity: CGPoint, targetContentOffset: UnsafePointer<CGPoint>) {
47         let offset = scrollView.contentOffset.y
48         let page = Int(offset / 100)
49         self.pageViewController?.setViewControllers([self.modelController!.pageAtIndex(page)], direction: .Right, animated: true)
50     }
51
52
53     func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
54         self.scrollViewWillEndDragging(scrollView, withVelocity: CGPointZero, targetContentOffset: UnsafePointer<CGPoint>(nil))
55     }
56
57
58
59     // MARK: - UIScrollViewDelegate methods
60
61
62     func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
63         self.scrollViewWillEndDragging(scrollView, withVelocity: CGPointZero, targetContentOffset: UnsafePointer<CGPoint>(nil))
64     }
65
66
67
68     // MARK: - UIScrollViewDelegate methods
69
70
71     func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
72         self.scrollViewWillEndDragging(scrollView, withVelocity: CGPointZero, targetContentOffset: UnsafePointer<CGPoint>(nil))
73     }
74
75
76
77     // MARK: - UIScrollViewDelegate methods
78
79
80     func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
81         self.scrollViewWillEndDragging(scrollView, withVelocity: CGPointZero, targetContentOffset: UnsafePointer<CGPoint>(nil))
82     }
83
84
85
86     // MARK: - UIScrollViewDelegate methods
87
88
89     func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
90         self.scrollViewWillEndDragging(scrollView, withVelocity: CGPointZero, targetContentOffset: UnsafePointer<CGPoint>(nil))
91     }
92
93
94
95     // MARK: - UIScrollViewDelegate methods
96
97
98     func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
99         self.scrollViewWillEndDragging(scrollView, withVelocity: CGPointZero, targetContentOffset: UnsafePointer<CGPoint>(nil))
100    }
101
102
103
```

```
//
// Created by T. Andrew Binkowski on 3/26/16.
// Copyright © 2016 The University of Chicago, Department of Computer Science. All rights reserved.
//
import UIKit
class RootViewController: UIViewController, UIPageViewControllerDelegate {
    var pageViewController: UIPageViewController?
    override func viewDidLoad() { ... }
    override func didReceiveMemoryWarning() { ... }
    lazy var modelController: ModelController = ModelController()
    //
    // MARK: - UIPageViewController delegate methods
    //
    func pageViewController(pageViewController: UIPageViewController, spineLocationForInterfaceOrientation: UIInterfaceOrientation) -> CGFloat {
        return 0.5
    }
}
```

# LAZY

COMPUTED PROPERTY  
- ALWAYS CALLED

```
//  
// Created by T. Andrew Binkowski on 3/26/16.  
// Copyright © 2016 The University of Chicago, Department of Com  
  
import UIKit  
  
class RootViewController: UIViewController, UIPageViewControllerData...  
  
var pageViewController: UIPageViewController?  
  
override func viewDidLoad() { ... }  
  
override func didReceiveMemoryWarning() { ... }  
  
lazy var modelController: ModelController = ModelController()  
  
//  
// MARK: - UIPageViewController delegate methods  
  
func pageViewCo...  
}  
  
LAZY IS ONLY EVALUATED ONCE
```



# ADVANCED iOS APPLICATION DEVELOPMENT

---

MPCS 51032 • SPRING 2020 • SESSION 1A

# SESSION 1C



# ADVANCED iOS APPLICATION DEVELOPMENT

---

MPCS 51032 • SPRING 2020 • SESSION 1C

# ASSIGNMENT 1

# POP QUIZ

What do John Travolta, Barack Obama, Shaquille O'Neil,  
Whoopi Goldberg, Glenn Beck, Jay Leno, Cheech Marin,  
Jerry Seinfeld, Sting, Queen Latifah, Jimmy Buffett,  
Madonna, Spike Lee, Jerry Garcia, Paul McCartney, Jamie  
Lee Curtis, Bob Dylan, The Dutchess of York and Steve  
Martin have in common?

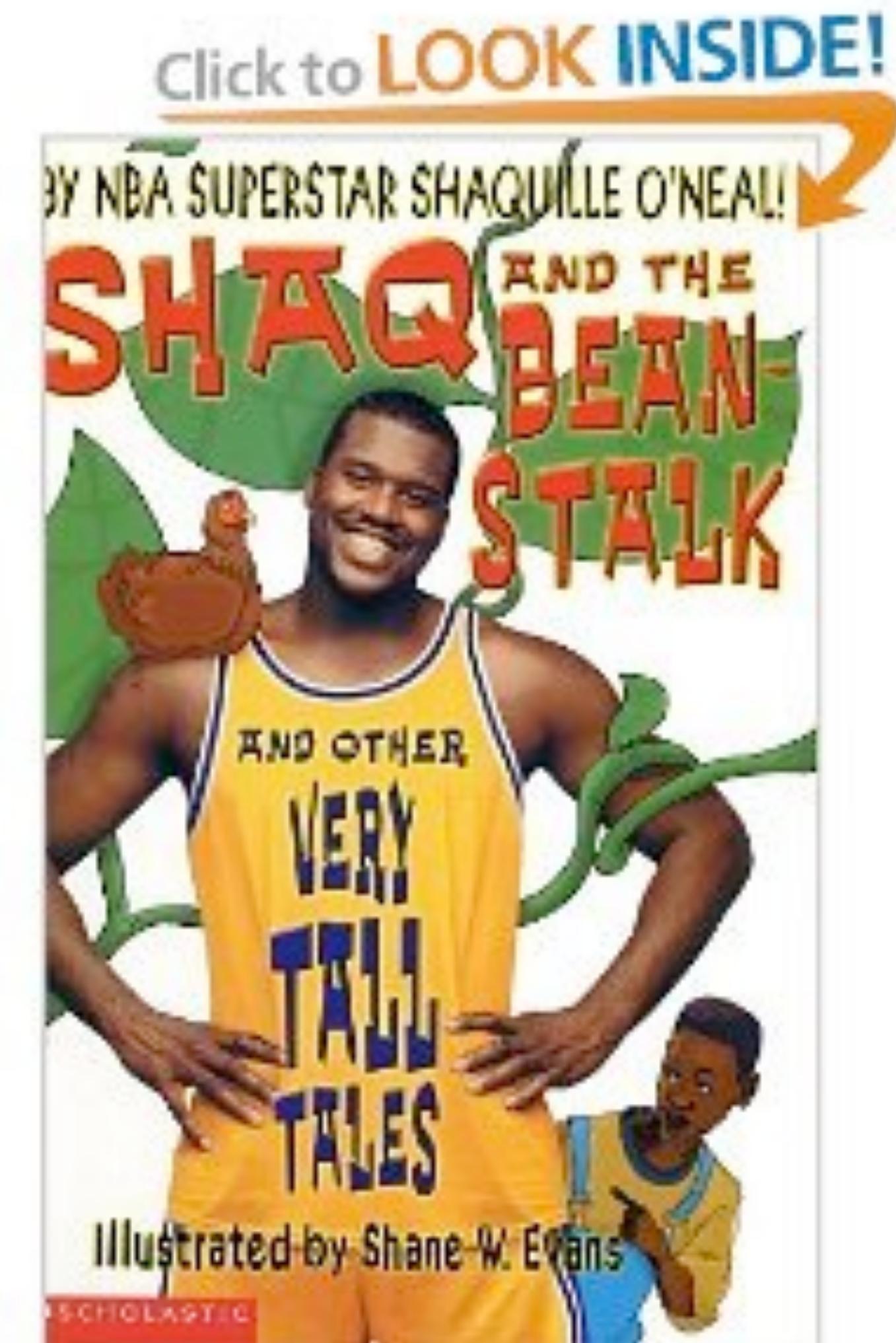
# ASSIGNMENT 1



# IF A CELEBRITY CAN DO IT, SO

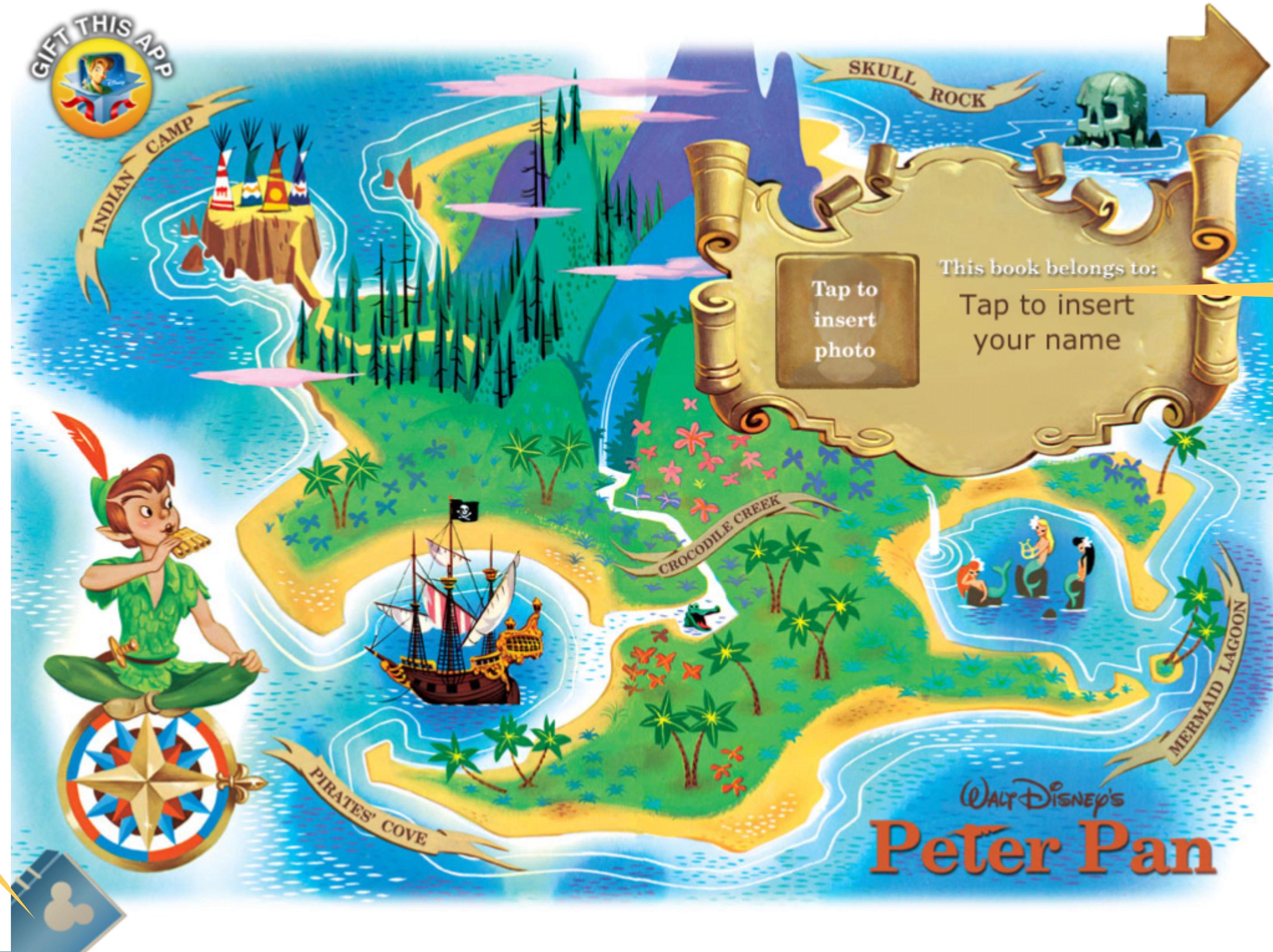
## CAN YOU

- Create a children's book using `UIPageViewController
- A hilarious, heartwarming and/or educational tale
  - At least 6 pages of unique content
- Must be better than "Shaq and Beanstalk"



# ASSIGNMENT 1

About the author

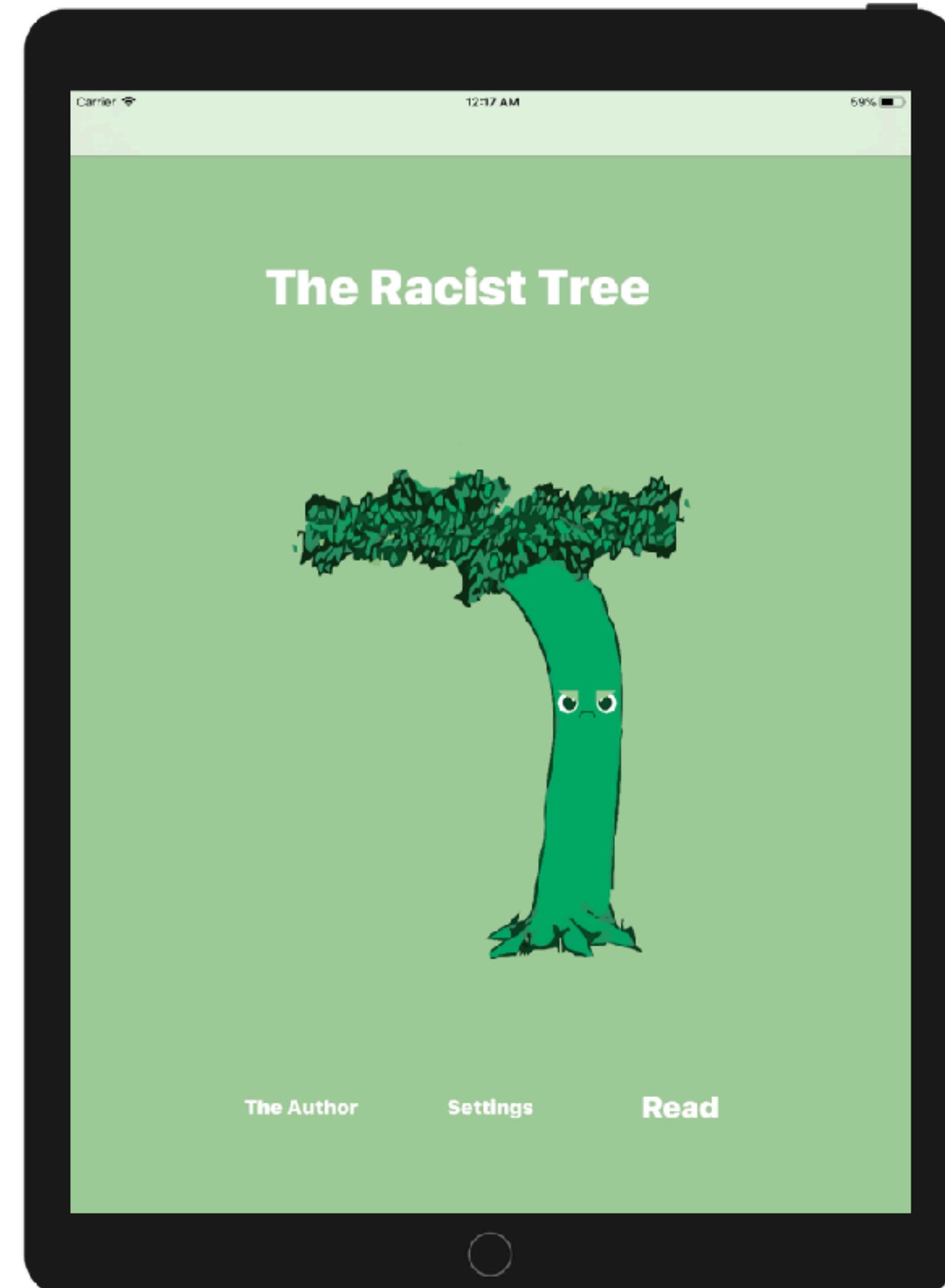


Settings

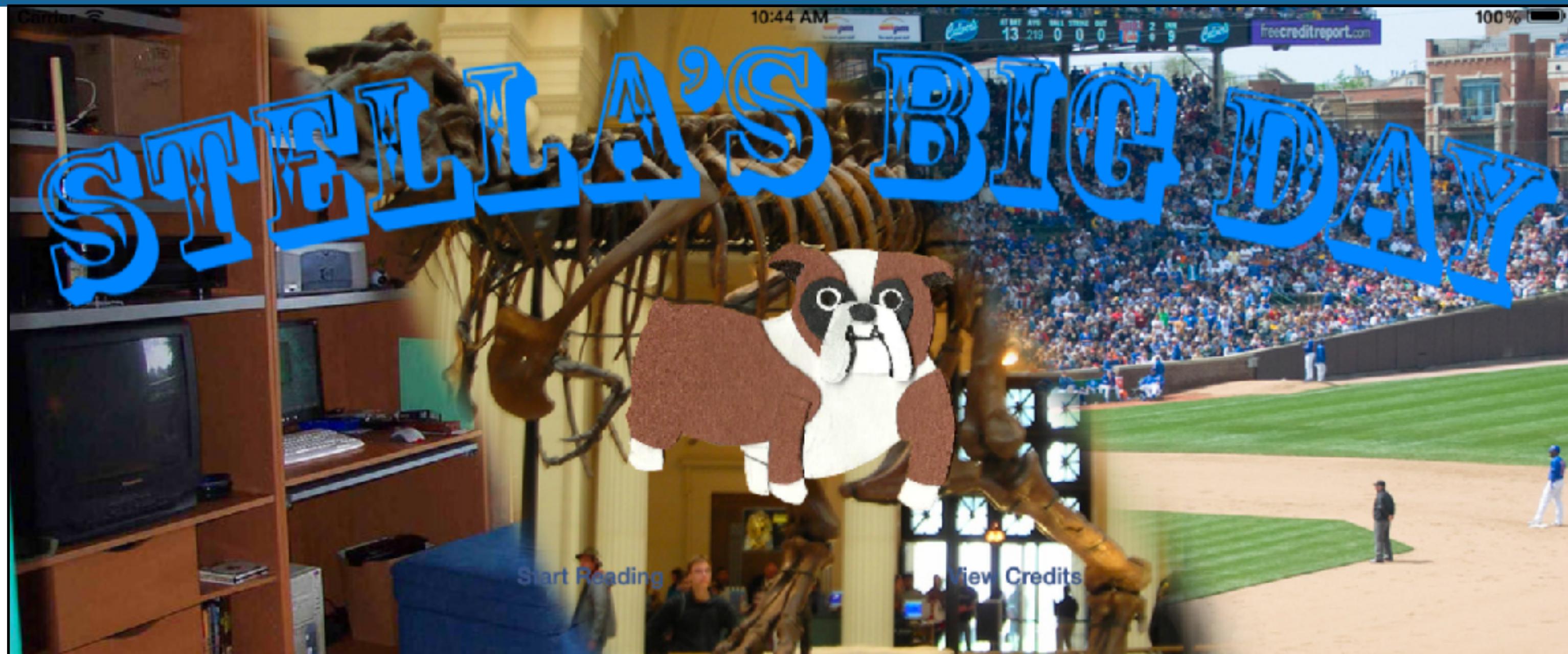
# ASSIGNMENT 1



# ASSIGNMENT 1



# ASSIGNMENT 1



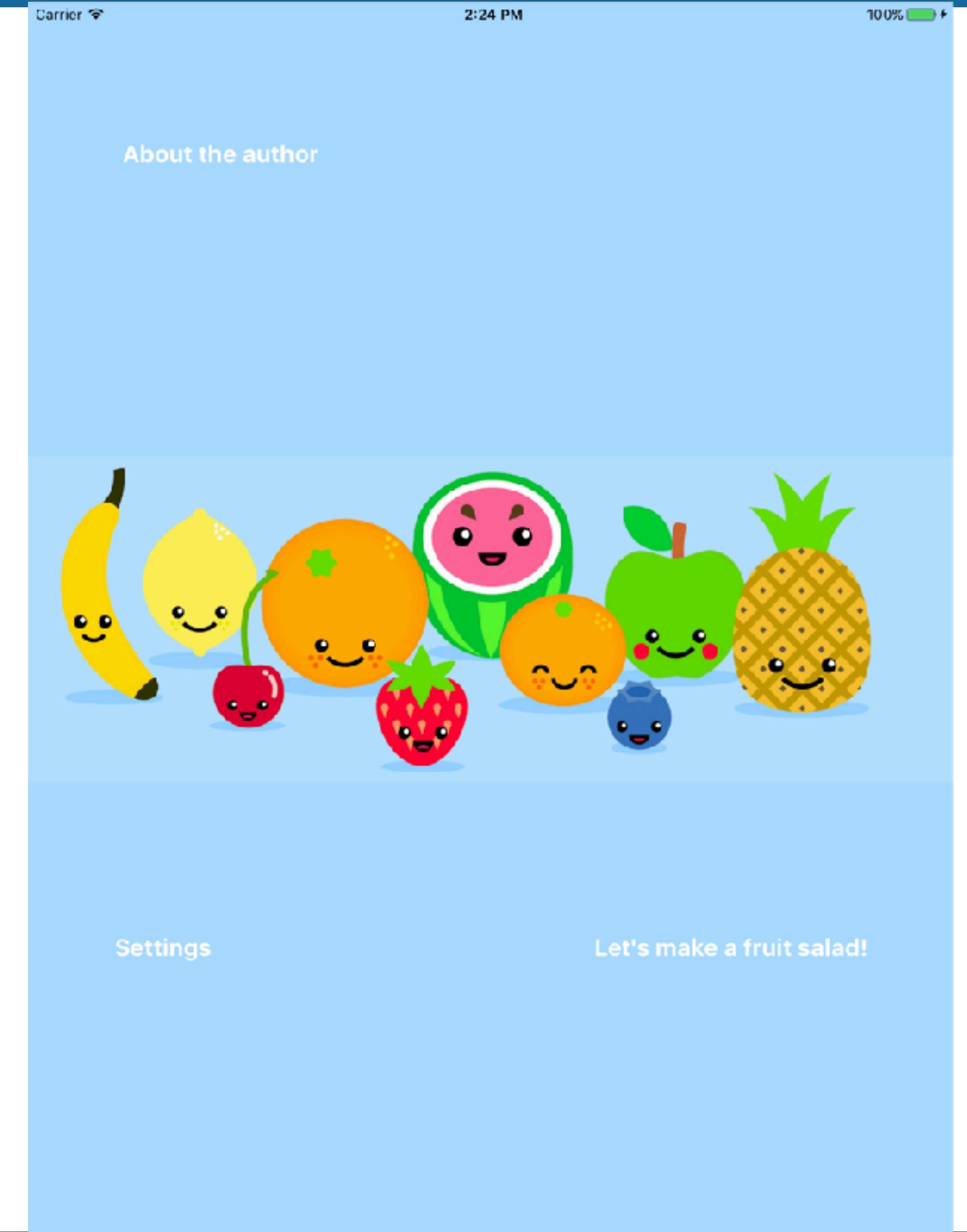
Tap this button to return to the home page



Tap this button to have the book read to you

Swipe or tap the edge of the screen to move between pages

Try tapping on different objects on screen to see what happens!



About the author

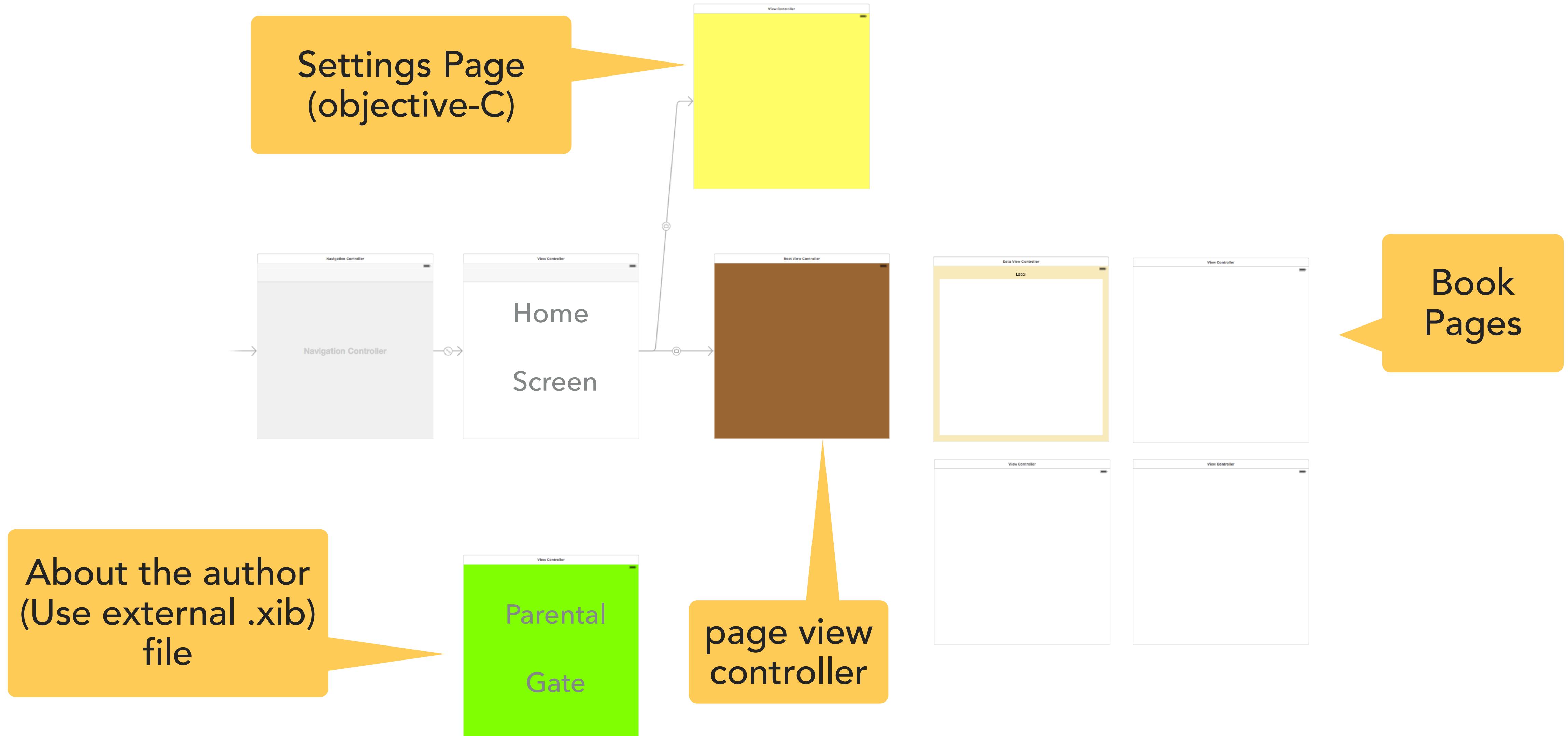
Settings

Let's make a fruit salad!

# ASSIGNMENT 1



# OVERALL REQUIREMENTS



# OVERALL REQUIREMENTS

- “Home Page” that represents the beginning of the book
- “Settings page” for “Read to Me” options (objective-c)
  - Automatically play on each page
  - Tap-to-play
  - Store these preferences in `UserDefaults`
- “About the Author” page protected by Parental Gate (external .xib file)
- Store currently reading page so that book opens to last view paged
- Create an App video of your book

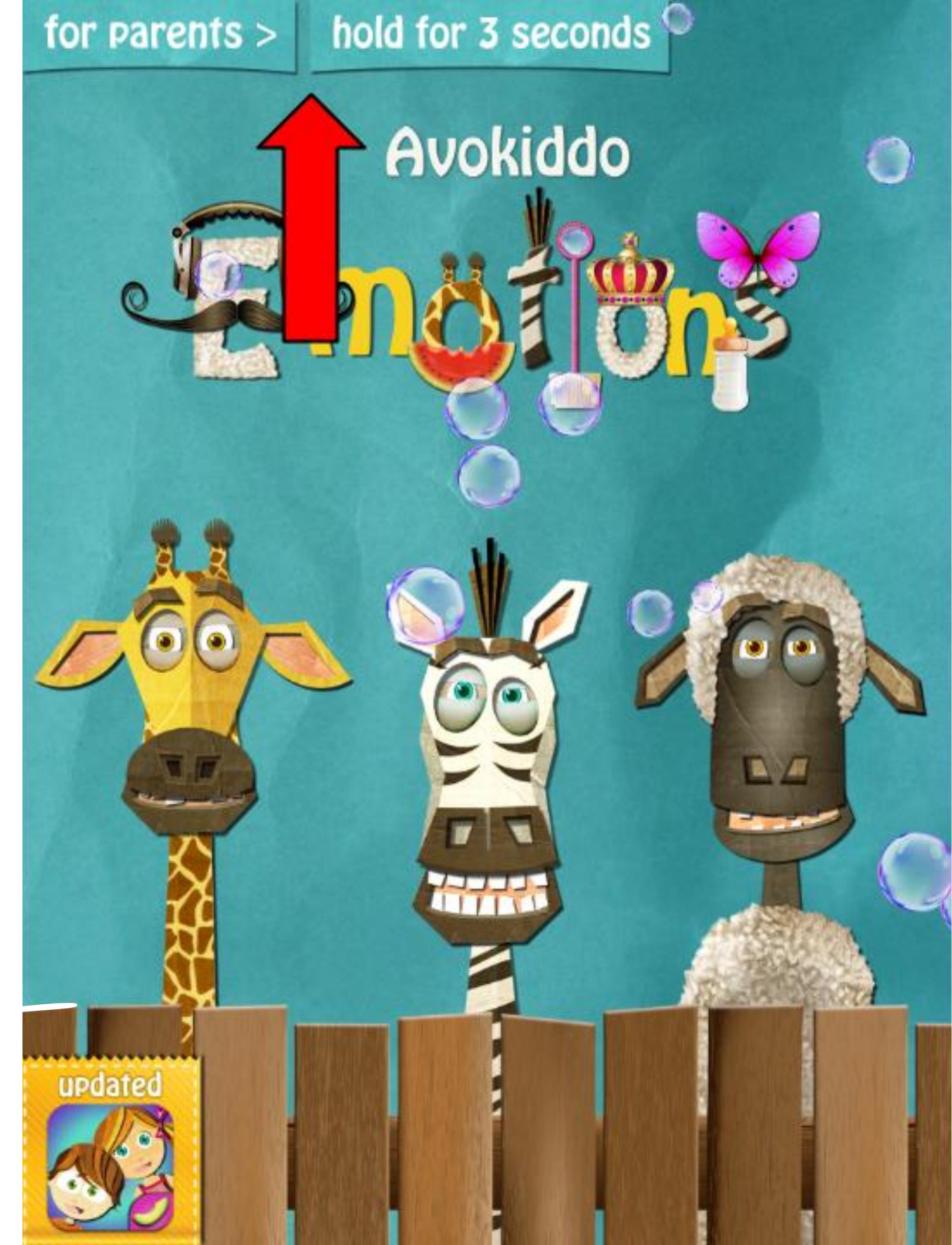
# BOOK PAGES

- Every content page must implement:
  - “Read to me” functionality
    - Record sound using Audacity, Garage Band, iPhone and email to self (UITextKit, AVSpeechSynthesis, next week)
    - Words should highlight as they are read
    - Start automatically or by button tap (depending on user preference in settings)
  - Picture and text
    - You must use `UIDynamics` for on-screen element (next week)
    - `UIGestureRecognizer` of on-screen element
    - UIViewPropertyAnimator based animations
    - Tap to hear sound effects
  - Button to return to home page

# PARENTAL GATE

SUBTITLE

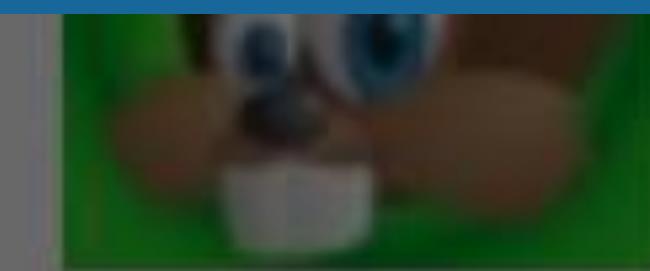
- Features are intended for parents, not kids
  - In-app purchase
  - Signup for news
  - Social Media
- Prevent kids from accessing inappropriate content



# PARENTAL GATE

- Implementation used to be a best practice, now it is required
  - 24. Kids Apps
    - 24.3- Apps primarily intended for use by kids under 13 must get parental permission or use a parental gate before allowing the user to link out of the app or engage in commerce
- This requires gating when linking out of the app or engaging in commerce such as selling in-app items.

# PARENTAL GATE



More Fun!



ARE YOU AN ADULT?

What is **20 - 5 =**

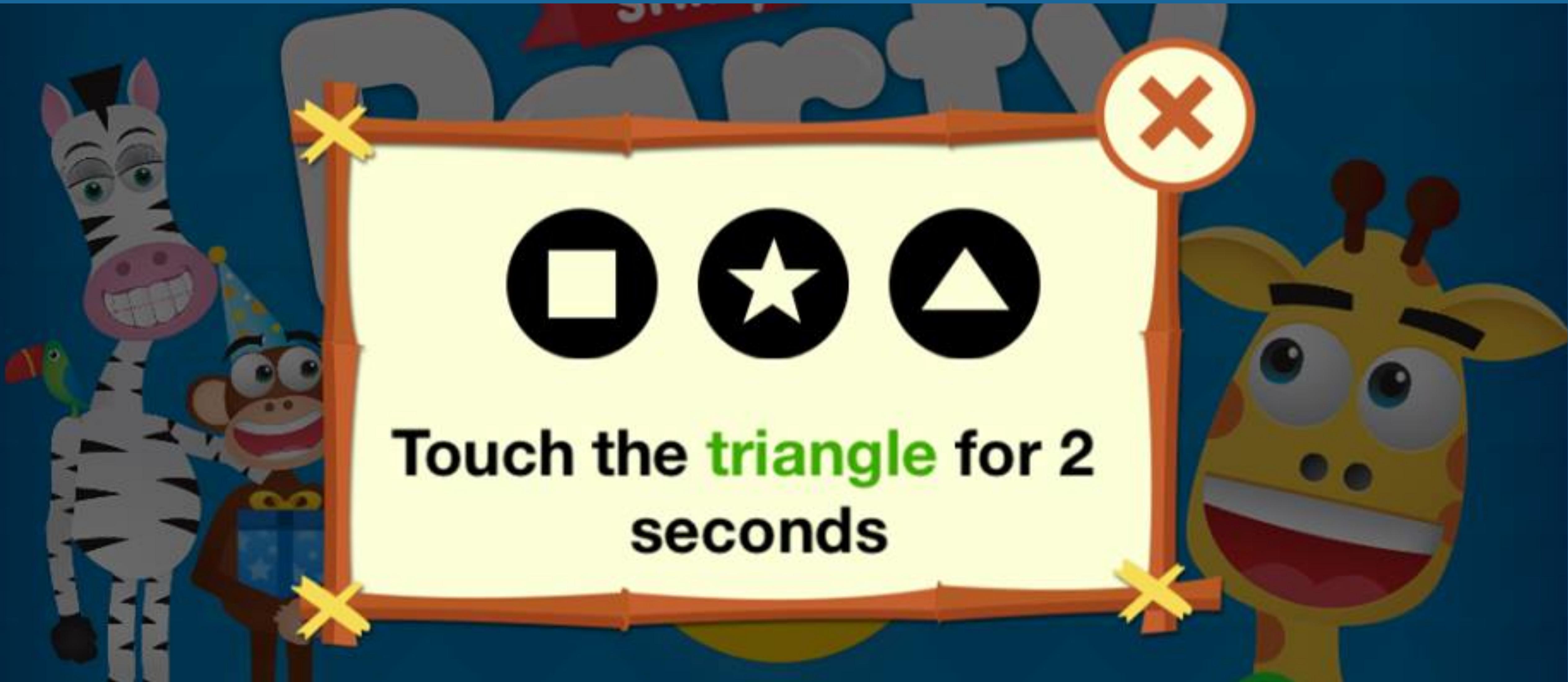
A white rectangular input field for entering the answer to the subtraction problem.

**Submit**



# PARENTAL GATE

Touch the **triangle** for 2  
seconds



# PARENTAL GATE

## REQUIREMENTS

- Implement a Parental Gate for your book app
  - Triggered when the user tries to view the “About the Author” screen
  - In practice the credits screen would typically have a link to social media
- Custom finger drawing view for validator
- Needs to use custom error handling do-try-throw
  - Custom ErrorType

# SETTINGS PAGE

## REQUIREMENTS

- Settings page in objective-c 😳
- Allow user to select preference to automatically read the page
  - Default should be "on"



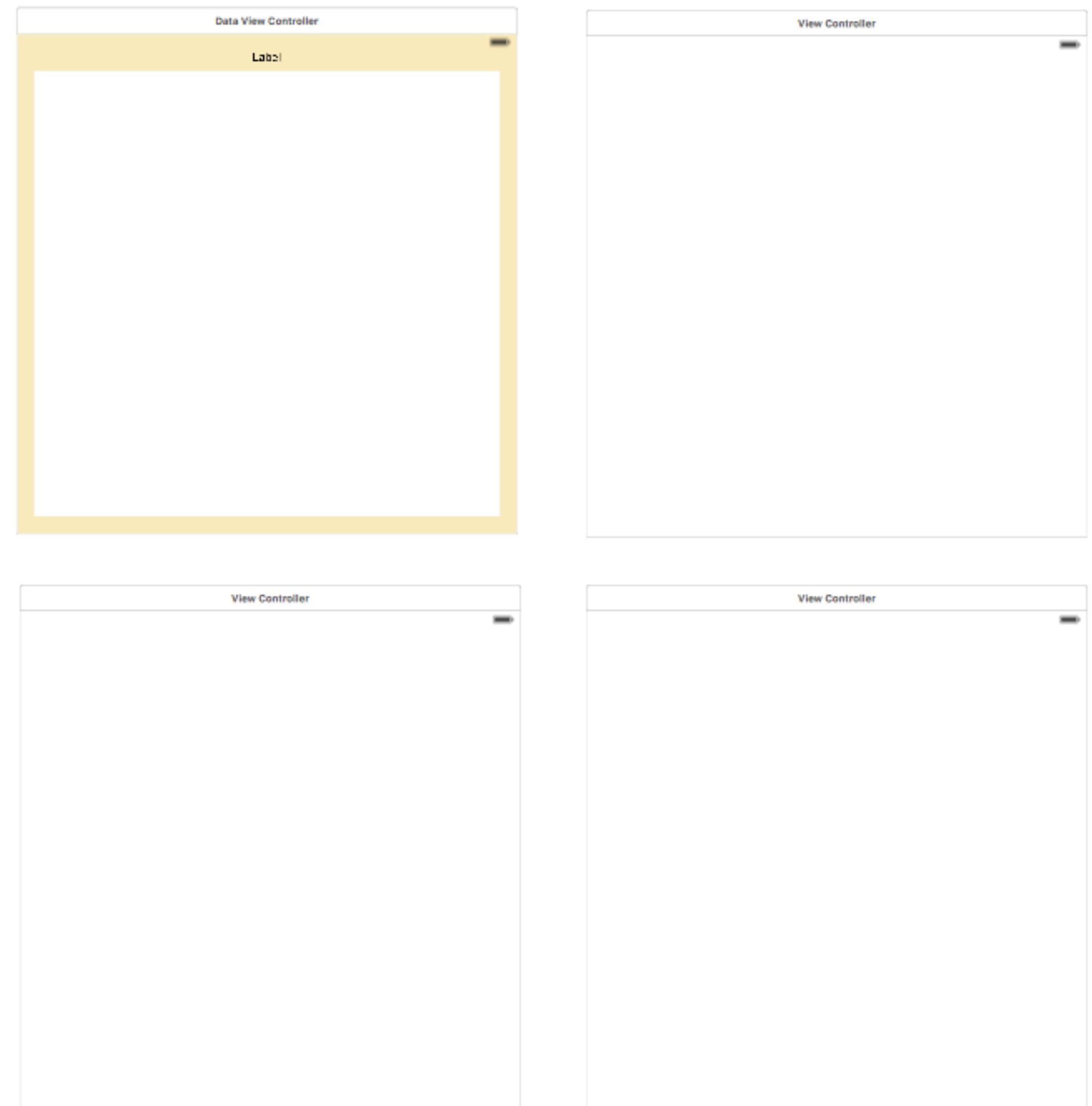
Read to me



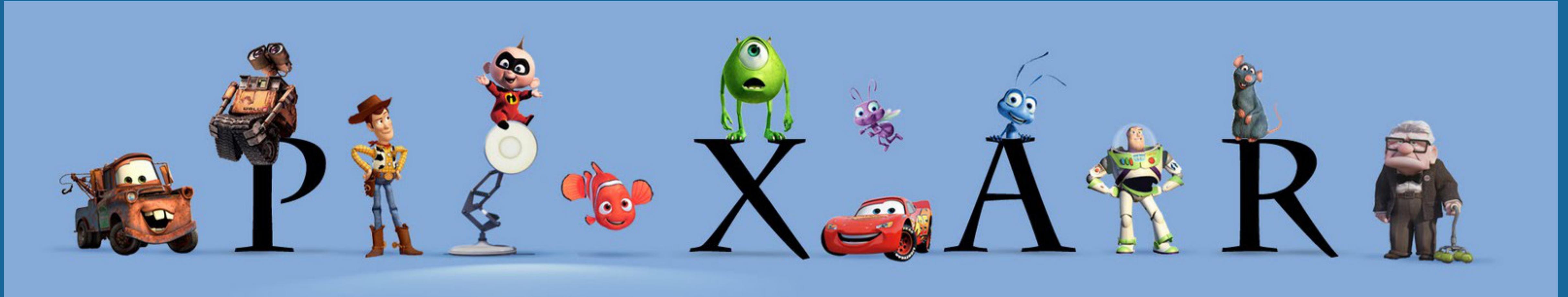
# BOOK PAGE OPTIONS

SUBTITLE

- Implementations options
  - View controller per page
  - Create view controller templates
  - Design 3 different styles of pages
  - Each page is a dictionary that includes images paths, text, sound file paths, animation start/stop frames



# DUE DATE



- Due April 16th at 5:29 PM
  - 2 Weeks
  - No late homework accepted
- “It’s all about story...story, story, story.” ~John Lasseter

# RECOMMENDED WORK FOR THIS WEEK

- Page View Controller
  - Get basic layout
  - Back to home button
  - Save current page
  - How to handle interrupted page turns
- Book pages
  - Draw them out
  - Sound effects
  - Interactive element
  - Text-to-speech

# GOING OVER NEXT WEEK

- Settings screen
  - Obj-c first app from Apple
  - Obj-c bridging header
- Parental gate
- App Preview
- Picture and text
  - `UIDynamics` for on-screen element
  - UIViewPropertyAnimator



# ADVANCED iOS APPLICATION DEVELOPMENT

---

MPCS 51032 • SPRING 2020 • SESSION 1C

# SESSION 1D



# ADVANCED iOS APPLICATION DEVELOPMENT

---

MPCS 51032 • SPRING 2020 • SESSION 1D

# ATTRIBUTED STRINGS

# NSATTRIBUTEDSTRING



Boring

- UILabel text can be set for the entire string

# NSATTRIBUTEDSTRING

- Attributed string

- An association of characters and their attributes

- Available since iOS5

- `NSAttributedString` object

- Manages character strings and associated sets of attributes

Class

## NSAttributedString

A string that has associated attributes (such as visual style, hyperlinks, or accessibility data) for portions of its text.

SDKs

iOS 3.2+

macOS 10.0+

tvOS 9.0+

watchOS 2.0+

Framework

Foundation

[On This Page](#)

[Overview](#) ⓘ

[Topics](#) ⓘ

[Relationships](#) ⓘ

[See Also](#) ⓘ

---

## Overview

An NSAttributedString object manages character strings and associated sets of attributes (for example, font and kerning) that apply to individual characters or ranges of characters in the string. An association of characters and attributes is called an attributed string. The cluster's two p

# NSATTRIBUTEDSTRING

- Applies to individual characters or ranges of characters in the string
  - `NSMakeRange(0, 15)`
- `NSMutableAttributedString` for modifiable attributed strings

Class

## NSAttributedString

A string that has associated attributes (such as visual style, hyperlinks, or accessibility data) for portions of its text.

SDKs

iOS 3.2+

macOS 10.0+

tvOS 9.0+

watchOS 2.0+

Framework

Foundation

On This Page

[Overview](#) ⓘ

[Topics](#) ⓘ

[Relationships](#) ⓘ

[See Also](#) ⓘ

### Overview

An NSAttributedString object manages character strings and associated sets of attributes (for example, font and kerning) that apply to individual characters or ranges of characters in the string. An association of characters and their attributes is called an attributed string. The cluster's two public

# NSATTRIBUTEDSTRING



- That's more like it!

# NSATTRIBUTEDSTRING

## SUBTITLE

- An attributed string identifies attributes by name
  - Uses NSDictionary object to store a value under the given name
- You can assign any attribute name/value pair you wish to a range of characters

## Using Attributed Strings with Labels

Create a UILabel, change the color and round the corners of the label. Font and color pro

```
let helloLabel = UILabel(frame: CGRect(x: 0, y: 0, width: 600, height: 100))
helloLabel.backgroundColor = UIColor.yellow
helloLabel.layer.masksToBounds = true
helloLabel.layer.cornerRadius = 10.0
helloLabel.textAlignment = NSTextAlignment.center
helloLabel.text = "Hello Label!"
```

Hello Label!

Let the label use an attributed string instead

```
var attributedString = NSMutableAttributedString(string: "Fancy Hello World!",
                                                attributes: [
                                                    NSAttributedStringEncoding.font:UIFont(name:
```

```
helloLabel.attributedText = attributedString
```

Fancy Hello World!

# NSATTRIBUTEDSTRING

```
let helloLabel = UILabel(frame: CGRect(x: 0, y: 0, width: 600, height: 100))
helloLabel.backgroundColor = UIColor.yellow
helloLabel.layer.masksToBounds = true
helloLabel.layer.cornerRadius = 10.0
helloLabel.textAlignment = NSTextAlignment.center
helloLabel.text = "Hello Label!"
```



Hello Label!

# NSATTRIBUTEDSTRING

Let the label use an attributed string instead

```
var attributedString = NSMutableAttributedString(string: "Fancy Hello World!",  
                                              attributes: [  
                                                NSAttributedStringEncodingKey.font:UIFont(name: "Courier", size: 50.0)!])  
helloLabel.attributedText = attributedString
```

Fancy Hello World!

# NSATTRIBUTEDSTRING

SUBTITLE

- If you are using attributed strings with the Core Text framework, you can also use the attribute keys defined by that framework
  - iOS X, standard attribute keys are described in the “Constants” section of NSAttributedString UIKit Additions Reference
  - OS X, standard attribute keys are described in the “Constants” section of NSAttributedString AppKit Additions Reference.

## Attributed String Showcase

Note that attributed string attributes are cumulative. They will persist on the string until you change/undo them. For highlighting words for the homework assignment, you will need to reset the previously highlighted words to their default attributes.

Different attributes between platforms

```
specified range defined by `NSRange`. ##  
SFFontAttributeName, value: UIFont(name:  
: 18.0)!, range: NSRange(location:0,length:5)  
  
!!  
  
the letter and change color ##  
SFFontAttributeName, value: UIFont(name:  
: NSRange(location: 0, length: 1))
```

Fancy Hello World!

```
27  
28  
29 // Stroke the first letter in red
```

# NSATTRIBUTEDSTRING

## Attributed String Programming Guide



### Table of Contents

[Introduction](#)

[Attributed Strings](#)

[Creating Attributed Strings in Cocoa](#)

▶ [Accessing Attributes](#)

▶ [Changing an Attributed String](#)

[Drawing Attributed Strings](#)

▶ [RTF Files and Attributed Strings](#)

▶ [Formatted Documents and Attributed Strings](#)

[Word and Line Calculations in Attributed Strings](#)

[Standard Attributes](#)

[Revision History](#)

[Index](#)

[Next](#)

## Introduction to Attributed String Programming Guide

*Attributed String Programming Guide* describes the attributed string objects, instantiated from the `NSAttributedString` class or the `CFAttributedString` Core Foundation opaque type, which manage sets of text attributes, such as font and kerning, that are associated with character strings or individual characters.

## Who Should Read This Document

You should read this document if you need to work directly with attributed string objects.

## Organization of This Document

### OBJECTIVE-C REFERENCE

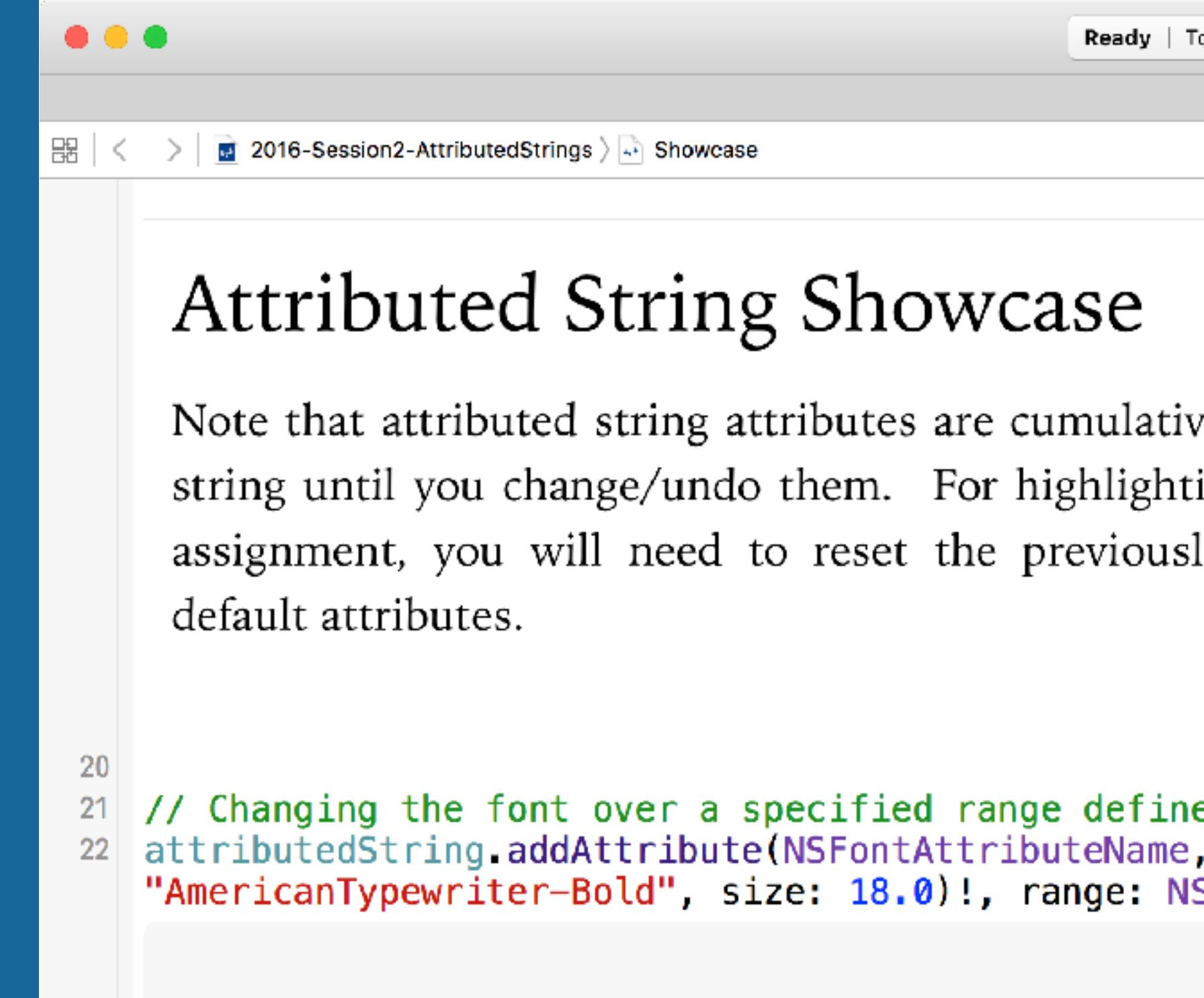
`NSAttributedString`

`NSMutableAttributedString`

# NSATTRIBUTEDSTRING

## SAMPLE ATTRIBUTES

- NSString \*const NSFontAttributeName;
- NSString \*const NSParagraphStyleAttributeName;
- NSString \*const NSForegroundColorAttributeName;
- NSString \*const NSBackgroundColorAttributeName;
- NSString \*const NSLigatureAttributeName;
- NSString \*const NSBaselineOffsetAttributeName;
- NSString \*const NSStrikethroughStyleAttributeName;
- NSString \*const NSSrokeColorAttributeName;
- NSString \*const NSSrokeWidthAttributeName;
- NSString \*const NSShadowAttributeName;



The screenshot shows a Xcode interface with a file named "2016-Session2-AttributedStrings>Showcase". The code in the editor is as follows:

```

20
21 // Changing the font over a specified range defined
22 attributedString.addAttribute(NSFontAttributeName,
  "AmericanTypewriter-Bold", size: 18.0)!, range: NSMakeRange(10, 10)
23
24 // Stroke the first letter in red
25 attributedString.addAttribute(NSStrokeColorAttributeName, color: NSColor.red,
  range: NSMakeRange(0, 1))
26
27 // Change the background color of the entire string
28 attributedString.addAttribute(NSBackgroundColorAttributeName, color: NSColor.cyan,
  range: NSMakeRange(0, attributedString.length))
29
30 // Change the foreground color of the entire string
31 attributedString.addAttribute(NSForegroundColorAttributeName, color: NSColor.purple,
  range: NSMakeRange(0, attributedString.length))
32
33 // Change the baseline offset of the entire string
34 attributedString.addAttribute(NSBaselineOffsetAttributeName, value: 10.0,
  range: NSMakeRange(0, attributedString.length))
35
36 // Add a ligature to the entire string
37 attributedString.addAttribute(NSLigatureAttributeName, value: true,
  range: NSMakeRange(0, attributedString.length))
38
39 // Add a strikethrough to the entire string
40 attributedString.addAttribute(NSStrikethroughStyleAttributeName, value: 1,
  range: NSMakeRange(0, attributedString.length))
41
42 // Add a stroke to the entire string
43 attributedString.addAttribute(NSStrokeColorAttributeName, color: NSColor.red,
  range: NSMakeRange(0, attributedString.length))
44
45 // Add a shadow to the entire string
46 attributedString.addAttribute(NSShadowAttributeName, color: NSColor.black,
  range: NSMakeRange(0, attributedString.length))

```

**Warning: In Swift 4,  
attributed objects have been  
replaced with a struct  
**NSAttributedStringKey****

# **ATTRIBUTED STRINGS IN SWIFT**

# ATTRIBUTED STRINGS IN SWIFT

Create attributes array

```
let attributes = [NSAttributedStringKey.backgroundColor: UIColor.red,  
                  NSAttributedStringKey.foregroundColor: UIColor.white,  
                  NSAttributedStringKey.font: UIFont(name: "Georgia", size: 50.0)!  
]  
  
var fancyAttributedString = NSMutableAttributedString(string: "Super Fancy", attributes: attributes)  
helloLabel.attributedText = fancyAttributedString
```

Create attributed string

Super Fancy

Set label property  
(not .text)

# NSATTRIBUTEDSTRINGS

```
let attributes = [NSAttributedStringKey.backgroundColor: UIColor.red,  
                  NSAttributedStringKey.foregroundColor: UIColor.white,  
                  NSAttributedStringKey.font: UIFont(name: "Georgia", size: 50.0)!  
]  
var fancyAttributedString = NSMutableAttributedString(string: "Super Fancy", attributes: attributes)  
helloLabel.attributedText = fancyAttributedString
```



Super Fancy

- Set `attributedText` property on UILabel to use an attributed string instead of `text`

# ATTRIBUTED STRINGS IN SWIFT

The screenshot shows a Mac OS X desktop with an Xcode playground window titled "Showcase.xcplaygroundpage". The window title bar includes "Ready | Today at 4:33 PM" and standard OS X window controls. The main area displays the following content:

Attributed String Showcase

Previous

```
7
8 import UIKit
9
10 // Create an attributed string
11 var attributedString = NSMutableAttributedString()
12
13 // Create an attributed string with fancy text
14 attributedString = NSMutableAttributedString(string: "Fancy Hello World!",  
UIFont(name: "ChalkboardSE-Regular", size: 50.0)!)]
```

A yellow callout bubble in the bottom right corner contains the word "Playground". Below the playground window, the text "Fancy Hello World!" is displayed in a large, bold, black font.

# **ATTRIBUTED STRINGS**

## **IN**

# **OBJECTIVE-C**



# NSATTRIBUTEDSTRINGS

```
{...}

- (void)writePrettyLabel
{
    NSMutableAttributedString *prettyString = [[NSMutableAttributedString alloc]
                                              initWithString:@"The Fox Jumped Over The Box!"];

    // Set font, notice the range is for the whole string
    UIFont *font = [UIFont fontWithName:@"Helvetica-Bold" size:18];
    [prettyString addAttribute:NSFontAttributeName value:font range:NSMakeRange(0, 10)];

    // Set background color, again for entire range
    [prettyString addAttribute:UIColorAttributeName
                           value:[UIColor colorWithRed:0.103 green:0.305 blue:0.492 alpha:1.0
                           range:NSMakeRange(5, 15)]];

    // Assign the attributed String to the @attributedText property of the UILabel
    self.prettyLabel.attributedText = prettyString;
}

@end
```

CREATE ATTRIBUTED STRING

SET ATTRIBUTES

SET LABEL PROPERTY

# NSATTRIBUTEDSTRINGS

```
infoString=@"This is an example of Attributed String";  
  
NSMutableAttributedString *attString=[[NSMutableAttributedString alloc] initWithString:infoString];  
NSInteger _stringLength=[infoString length];  
  
UIColor *_black=[UIColor blackColor];  
UIFont *font=[UIFont fontWithName:@"Helvetica-Bold" size:30.0f];  
[attString addAttribute:NSFontAttributeName value:font range:NSMakeRange(0, _stringLength)];  
[attString addAttribute:NSForegroundColorAttributeName value:_black range:NSMakeRange(0, _stringLength)];
```

This is an example of Attributed String.

# NSATTRIBUTEDSTRINGS

```
UIColor *_red=[UIColor redColor];
UIFont *font=[UIFont fontWithName:@"Helvetica-Bold" size:72.0f];
[attString addAttribute:NSFontAttributeName value:font range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSSrokeColorAttributeName value:_red range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSSrokeWidthAttributeName value:[NSNumber numberWithFloat:3.0] range:NSMakeRange(0,
_stringLength)];
```

This is an  
example of  
**Attributed String.**

# NSATTRIBUTEDSTRINGS

```
UIColor *_red=[UIColor redColor];
UIFont *font=[UIFont fontWithName:@"Helvetica-Bold" size:72.0f];
[attString addAttribute:NSFontAttributeName value:font range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSSrokeColorAttributeName value:_red range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSSrokeWidthAttributeName value:[NSNumber numberWithFloat:-3.0] range:NSMakeRange(0,
_stringLength)];
```

**This is an  
example of  
Attributed String.**

# NSATTRIBUTEDSTRINGS

```
UIColor *_red=[UIColor redColor];
UIColor *_green=[UIColor greenColor];
UIFont *font=[UIFont fontWithName:@"Helvetica-Bold" size:72.0f];
[attString addAttribute:NSFontAttributeName value:font range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSMutableAttributedStringAttribute value:_green range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSMutableAttributedStringAttribute value:_red range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSMutableAttributedStringAttribute value:[NSNumber numberWithFloat:-3.0] range:NSMakeRange(0, _stringLength)];
```

This is an  
example of  
**Attributed String.**

# NSATTRIBUTEDSTRINGS

```
UIColor *_green=[UIColor greenColor];
UIFont *font=[UIFont fontWithName:@"Helvetica-Bold" size:72.0f];

NSShadow *shadowDic=[[NSShadow alloc] init];
[shadowDic setShadowBlurRadius:5.0];
[shadowDic setShadowColor:[UIColor grayColor]];
[shadowDic setShadowOffset:CGSizeMake(0, 3)];

[attString addAttribute:NSFontAttributeName value:font range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSSForegroundColorAttributeName value:_green range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSShadowAttributeName value:shadowDic range:NSMakeRange(0, _stringLength)];
```

This is an  
example of  
**Attributed String.**

# NSATTRIBUTEDSTRINGS

```
UIColor *_red=[UIColor redColor];
UIFont *font=[UIFont fontWithName:@"Helvetica-Bold" size:72.0f];
[attString addAttribute:NSFontAttributeName value:font range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSMutableForegroundColorAttributeName value:_red range:NSMakeRange(0, _stringLength)];

[attString addAttribute:NSMutableKernAttributeName value:[NSNumber numberWithInt:5] range:NSMakeRange(0, _stringLength)];
```

This is an example of Attributed String.

# NSATTRIBUTEDSTRINGS

```
UIColor *_red=[UIColor redColor];
UIFont *font=[UIFont fontWithName:@"Helvetica-Bold" size:30.0f];
[attString addAttribute:NSFontAttributeName value:font range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSForegroundColorAttributeName value:_red range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSMutableAttributedString underlineStyle value:[NSNumber numberWithInt:2] range:NSMakeRange(0, _stringLength)];
```

This is an example of Attributed String.

```
UIColor *_blue=[UIColor blueColor];
UIColor *_blueL=[UIColor colorWithRed:0 green:0 blue:0.5 alpha:0.7];
UIFont *font=[UIFont fontWithName:@"Helvetica-Bold" size:30.0f];

[attString addAttribute:NSFontAttributeName value:font range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSForegroundColorAttributeName value:_blue range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSMutableAttributedString backgroundColor value:_blueL range:NSMakeRange(0, 20)];
```

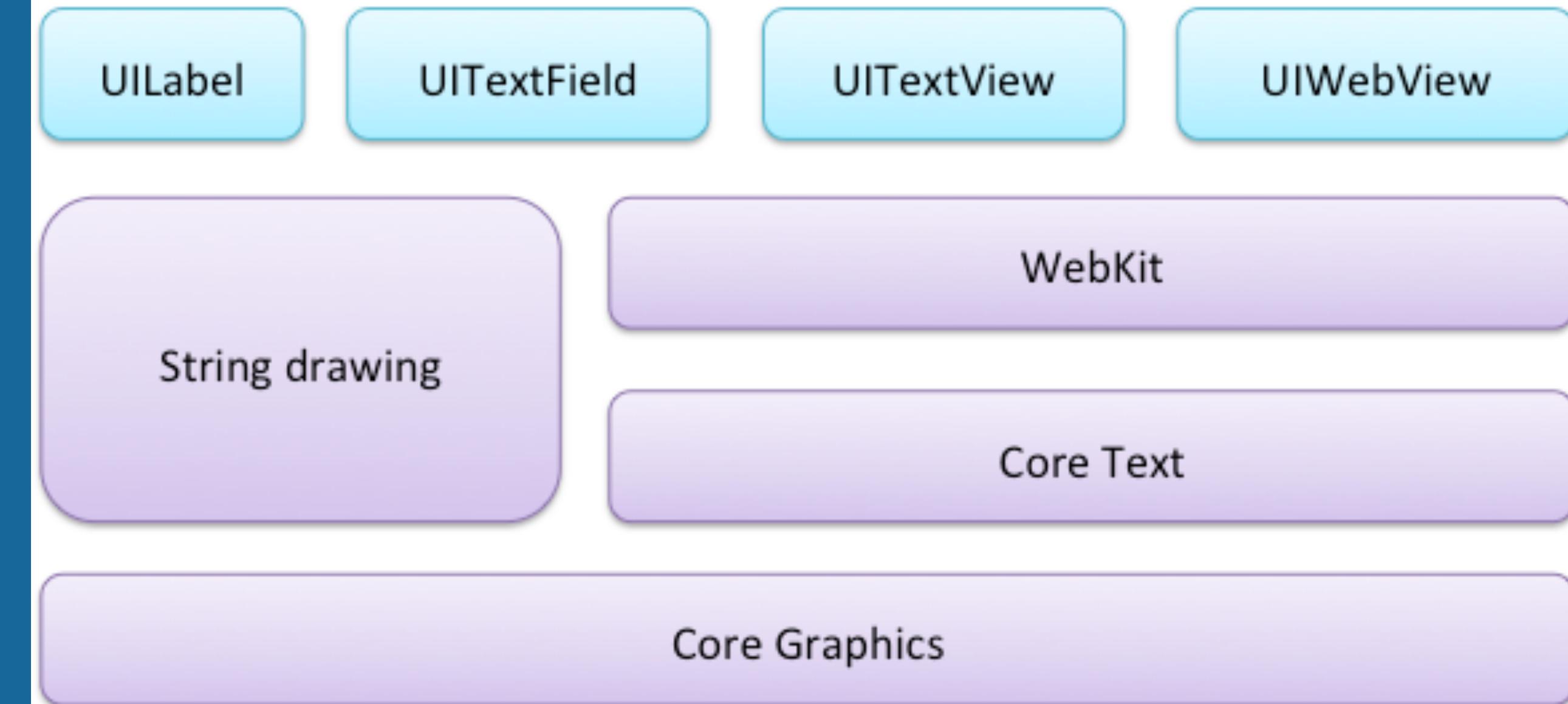
This is an example of Attributed String.

A BIT OF TEXT KIT

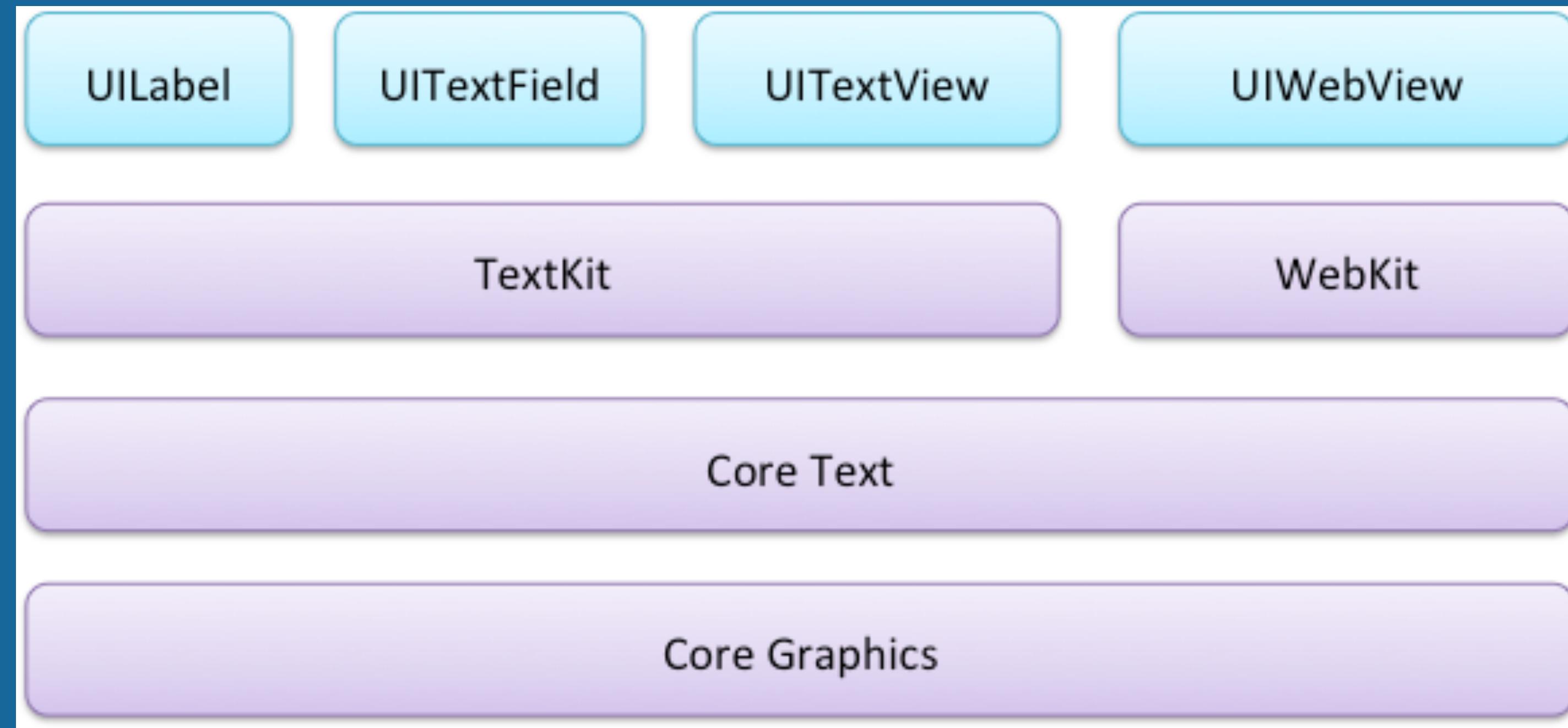
# TEXT KIT

SUBTITLE

- Text in iOS
  - iOS5- UIWebView
  - iOS6 NSAttributedString support
  - iOS7 TextKit
- Support the text heavy redesign of iOS7



# TEXT KIT

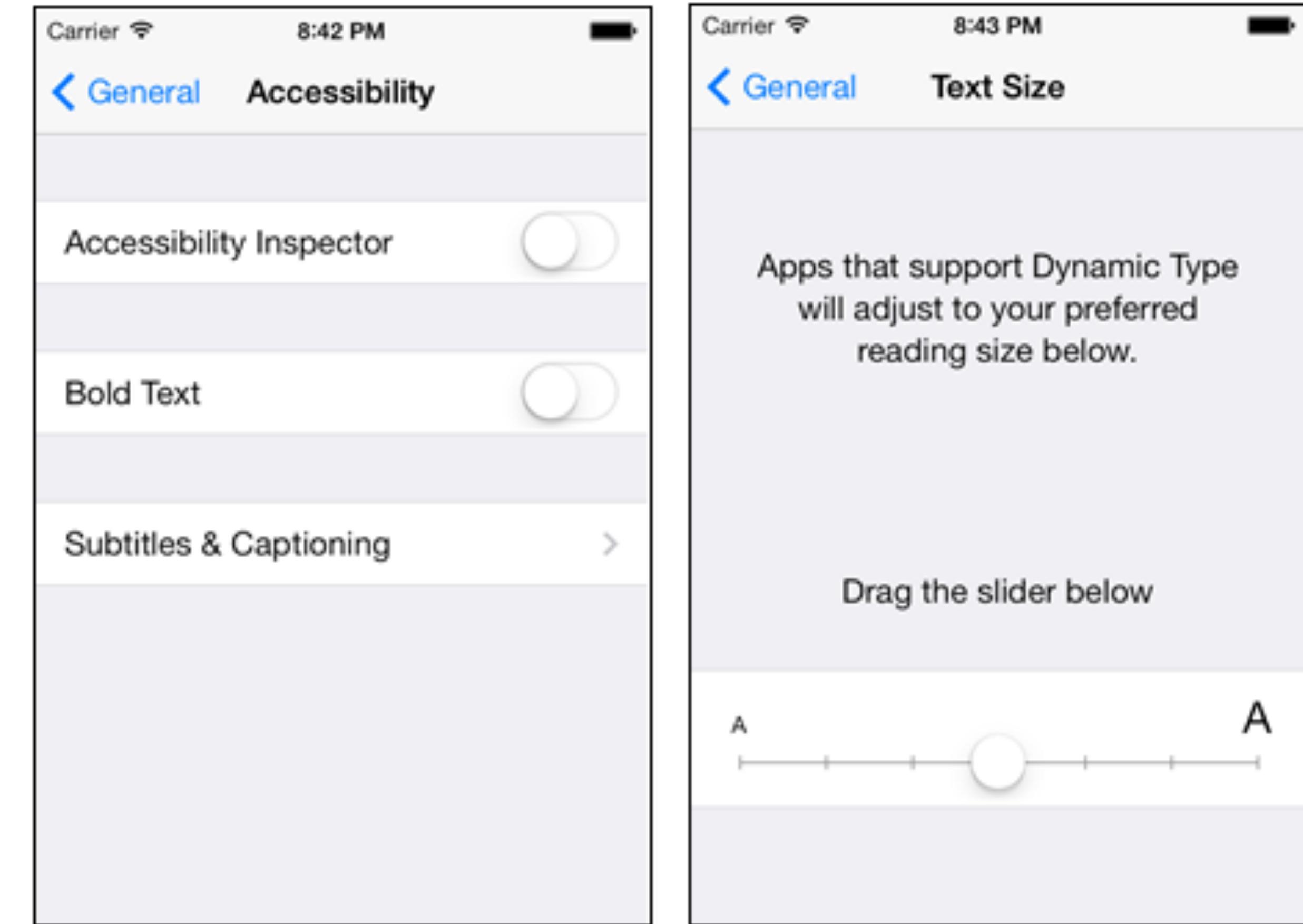


- Built on top of Core Text framework
  - Notoriously difficult API to work with

# TEXT KIT

## SUBTITLE

- Text Kit signature features
  - Dynamic type
  - Letterpress effect
  - Exclusion Path
  - Dynamic text formatting and storage

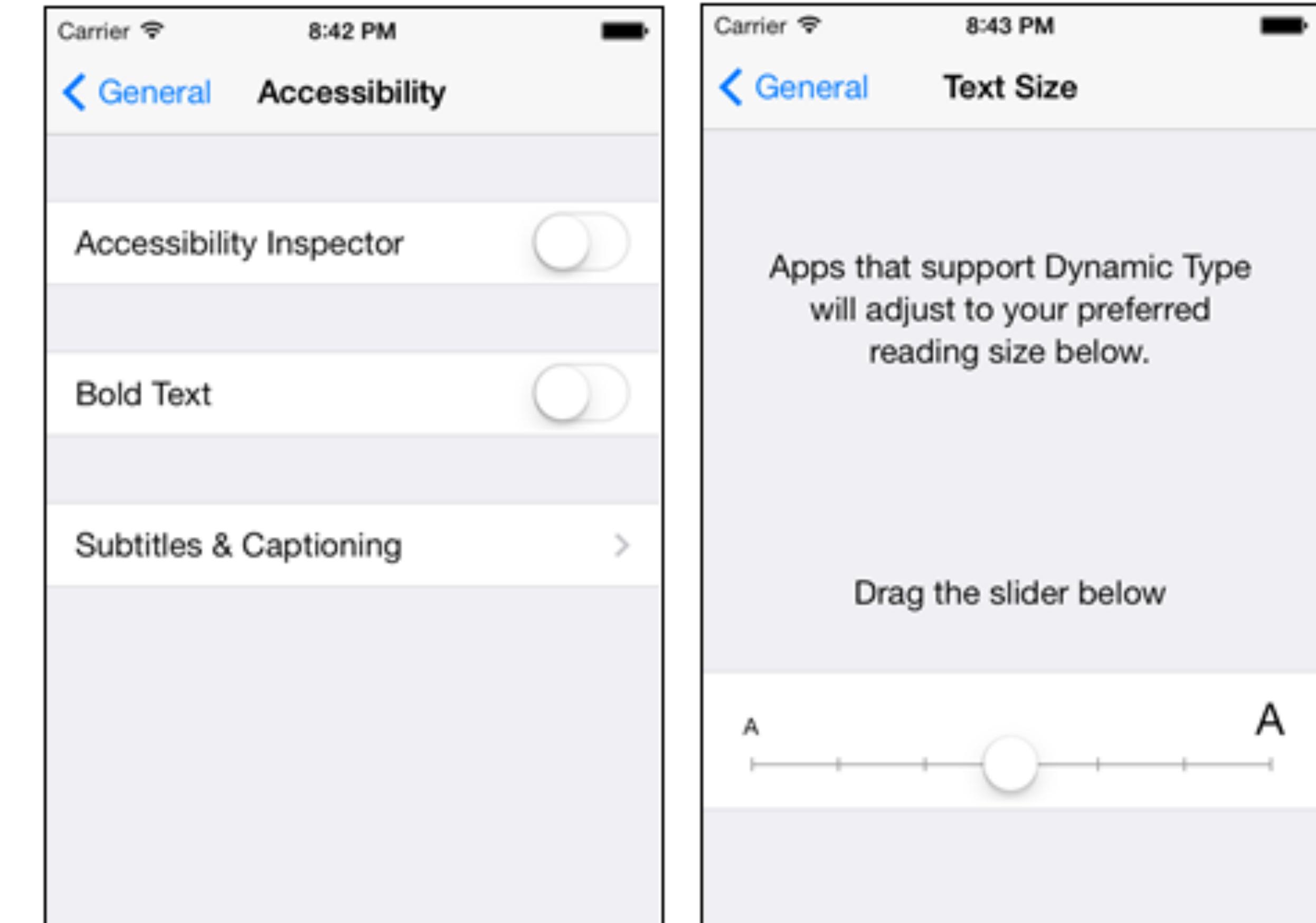


# TEXT KIT

## SUBTITLE

- DynamicType
  - When using default text, user can resize globally

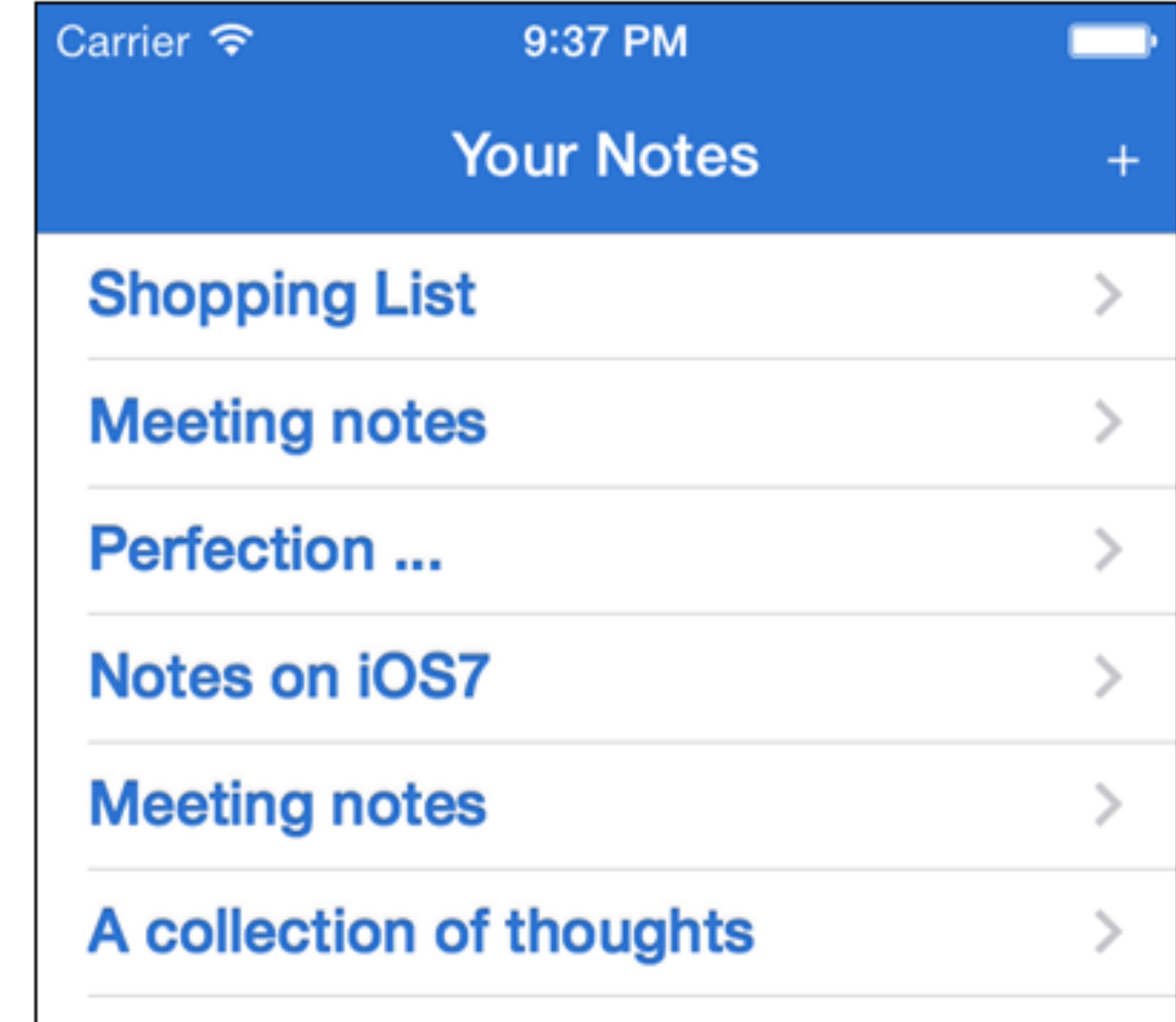
Users can ruin your interface



# TEXT KIT

SUBTITLE

- Letterpress effect
  - Letterpress effects add subtle shading and highlights to text that give it a sense of depth — much like the text has been slightly pressed into the screen



TEXT KIT

# Reminders

NSTextEffectLetterpressStyle

# TEXT KIT

## SUBTITLE

- Exclusion Path
  - Flowing text around images or other objects is a standard feature of most word processors
  - Text Kit allows you to render text around complex paths and shapes through exclusion paths



- TextKit tutorial
  - <http://www.raywenderlich.com/77092/text-kit-tutorial-swift>

# Text Kit Tutorial in Swift



*Gabriel Hauber on September 30, 2014*

**Update 12/12/14:** Updated for Xcode 6.1.1.

**Note from Ray:** This is a Swift update to a popular Objective-C tutorial on our site, released as part of the [iOS 8 Feast](#). Update by Gabriel Hauber, [Original post](#) by Tutorial Team member [Colin Eberhardt](#). Enjoy!

The way that iOS renders text continues to grow more powerful over the years as Apple adds more features and capabilities. The release of iOS 7



# ADVANCED iOS APPLICATION DEVELOPMENT

---

MPCS 51032 • SPRING 2020 • SESSION 1D

# SESSION 1E



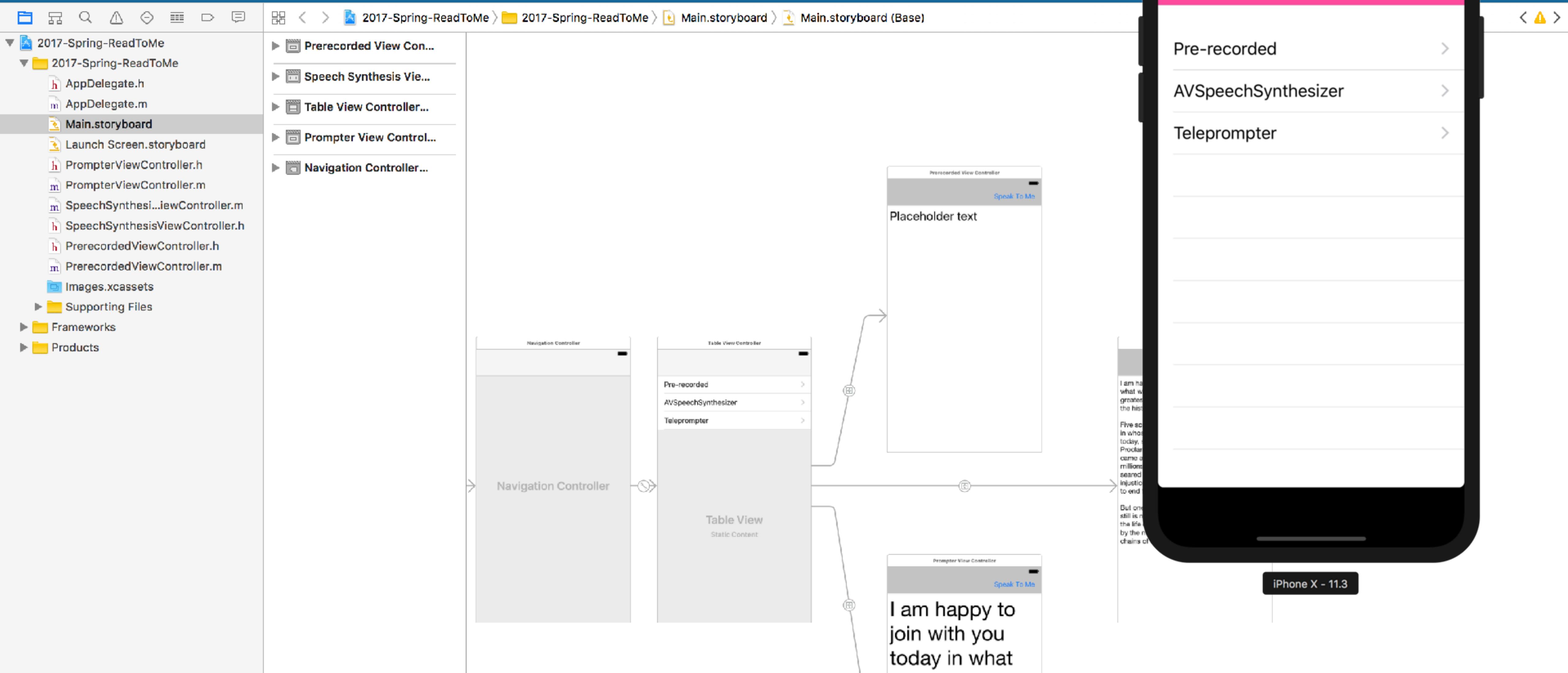
# ADVANCED iOS APPLICATION DEVELOPMENT

---

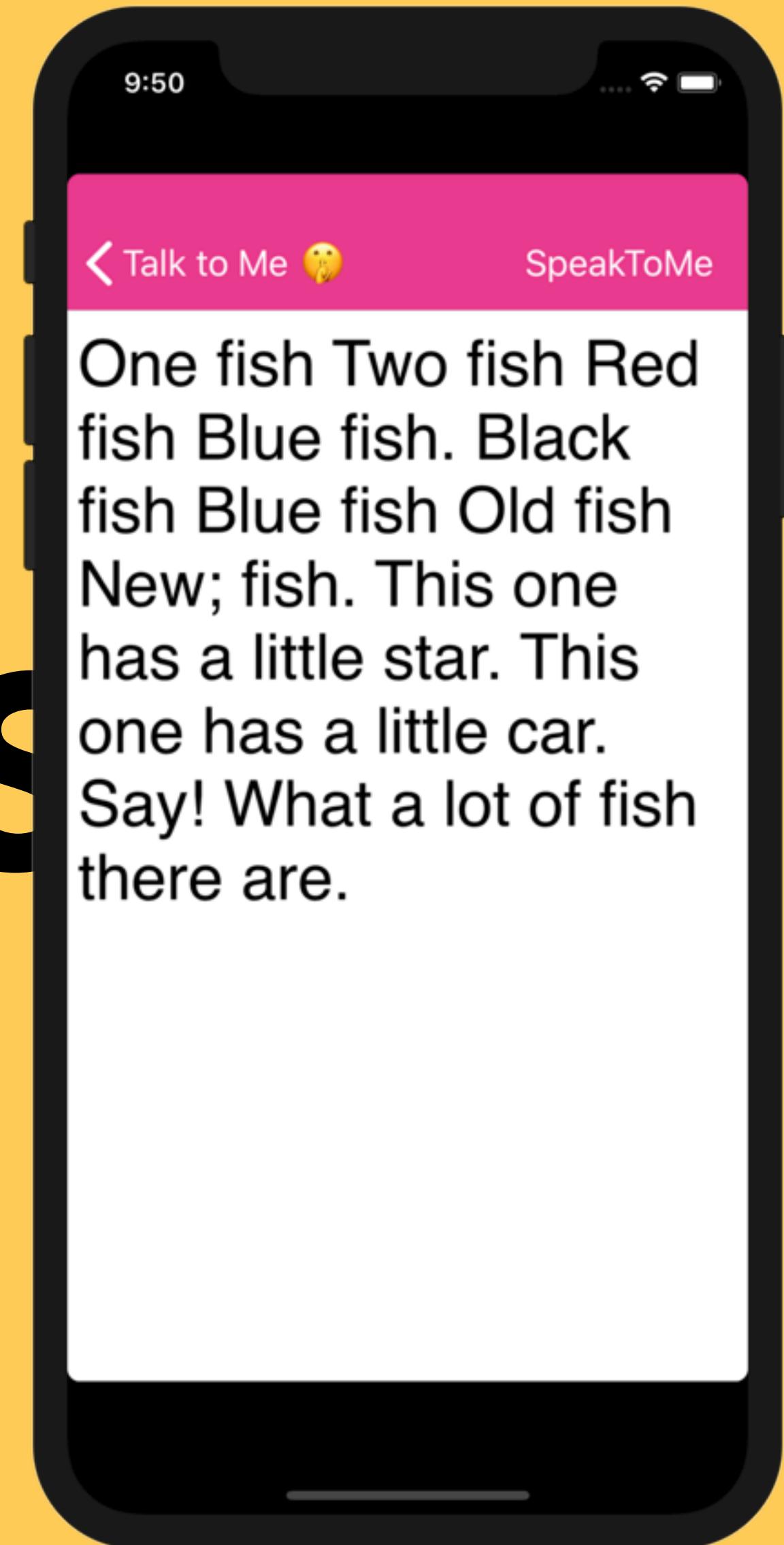
MPCS 51032 • SPRING 2020 • SESSION 1E

**READ-TO-ME DEMO**

# READ-TO-ME

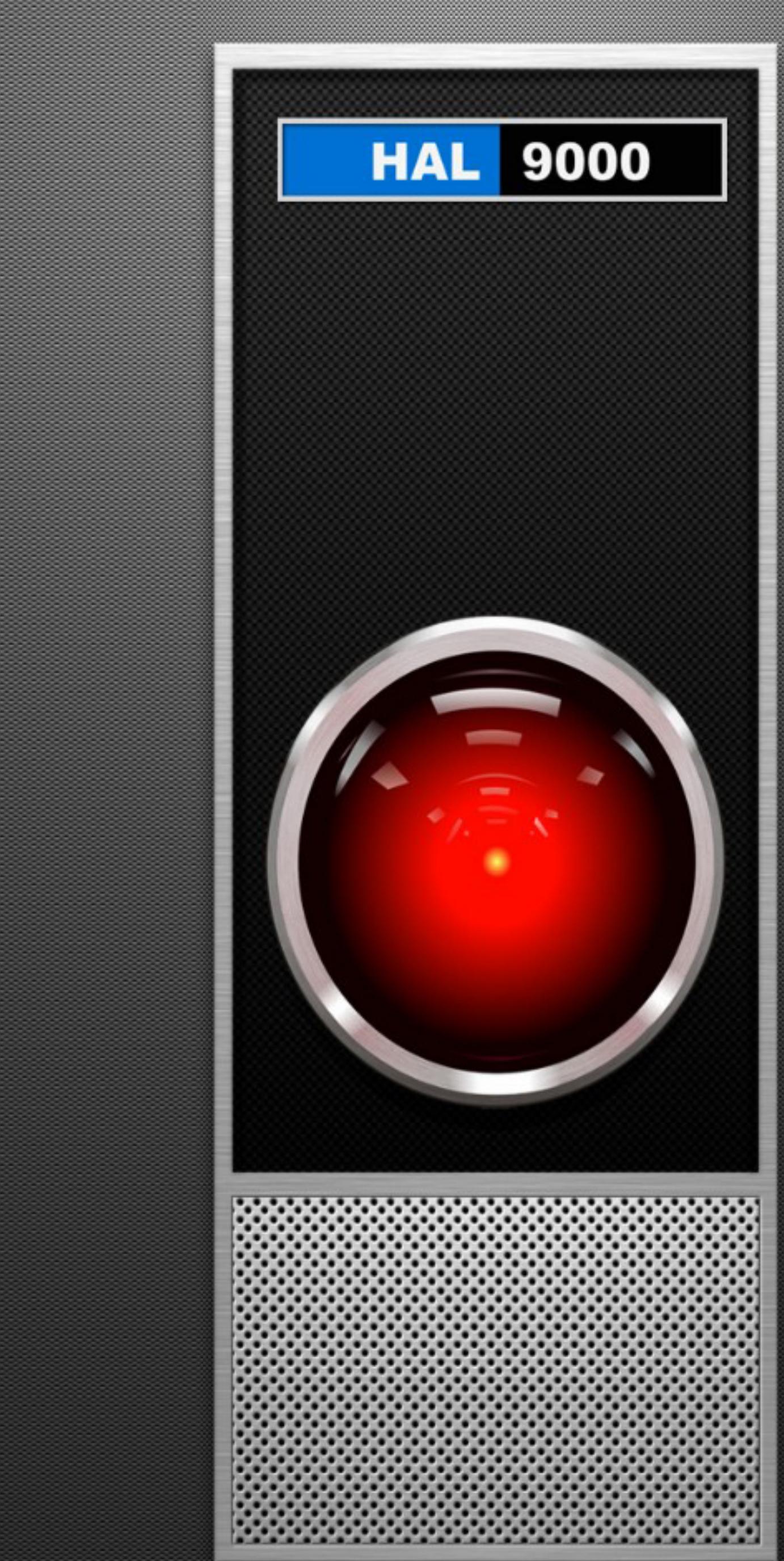


# SPEECH SYNTHESIS



# SPEECH SYNTHESIS

- AVSpeechSynthesizer
  - Introduced in iOS7
- Offline speech synthesis capability



## AVSpeechSynthesizer

The `AVSpeechSynthesizer` class produces synthesized speech from text on an iOS device, and provides methods for controlling or monitoring the progress of ongoing speech.

Language

Swift | [Objective-C](#)

SDKs

iOS 7.0+

tvOS 7.0+

watchOS 2.0+

## Overview

To speak some amount of text, you must first create an `AVSpeechUtterance` instance containing the text. (Optionally, you may also use the utterance object to control parameters affecting its speech, such as voice, pitch, and rate.) Then, pass it to the `speak(_:)` method on a speech synthesizer instance to speak that utterance.

The speech synthesizer maintains a queue of utterances to be spoken. If the synthesizer is

On This Page

[Overview](#) ▾

[Symbols](#) ▾

[Relationships](#) ▾

# SPEECH SYNTHESIS

- Support for 30 voices (specific country variations)
- Adjust rate of speed
- Adjust pitch
- Adjust volume
- Delegate protocol to allow interaction during speech

Class

## AVSpeechSynthesizer

The AVSpeechSynthesizer class produces synthesized speech from text on an iOS device, and provides methods for controlling or monitoring the progress of ongoing speech.

Language  
Swift | Objective-C

SDKs  
iOS 7.0+  
tvOS 7.0+  
watchOS 2.0+

On This Page  
[Overview](#) ▾  
[Symbols](#) ▾  
[Relationships](#) ▾

### Overview

To speak some amount of text, you must first create an [AVSpeechUtterance](#) instance containing the text. (Optionally, you may also use the utterance object to control parameters affecting its speech, such as voice, pitch, and rate.) Then, pass it to the [speak\(\\_:\) method](#) on a speech synthesizer instance to speak that utterance.

The speech synthesizer maintains a queue of utterances to be spoken. If the synthesizer is not currently speaking, calling [speak\(\\_:\) begins speaking that utterance immediately \(or begin waiting through its \[preUtteranceDelay\]\(#\) if one is set\).](#) If the synthesizer is speaking, utterances are added to a queue and spoken in the order they are received.

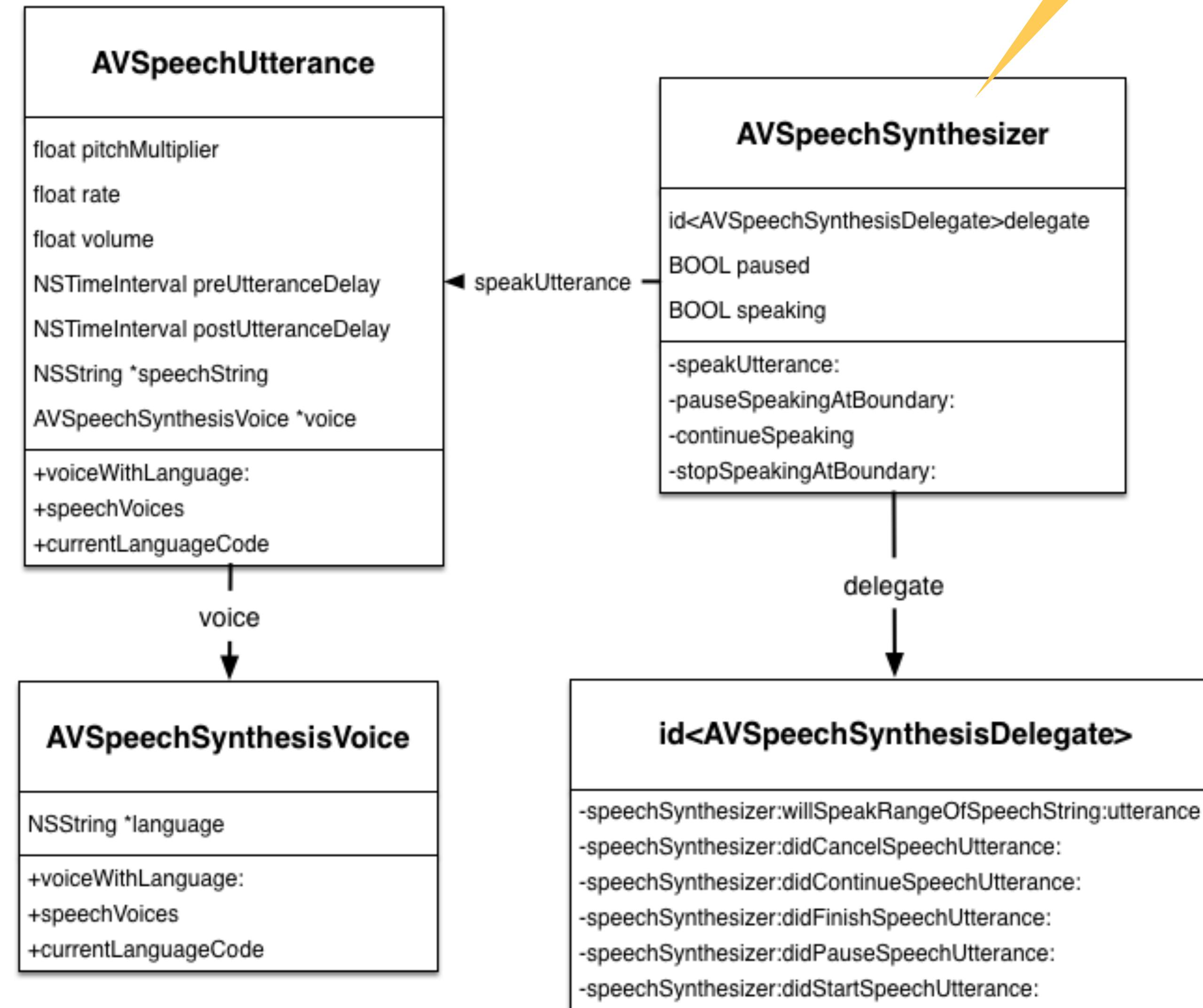
After speech has begun, you can use the synthesizer object to pause or stop speech. After speech is paused, it may be continued from the point at which it left off; stopping ends speech entirely, removing any utterances yet to be spoken from the synthesizer's queue.

You may monitor the speech synthesizer by examining its speaking and paused properties, or by setting a delegate. Messages in the [AVSpeechSynthesizerDelegate](#) protocol are sent as significant events occur during speech synthesis.

### Symbols

# SPEECH SYNTHESIS

Speech Synthesis API



# SPEECH SYNTHESIS

SUBTITLE

- Create instance of `AVSpeechSynthesis`
- Create `AVSpeechUtterance` for each section of text to be spoken

```
let synthesizer = AVSpeechSynthesizer()  
synthesizer.stopSpeaking(at: .word)  
let voice = AVSpeechSynthesisVoice(language: "en_US")  
let utterance = AVSpeechUtterance(string: string)  
utterance.voice = voice  
synthesizer.speak(utterance)
```

Wait until word is spoken before pausing



# SPEECH SYNTHESIS

```
let string = "One fish Two fish Red fish Blue fish. Black fish Blue fish Old fish New; fish.  
This one has a little star. This one has a little car. Say! What a lot of fish there are;"
```

```
let synthesizer = AVSpeechSynthesizer()  
synthesizer.stopSpeaking(at: .word)  
let voice = AVSpeechSynthesisVoice(language: "en_US")  
let utterance = AVSpeechUtterance(string: string)  
utterance.voice = voice  
utterance.pitchMultiplier = 1.0  
utterance.rate = 1.0  
synthesizer.speak(utterance)
```

Variations to the  
spoken text



# SPEECH SYNTHESIS

## SUBTITLE

- AVSpeech Utterance properties
  - Properties are meant to better emulate and customize the spoken text
  - Example:
    - Different utterances can be varied for 'emphasis' in reading
    - Pause for dramatic effect

var `pitchMultiplier`: Float

The baseline pitch at which the utterance will be spoken.

var `postUtteranceDelay`: TimeInterval

The amount of time a speech synthesizer will wait after the utterance has finished playing before handling the next queued utterance.

var `preUtteranceDelay`: TimeInterval

The amount of time a speech synthesizer will wait before actual utterance begins.

var `rate`: Double

The rate at which the utterance is spoken.

var `text`: String

Sometimes useful to add punctuation, capitalize or misspell to get desired effect

var `voice`: AVSpeechSynthesisVoice

The voice used for the utterance.

var `volume`: Float

The volume used when speaking the utterance.

# SPEECH SYNTHESIS

## SUBTITLE

- Available voices

- List them using  
[AVSpeechSynthesisVoice  
speechVoices]

Arabic (Saudi Arabia) - ar-SA  
Chinese (China) - zh-CN  
Chinese (Hong Kong SAR China) - zh-HK  
Chinese (Taiwan) - zh-TW  
Czech (Czech Republic) - cs-CZ  
Danish (Denmark) - da-DK  
Dutch (Belgium) - nl-BE  
Dutch (Netherlands) - nl-NL  
English (Australia) - en-AU  
English (Ireland) - en-IE  
English (South Africa) - en-ZA  
English (United Kingdom) - en-GB  
English (United States) - en-US  
Finnish (Finland) - fi-FI  
French (Canada) - fr-CA  
French (France) - fr-FR  
German (Germany) - de-DE  
Greek (Greece) - el-GR  
Hindi (India) - hi-IN  
Hungarian (Hungary) - hu-HU  
Indonesian (Indonesia) - id-ID  
Italian (Italy) - it-IT  
Japanese (Japan) - ja-JP  
Korean (South Korea) - ko-KR  
Norwegian (Norway) - no-NO  
Polish (Poland) - pl-PL  
Portuguese (Brazil) - pt-BR  
Portuguese (Portugal) - pt-PT

# SPEECH SYNTHESIS

```
let string = "あのイーハトーヴォのすきとおった風、夏でも底に冷たさをもつ青いそら、うつくしい森で  
飾られたモーリオ市、郊外のぎらぎらひかる草の波。またそのなかでいっしょになつたたくさんのひとた  
ち、ファゼー口とロザー口、羊飼のミー口や、顔の赤いこどもたち、地主のテーモ、山猫博士のボーガン  
ト・テストゥパーゴなど、いまこの暗い巨きな石の建物のなかで考えていると、みんなむかし風のなつか  
しい青い幻燈のように思われます。"
```

```
let synthesizer = AVSpeechSynthesizer()  
let voice = AVSpeechSynthesisVoice(language: "ja_JP")  
let utterance = AVSpeechUtterance(string: string)  
utterance.voice = voice  
synthesizer.speak(utterance)
```



# SPEECH SYNTHESIS

- Does not translate into languages
  - You need to provide the correctly written text in a language
  - Sometimes different languages sound better
    - Australian English

Arabic (Saudi Arabia) - ar-SA  
Chinese (China) - zh-CN  
Chinese (Hong Kong SAR China) - zh-HK  
Chinese (Taiwan) - zh-TW  
Czech (Czech Republic) - cs-CZ  
Danish (Denmark) - da-DK  
Dutch (Belgium) - nl-BE  
Dutch (Netherlands) - nl-NL  
English (Australia) - en-AU  
English (Ireland) - en-IE  
English (South Africa) - en-ZA  
English (United Kingdom) - en-GB  
English (United States) - en-US  
Finnish (Finland) - fi-FI  
French (Canada) - fr-CA  
French (France) - fr-FR  
German (Germany) - de-DE  
Greek (Greece) - el-GR  
Hindi (India) - hi-IN  
Hungarian (Hungary) - hu-HU  
Indonesian (Indonesia) - id-ID  
Italian (Italy) - it-IT  
Japanese (Japan) - ja-JP  
Korean (South Korea) - ko-KR  
Norwegian (Norway) - no-NO  
Polish (Poland) - pl-PL  
Portuguese (Brazil) - pt-BR  
Portuguese (Portugal) - pt-PT  
Romanian (Romania) - ro-RO  
Russian (Russia) - ru-RU  
Slovak (Slovakia) - sk-SK  
Spanish (Mexico) - es-MX  
Spanish (Spain) - es-ES  
Swedish (Sweden) - sv-SE  
Thai (Thailand) - th-TH  
Turkish (Turkey) - tr-TR

# SPEECH SYNTHESIS

- AVSpeechSynthesisDelegate Protocol
  - Defines methods that the delegate of an AVSpeechSynthesizer object may implement
  - All methods in this protocol are optional
  - Respond to events that occur during speech synthesis

## Responding to Speech Synthesis Events

Useful for kids with short attention spans

```
func speechSynthesizer(AVSpeechSynthesizer, didCancel: AVSpeechUtterance)
```

Tells the delegate when the synthesizer has canceled speaking an utterance.

```
func speechSynthesizer(AVSpeechSynthesizer, didContinue: AVSpeechUtterance)
```

Tells the delegate when the synthesizer has resumed speaking an utterance after being paused.

```
func speechSynthesizer(AVSpeechSynthesizer, didFinish: AVSpeechUtterance)
```

Tells the delegate when the synthesizer has finished speaking an utterance.

```
func speechSynthesizer(AVSpeechSynthesizer, didPause: AVSpeechUtterance)
```

Tells the delegate when the synthesizer has paused while speaking an utterance.

```
func speechSynthesizer(AVSpeechSynthesizer, didStart: AVSpeechUtterance)
```

Tells the delegate when the synthesizer has begun speaking an utterance.

```
func speechSynthesizer(AVSpeechSynthesizer, willSpeakRangeOfSpeechString: NSRange, utterance: AVSpeechUtterance)
```

Tells the delegate when the synthesizer is about to speak a portion of an utterance's text.

Identify what is being spoken

# SPEECH SYNTHESIS

- Identify start of speaking
- Identify range of spoken characters

```
let string = "One fish Two fish Red fish Blue fish. Black fish Blue fish Old fish New; fish.  
This one has a little star. This one has a little car. Say! What a lot of fish there are."  
  
class PlaygroundViewController: UIViewController {  
    let synthesizer = AVSpeechSynthesizer()  
  
    override func viewDidLoad() {  
  
        super.viewDidLoad()  
  
        synthesizer.delegate = self  
        let voice = AVSpeechSynthesisVoice(language: "en_US")  
        let utterance = AVSpeechUtterance(string: string)  
        utterance.voice = voice  
        utterance.pitchMultiplier = 1.0  
        utterance.rate = 0.25  
        synthesizer.speak(utterance)  
    }  
}  
  
extension PlaygroundViewController: AVSpeechSynthesizerDelegate {  
  
    func speechSynthesizer(_ synthesizer: AVSpeechSynthesizer,  
                          didStart utterance: AVSpeechUtterance) {  
        print("Started")  
    }  
  
    func speechSynthesizer(_ synthesizer: AVSpeechSynthesizer,  
                          willSpeakRangeOfSpeechString characterRange: NSRange,  
                          utterance: AVSpeechUtterance) {  
        print("▶: \(characterRange.location) - \(characterRange.length+characterRange.location)")  
    }  
}  
  
// Create an instance of the view controller  
let viewController = PlaygroundViewController()
```

# SPEECH SYNTHESIS

- Can get errors when running on simulator (not on device)
- Does not effect anything
- Open radar

```
AVAudioSessionUtilities.h:88: GetProperty_DefaultToZero: AudioSessionGetProperty  
('disa') failed with error: '?ytp'  
2014-04-07 10:29:20.932 2014-Spring-ReadToMe[98703:3f03] 10:29:20.932 ERROR:  
AVAudioSessionUtilities.h:88: GetProperty_DefaultToZero: AudioSessionGetProperty  
('disa') failed with error: '?ytp'
```

?

# SPEECH SYNTHESIS

```
- (NSString*)detectLanguage:(NSString*)string
{
    NSArray *tagschemes = [NSArray arrayWithObjects:NLTagSchemeLanguage, nil];
    NSLinguisticTagger *tagger = [[NSLinguisticTagger alloc] initWithTagSchemes:tagschemes options:0];
    [tagger setString:string];
    NSString *language = [tagger tagAtIndex:0 scheme:NLTagSchemeLanguage
                                         tokenRange:NULL sentenceRange:NULL];
    return language;
}
```

- Detect text language from text

# PRE-RECORDED AUDIO



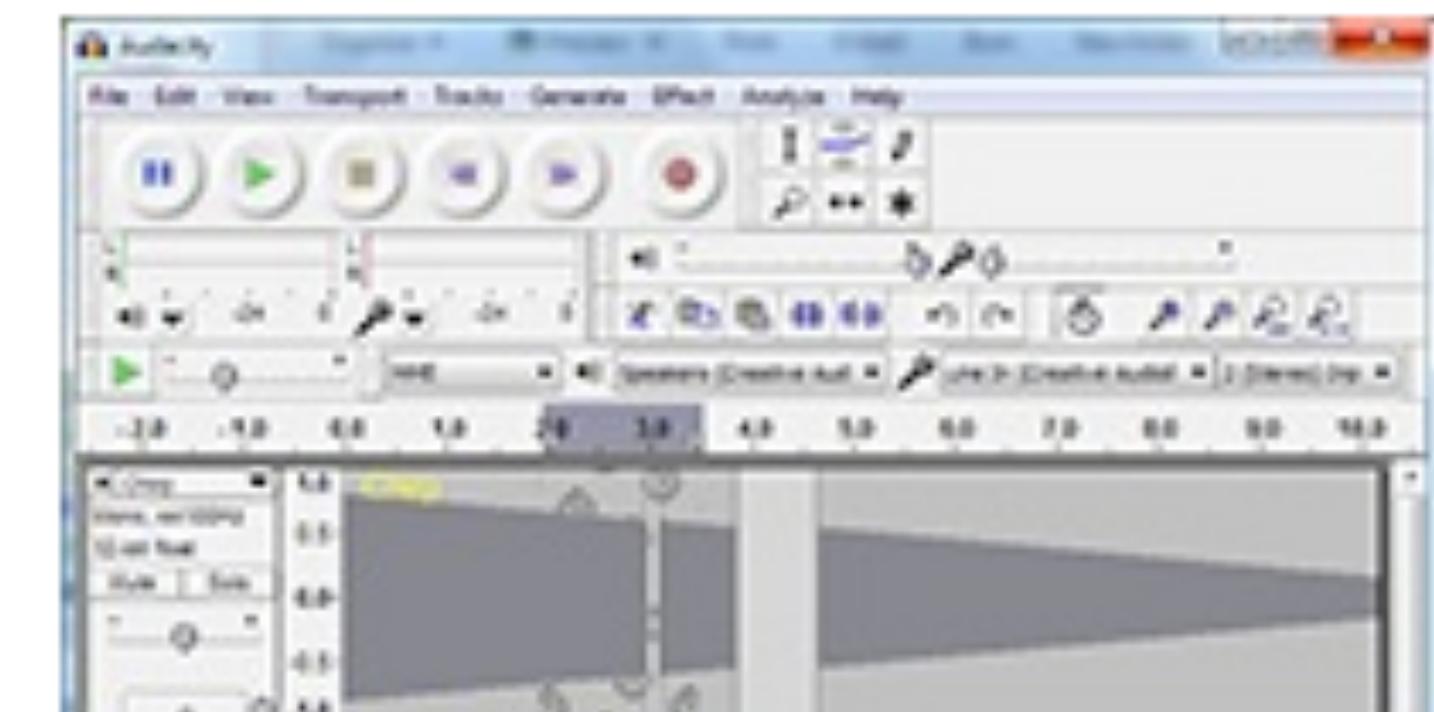
# AUDACITY



[Home](#) [About](#) [Download](#) [Help](#) [Contact Us](#) [Get Involved](#) [Donate](#)

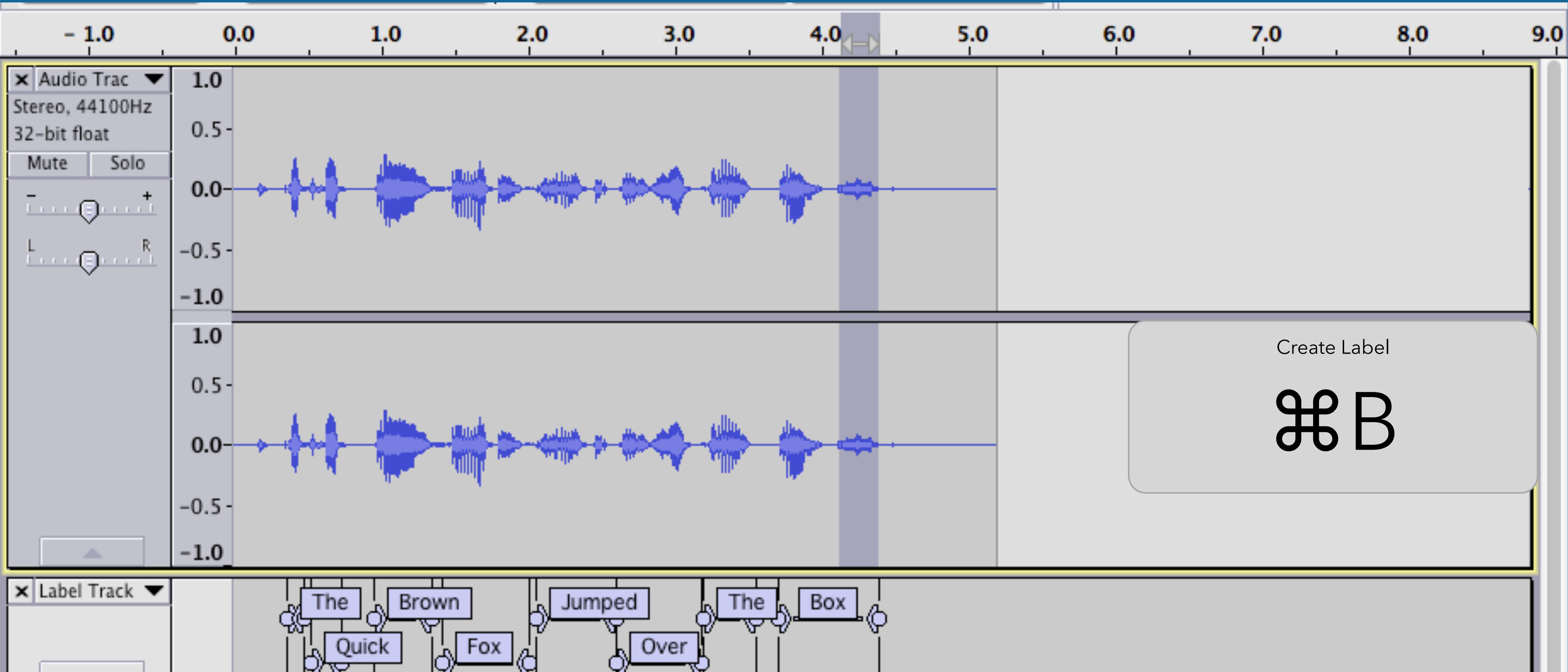
**Audacity® is free, open source, cross-platform software for recording and editing sounds.**

Audacity is available for Windows®, Mac®, GNU/Linux® and other operating systems. Check our [feature list](#), [Wiki](#) and [Forum](#) for more information.

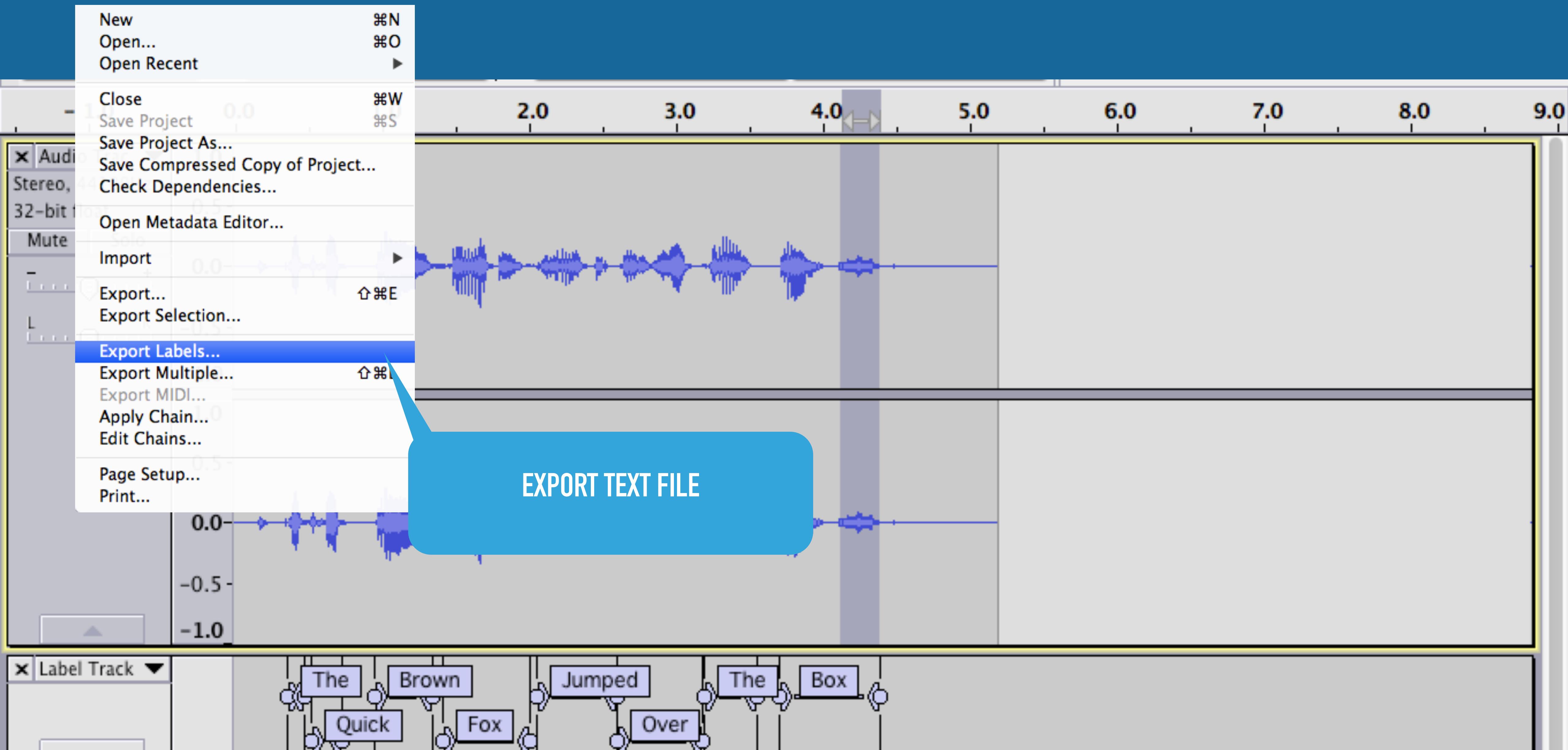


[Download Audacity 2.0.3](#)

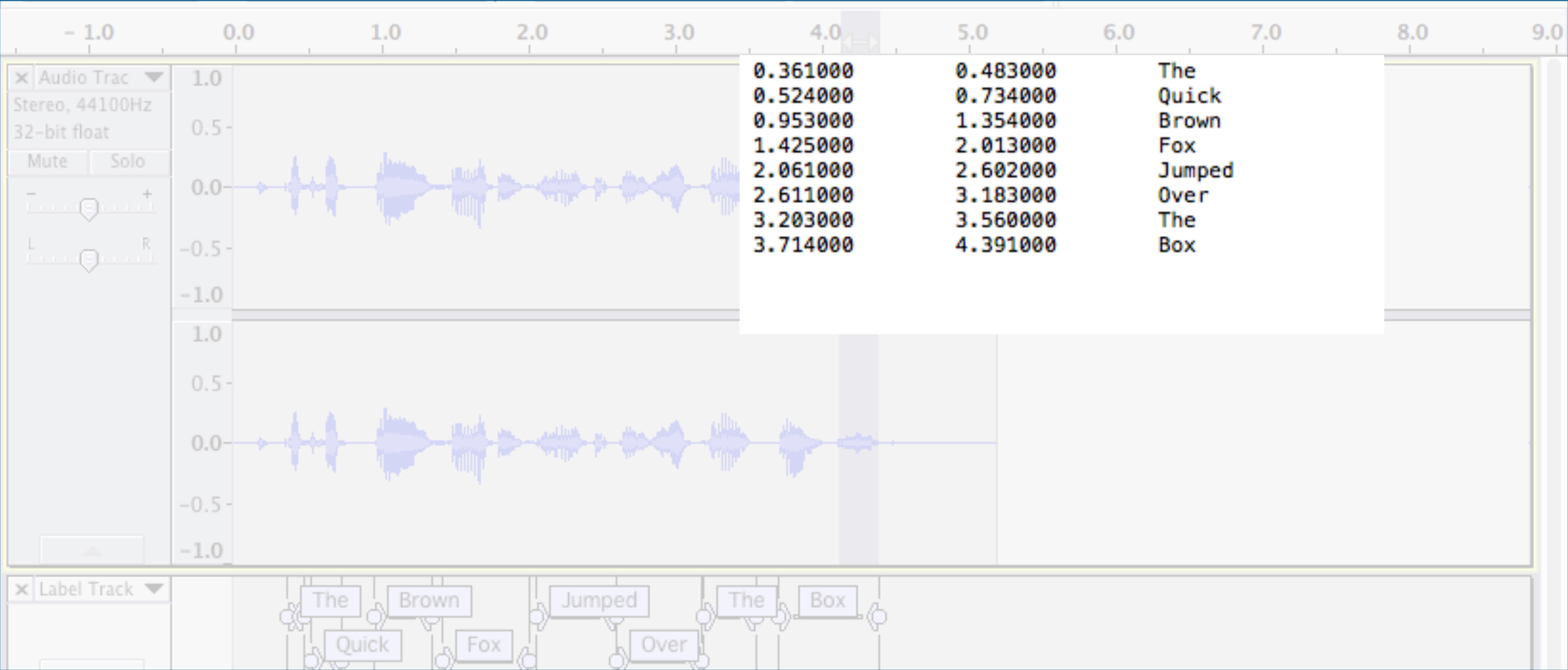
# AUDACITY



# AUDACITY



# AUDACITY



# APPROACHES

- Strategy #1
  - Get start and stop time for each word.
  - Monitor that with a timer and update based on the time
  - Gets out of sync
- Strategy #2
  - Use AVAudioPlayer timer to update the strings
  - Gets out of sync (especially on short words)
- Strategy #3
  - Set a timer for each word
  - Clear all attributes and update based on the timer

# APPROACHES

- NSTimer for each word

```
var startIndex = 0
for i in 0..<timer.count{
    let word = timer[i]["word"] as! String
    let time = timer[i]["start"] as! Double
    let length = word.count
    let wordTimer = Timer.scheduledTimer(timeInterval: time, target: self, selector:
        #selector(updateAttributedString(_)), userInfo: [startIndex, length], repeats: false)
    timerList.append(wordTimer)
    startIndex += length + 1
}
```

- AVFoundation to play audio

```
let audioFileURL = URL(fileURLWithPath: Bundle.main.path(forResource: name, ofType: "mp3")!)

do {
    try audioFile = AVAudioPlayer(contentsOf: audioFileURL as URL, fileTypeHint: nil)
    audioFile.prepareToPlay()
} catch let err as NSError {
    print(err.debugDescription)
}
audioFile.play()
```

# APPROACHES

```
let audioFileURL = URL(fileURLWithPath: Bundle.main.path(forResource: name, ofType: "mp3")!)  
do {  
    try audioFile = AVAudioPlayer(contentsOf: audioFileURL as URL, fileTypeHint: nil)  
    audioFile.prepareToPlay()  
} catch let err as NSError {  
    print(err.debugDescription)  
}  
audioFile.play()
```

Create the audio player

# APPROACHES

Create an individual timer for each word.

```
var startIndex = 0
for i in 0..<timer.count{
    let word = timer[i]["word"] as! String
    let time = timer[i]["start"] as! Double
    let length = word.count
    let wordTimer = Timer.scheduledTimer(timeInterval: time, target: self, selector:
        #selector(updateAttributedString(_:)), userInfo: [startIndex, length], repeats: false)
    timerList.append(wordTimer)
    startIndex += length + 1
}
```

# APPROACHES

```
@objc func updateAttributedString(_ timer: Timer) {  
    let range = timer.userInfo as! [Int]  
  
    attributedString.addAttribute(NSBackgroundColorAttributeName,  
        value: UIColor.clear,  
        range: NSRange(location: 0, length: attributedString.length))  
  
    attributedString.addAttribute(NSForegroundColorAttributeName,  
        value: blue,  
        range: NSRange(location: 0, length: attributedString.length))  
  
    attributedString.addAttribute(NSBackgroundColorAttributeName,  
        value: blue,  
        range: NSRange(location: range[0], length: range[1]))  
  
    attributedString.addAttribute(NSForegroundColorAttributeName,  
        value: UIColor.white,  
        range: NSRange(location: range[0], length: range[1]))  
  
   .textLabel.attributedText = attributedString  
}
```

When the timer fires, update the string

Clear the color, update the color



# ADVANCED iOS APPLICATION DEVELOPMENT

---

MPCS 51032 • SPRING 2020 • SESSION 1E

# SESSION 1F



# ADVANCED iOS APPLICATION DEVELOPMENT

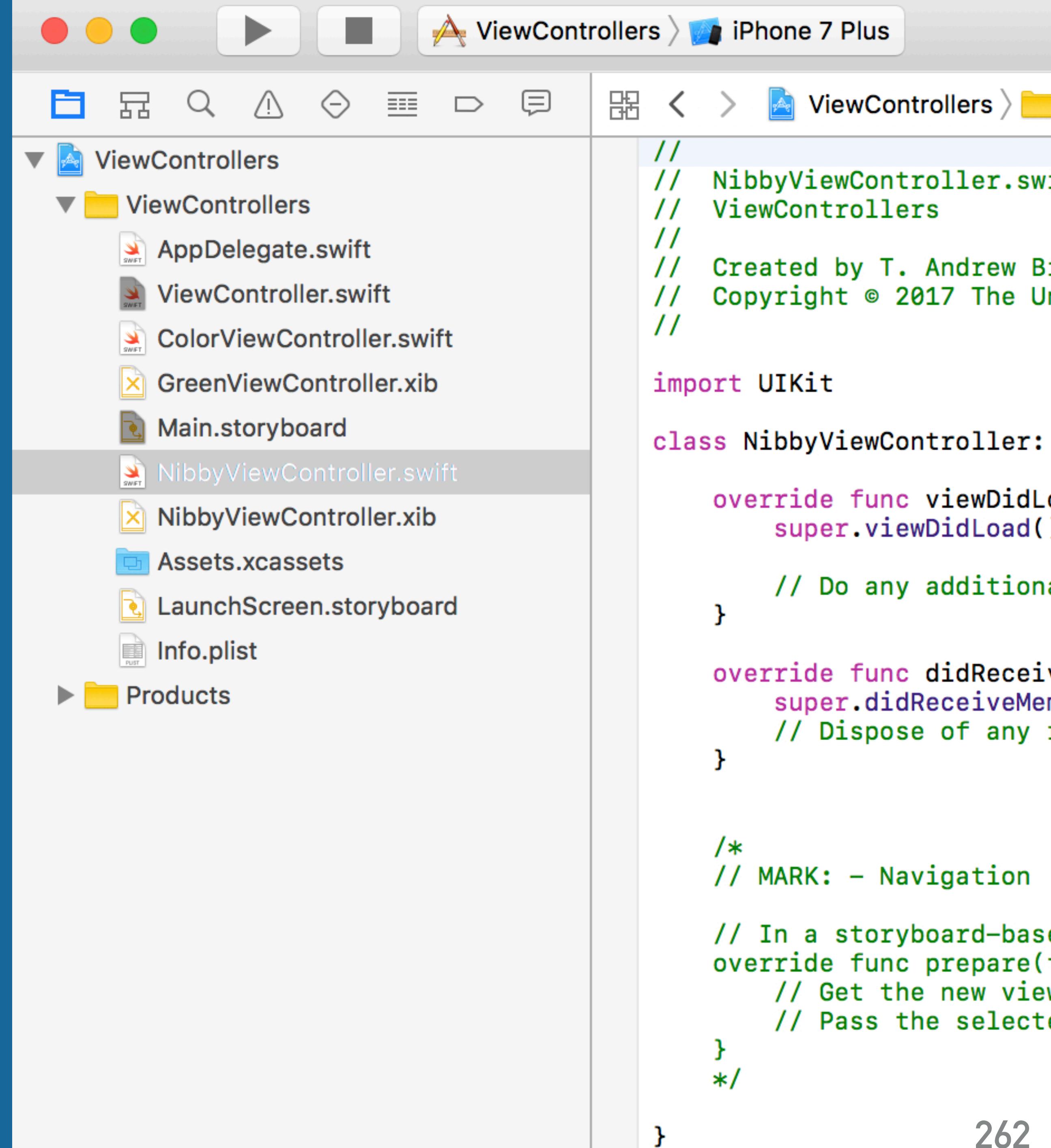
---

MPCS 51032 • SPRING 2020 • SESSION 1F

# NIB BASED VIEWCONTROLLERS

# NIB BASED VIEWCONTROLLERS

- A storyboard is just a collection of NIB files
- You can still use individual NIB files in a project
  - Easier to work with VCS
  - Separation of tasks
  - ...



The screenshot shows the Xcode interface with the following details:

- File Browser:** Shows the project structure under "ViewControllers".
  - AppDelegate.swift
  - ViewController.swift
  - ColorViewController.swift
  - GreenViewController.xib
  - Main.storyboard
  - NibbyViewController.swift** (selected)
  - NibbyViewController.xib
  - Assets.xcassets
  - LaunchScreen.storyboard
  - Info.plist
- Code Editor:** Displays the code for **NibbyViewController.swift**. The code includes imports, class definitions, and overridden methods like `viewDidLoad()` and `didReceiveMemoryWarning()`. It also contains a section for navigation preparation.

```
// NibbyViewController.swift
// ViewControllers
// Created by T. Andrew B...
// Copyright © 2017 The Un...
//
import UIKit

class NibbyViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

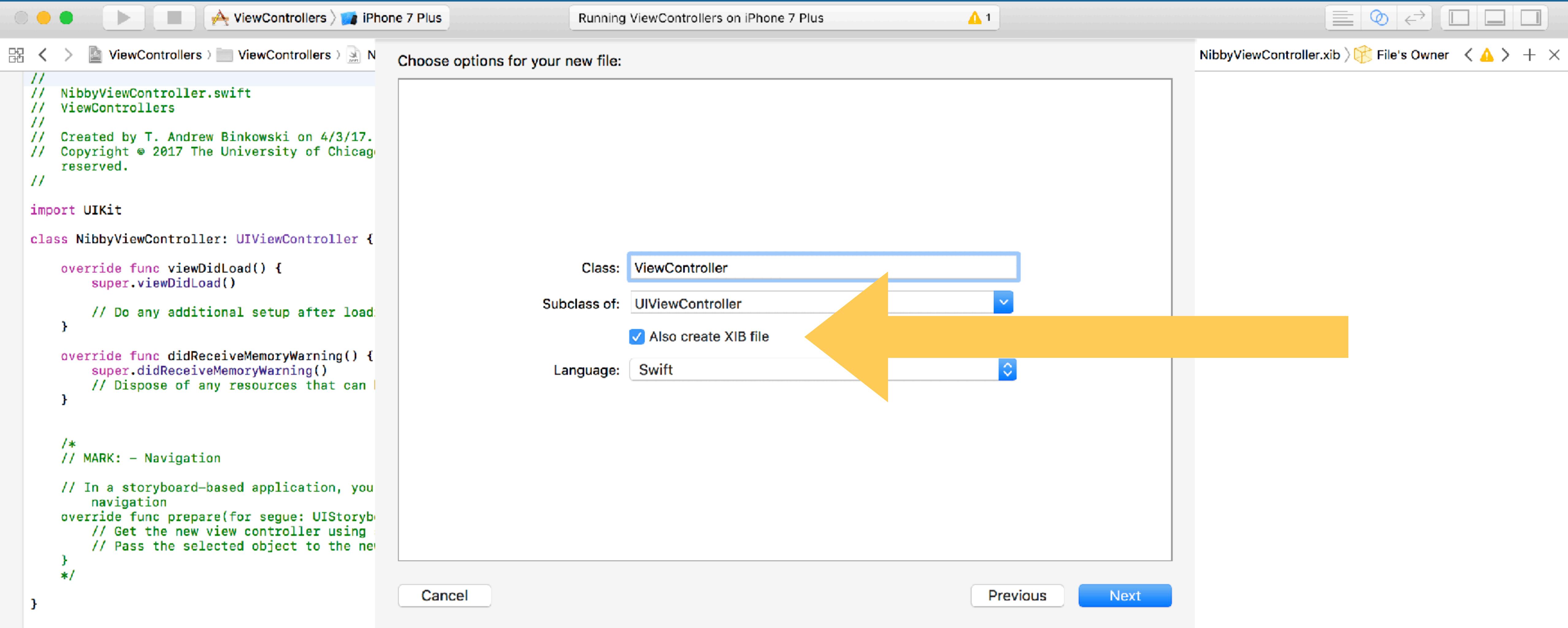
        // Do any additional setup after loading the view.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    /*
    // MARK: - Navigation

    // In a storyboard-based application, you will often want to do a little preparation before navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // Get the new view controller using segue.destination.
        // Pass the selected object to the new view controller.
    }
    */
}
```

# NIB BASED VIEWCONTROLLERS



# NIB BASED VIEWCONTROLLERS

The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure for "2018-session1-nibs". The "PinkViewController.xib" file is selected.
- Preview View:** Displays a pink view controller with the word "Dismiss" centered on it.
- File Navigator:** Shows the file "PinkViewController.swift" is selected.
- Code Editor:** Displays the Swift code for "PinkViewController". The code includes a tap dismissal action.

```
1 //  
2 //  PinkViewController.swift  
3 //  2018-session1-nibs  
4 //  
5 //  Created by T. Andrew Binkowski on 4/1/18.  
6 //  Copyright © 2018 T. Andrew Binkowski. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class PinkViewController: ViewController {  
12  
13    @IBAction func tapDismiss(_ sender: UIButton) {  
14        self.dismiss(animated: true, completion: nil)  
15    }  
16  
17 }  
18
```

- Star Icon:** A yellow star icon is located in the bottom right corner of the slide.

# NIB BASED VIEWCONTROLLERS

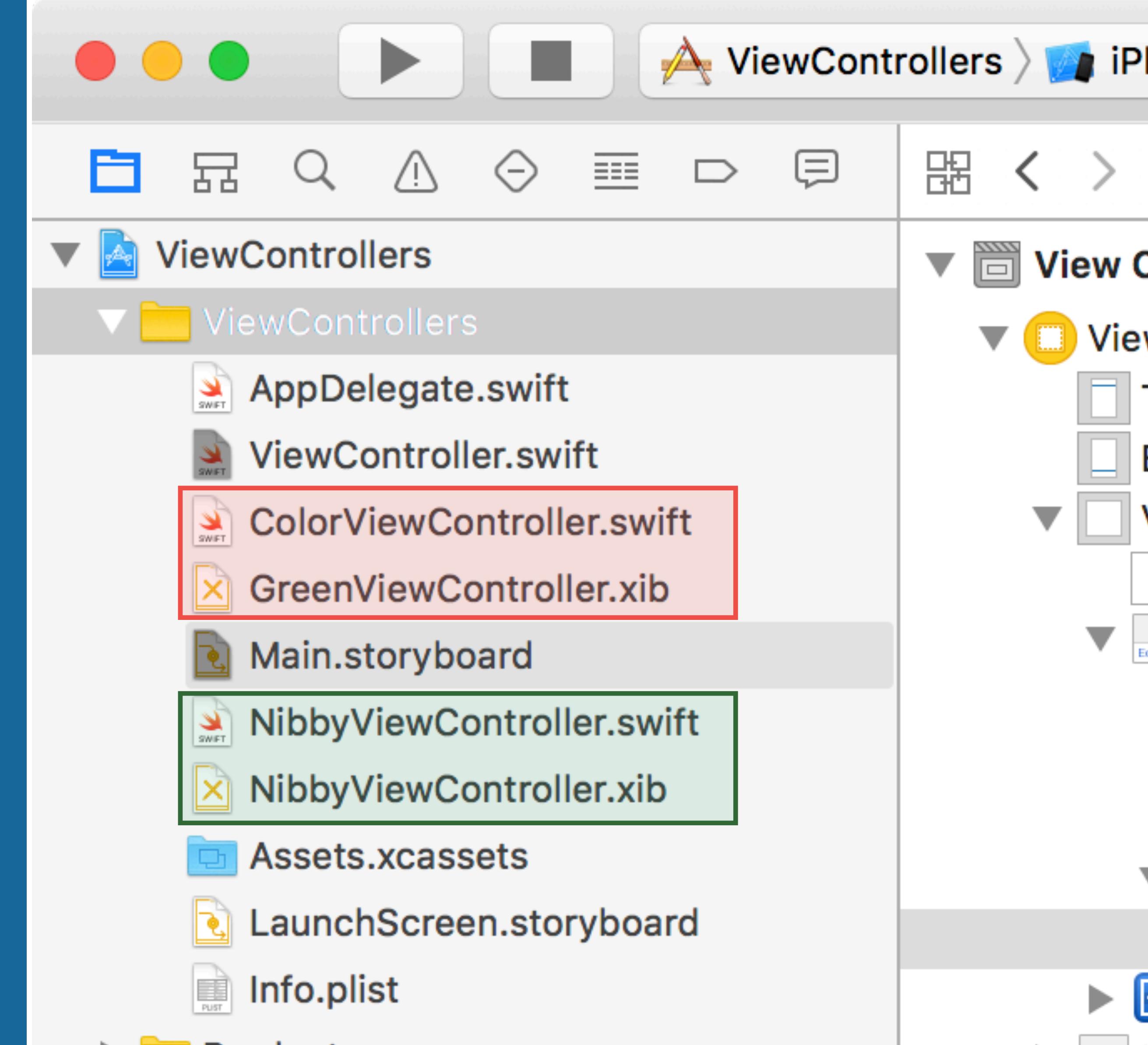
SUBTITLE

- Looks for same named .xib and .swift files

```
let vc = NibbyViewController()
```

- You can change the name, but you will have to reference it

```
let gvc = ColorViewController(nibName: "GreenViewController", bundle: nil)
```



# NIB BASED VIEWCONTROLLERS

```
let vc = NibbyViewController()
```

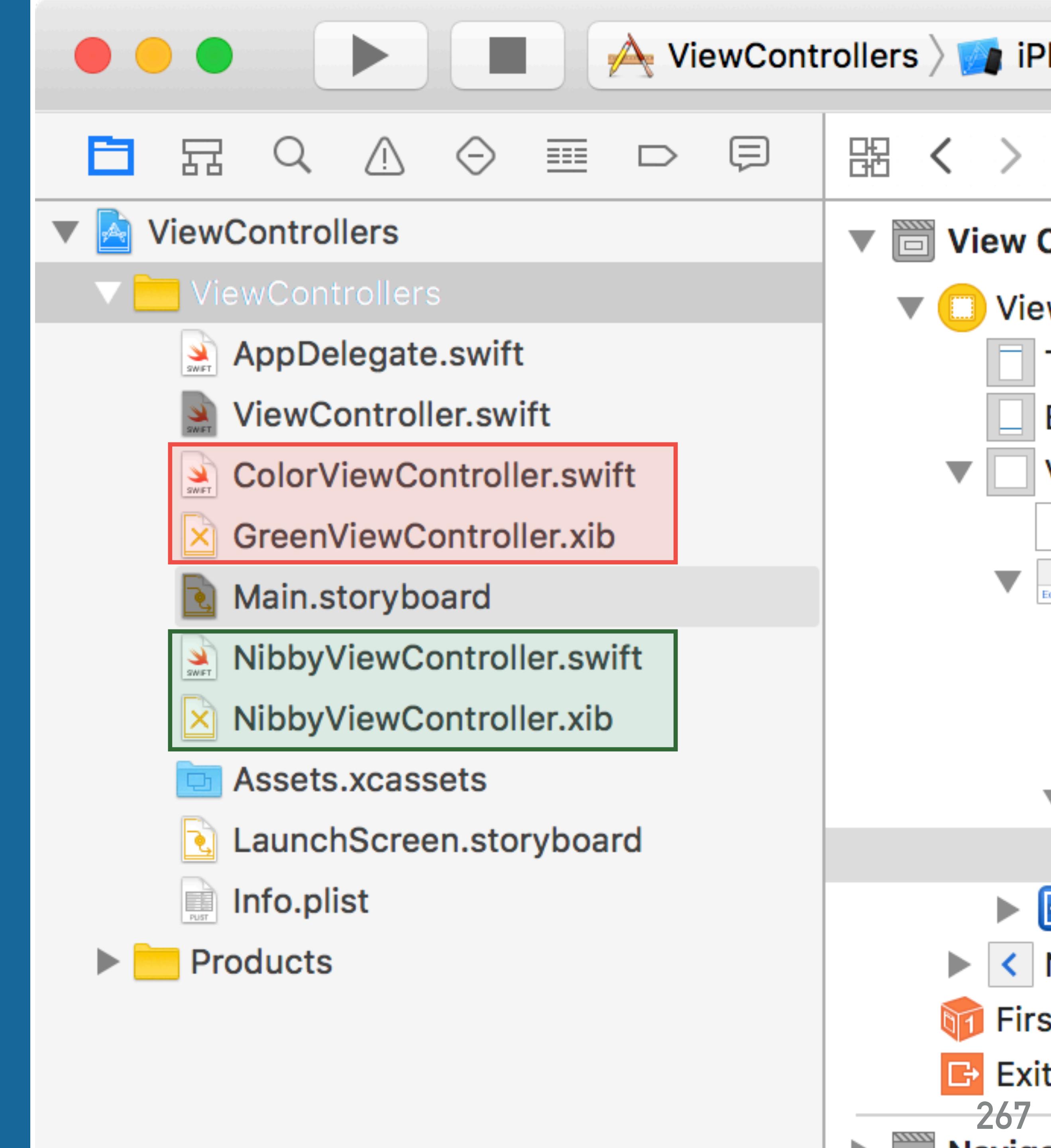
```
let gvc = ColorViewController(nibName: "GreenViewController", bundle: nil)
```

- Use GreenViewController.nib for ColorViewController class

# NIB BASED VIEWCONTROLLERS

SUBTITLE

- Just use it as you would any other view controller
- Pay attention to lifecycle





# ADVANCED iOS APPLICATION DEVELOPMENT

---

MPCS 51032 • SPRING 2020 • SESSION 1F