SESSION 0

WELCOME TO ADVANCED IOS DEVELOPMENT

HOW IS THIS GOING TO WORK?

COURSE LOGISTICS

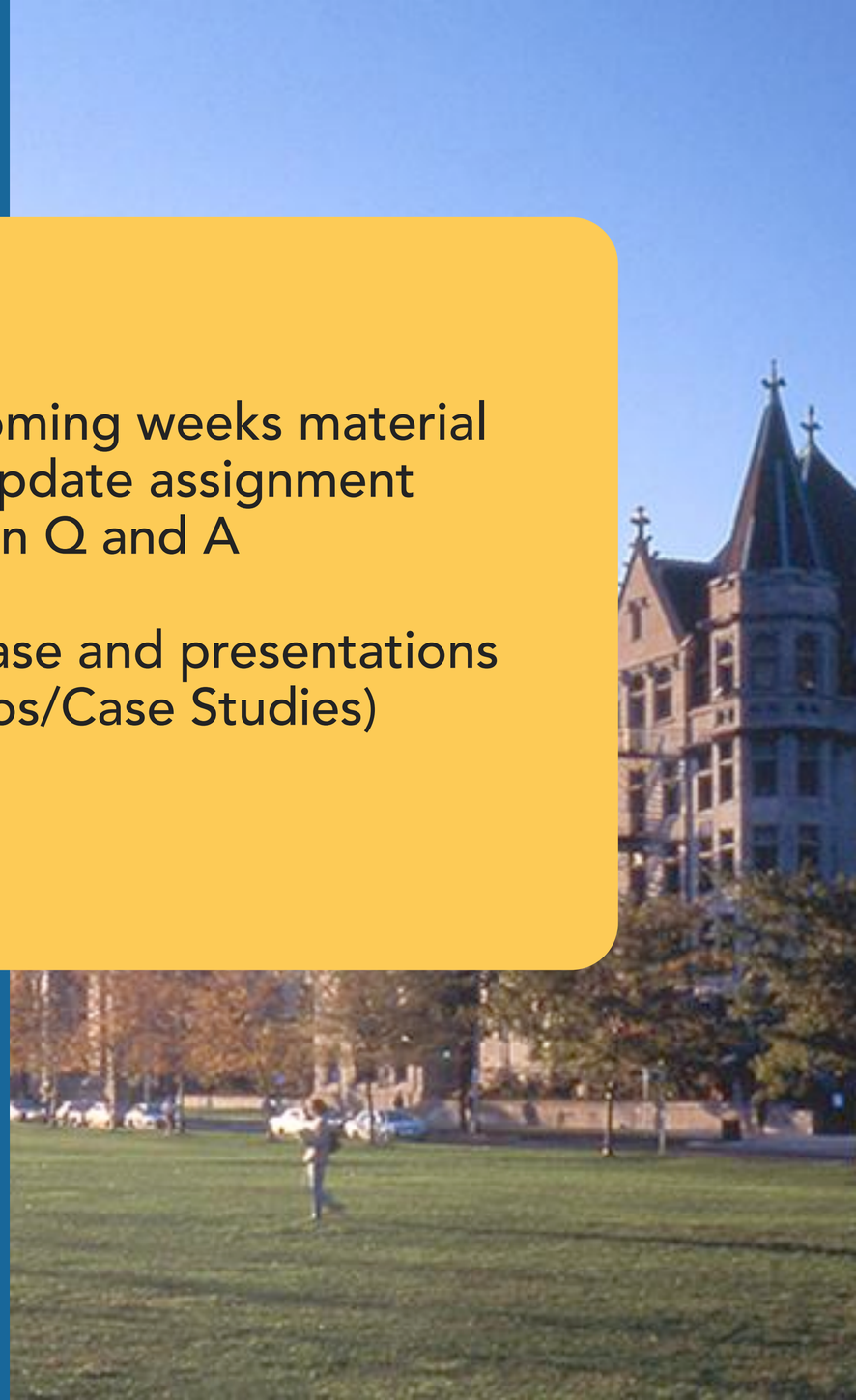
- Flipped Classroom
  - Videos of lectures
  - Meet for discussions and office hours

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)



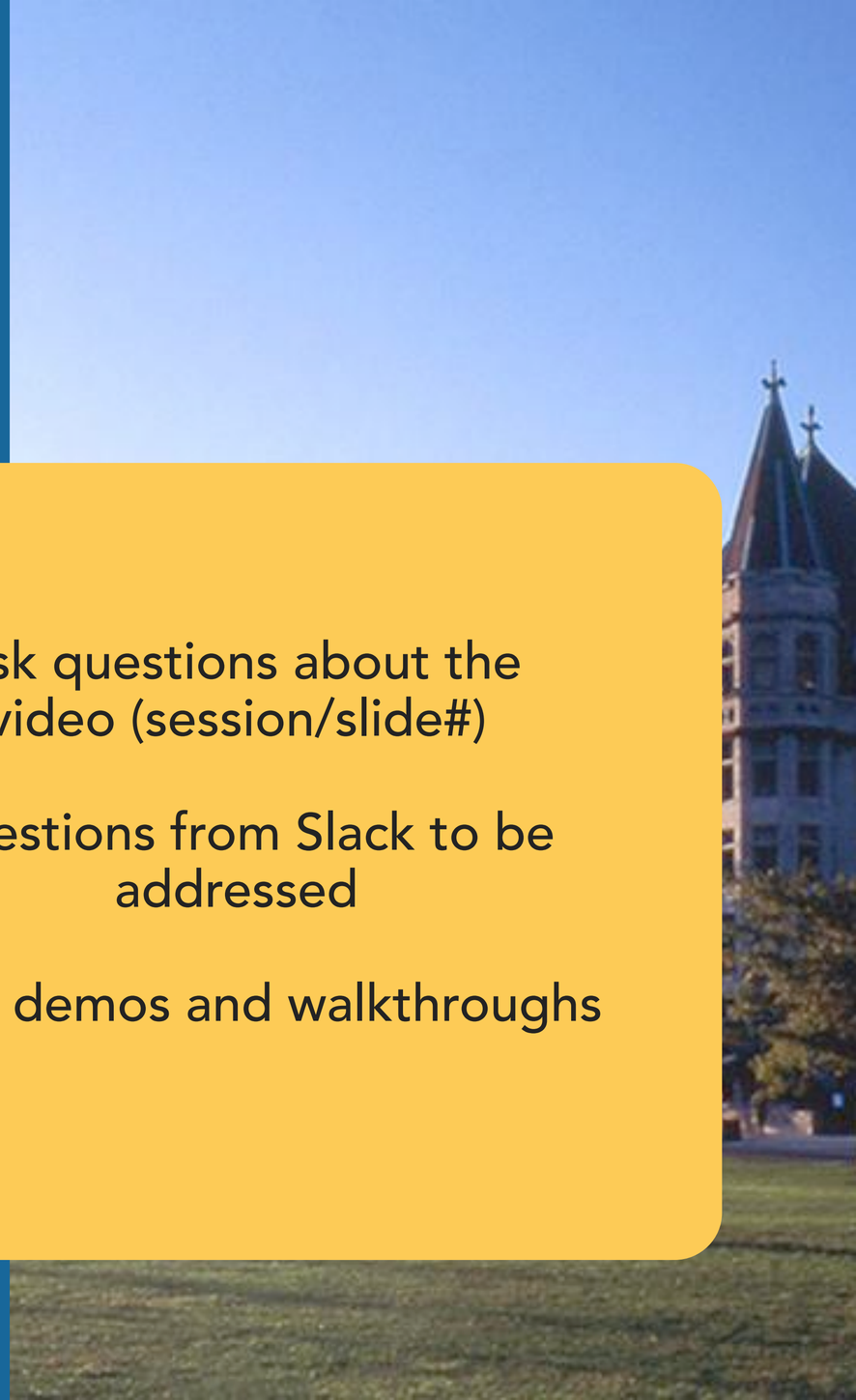
COURSE LOGISTICS

- 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

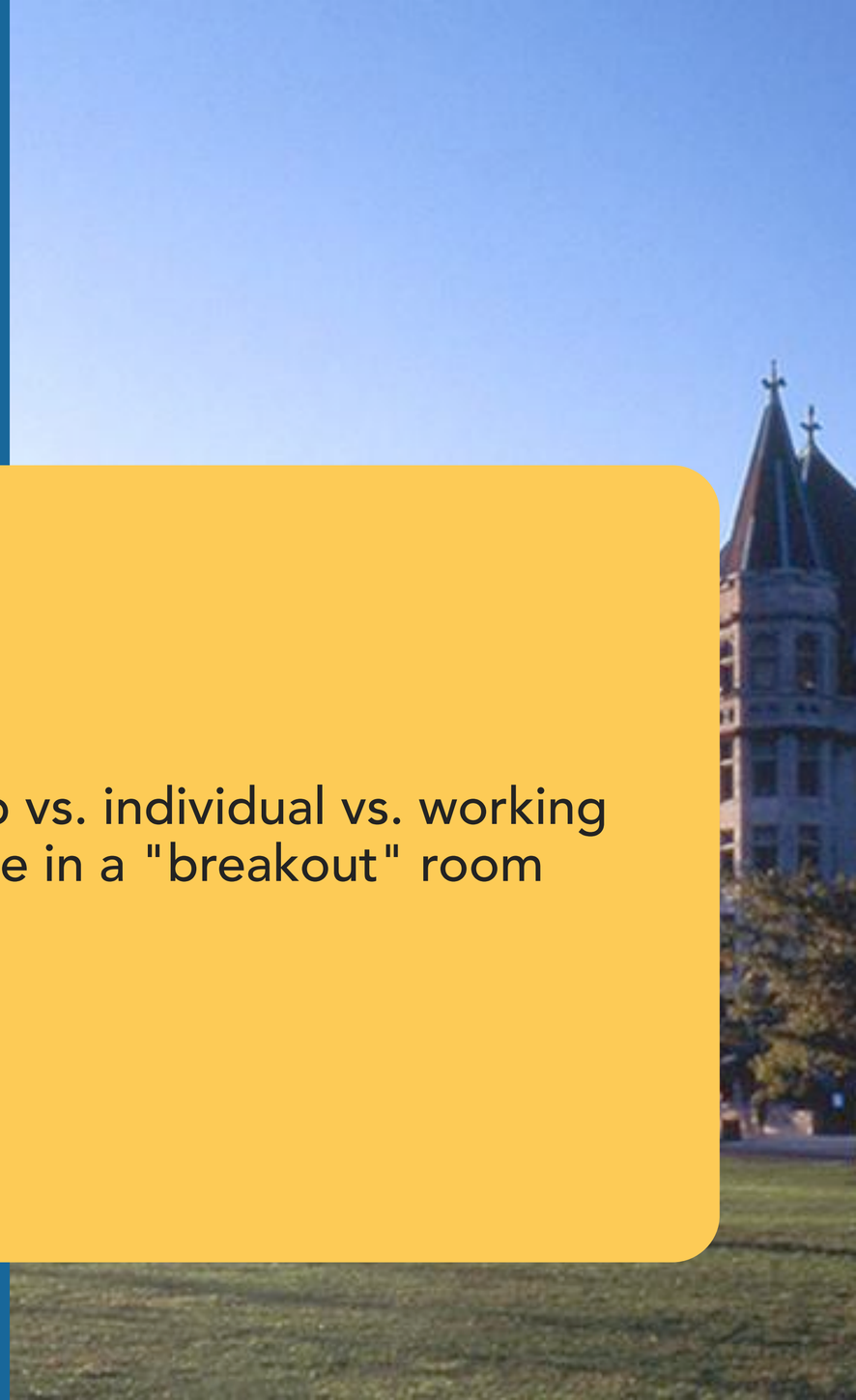
Code demos and walkthroughs



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

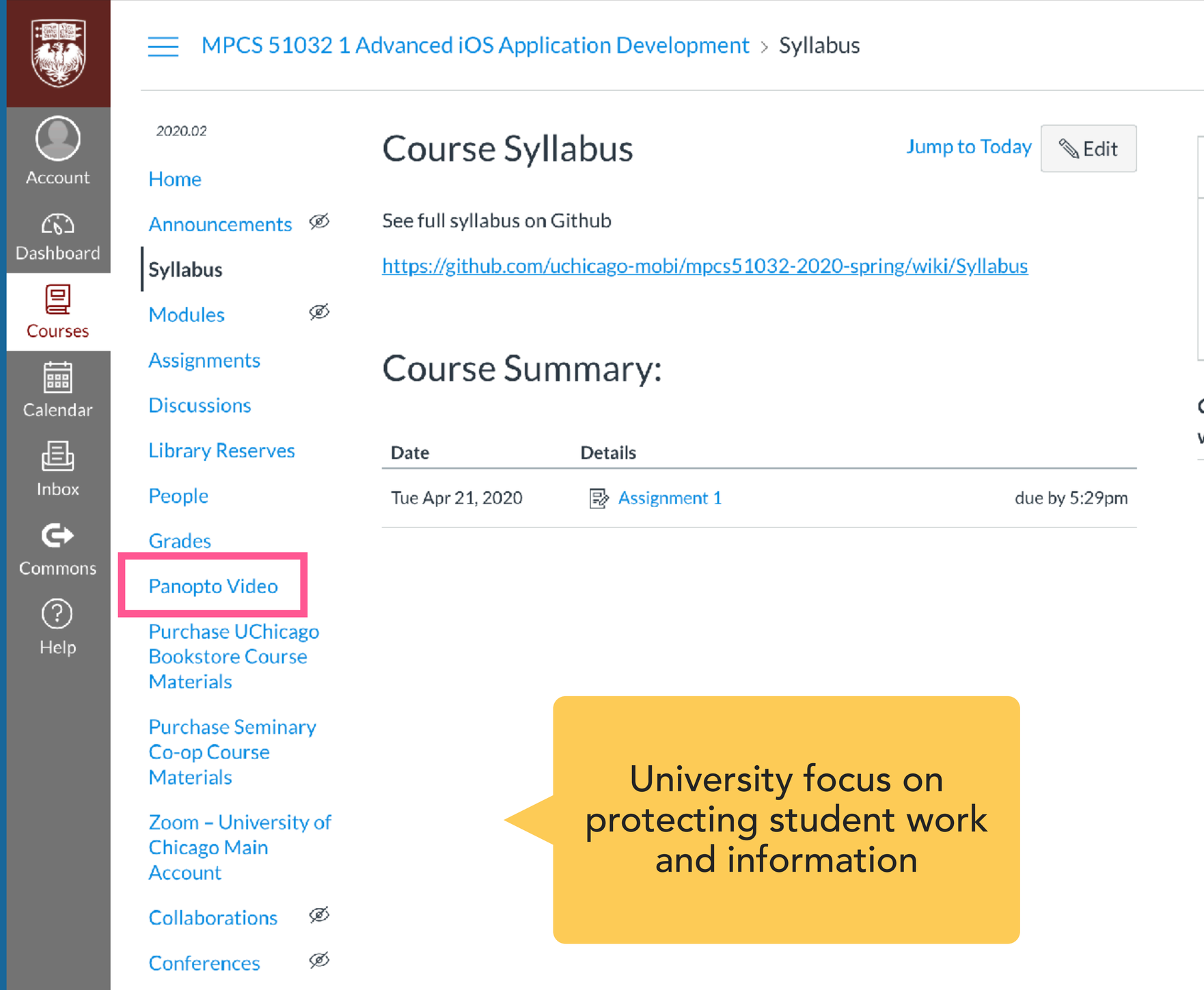
Group vs. individual vs. working time in a "breakout" room



COURSE RESOURCES

COURSE RESOURCES

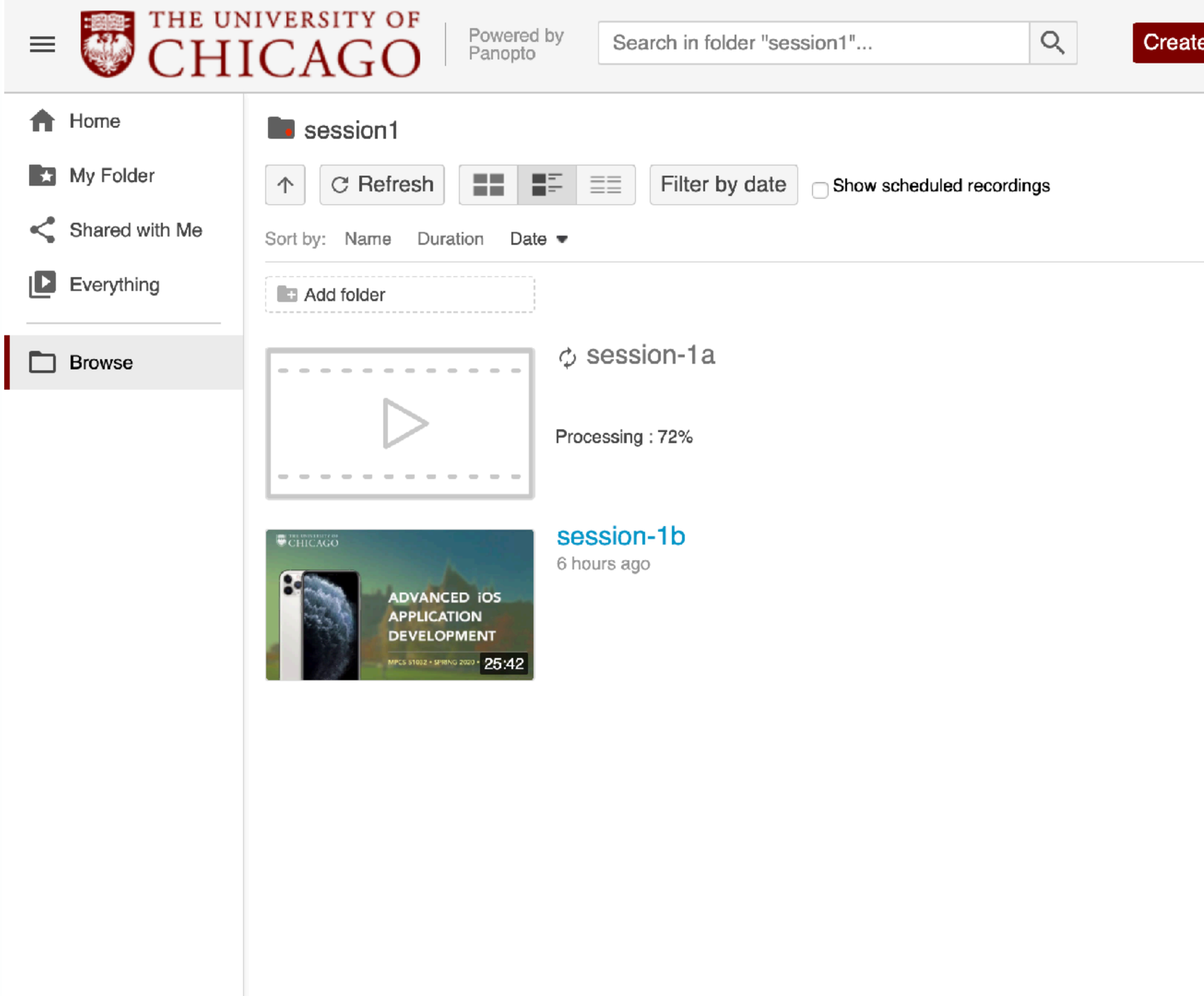
- Canvas "Lite"
- Links to materials
  - Github
  - Panopto



COURSE RESOURCES

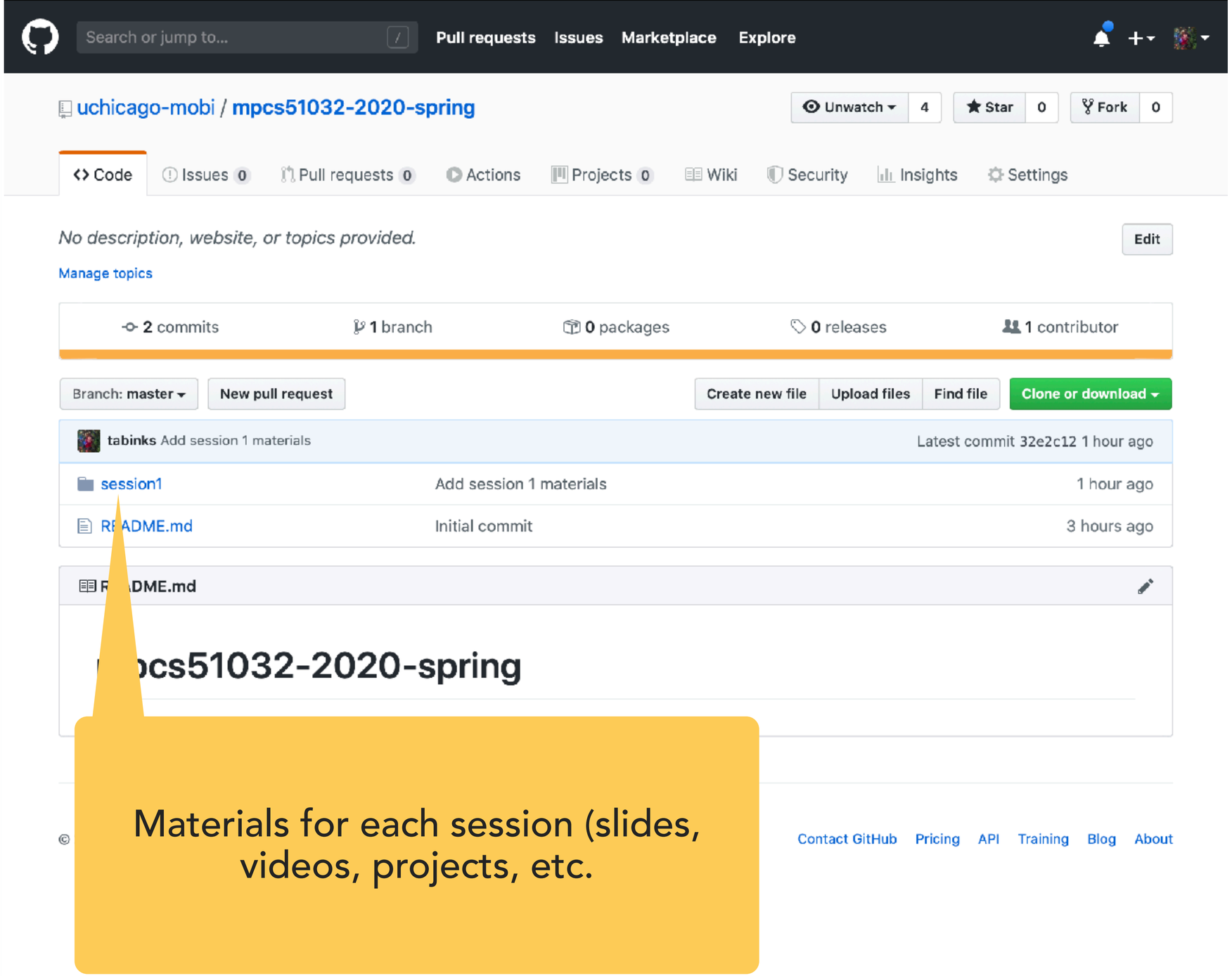
- Panopto video hosting
- Organized by "Sessions"
- Link directly from within canvas

-



COURSE RESOURCES

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



COURSE RESOURCES

Search or jump to... / Pull requests Issues Marketplace Explore

uchicago-mobi / mpcs51032-2020-spring

Unwatch 4 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

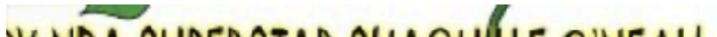
Assignment 1

Edit New Page

tabinks edited this page 2 hours ago · 5 revisions

What do John Travollta, 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?

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



Pages 4

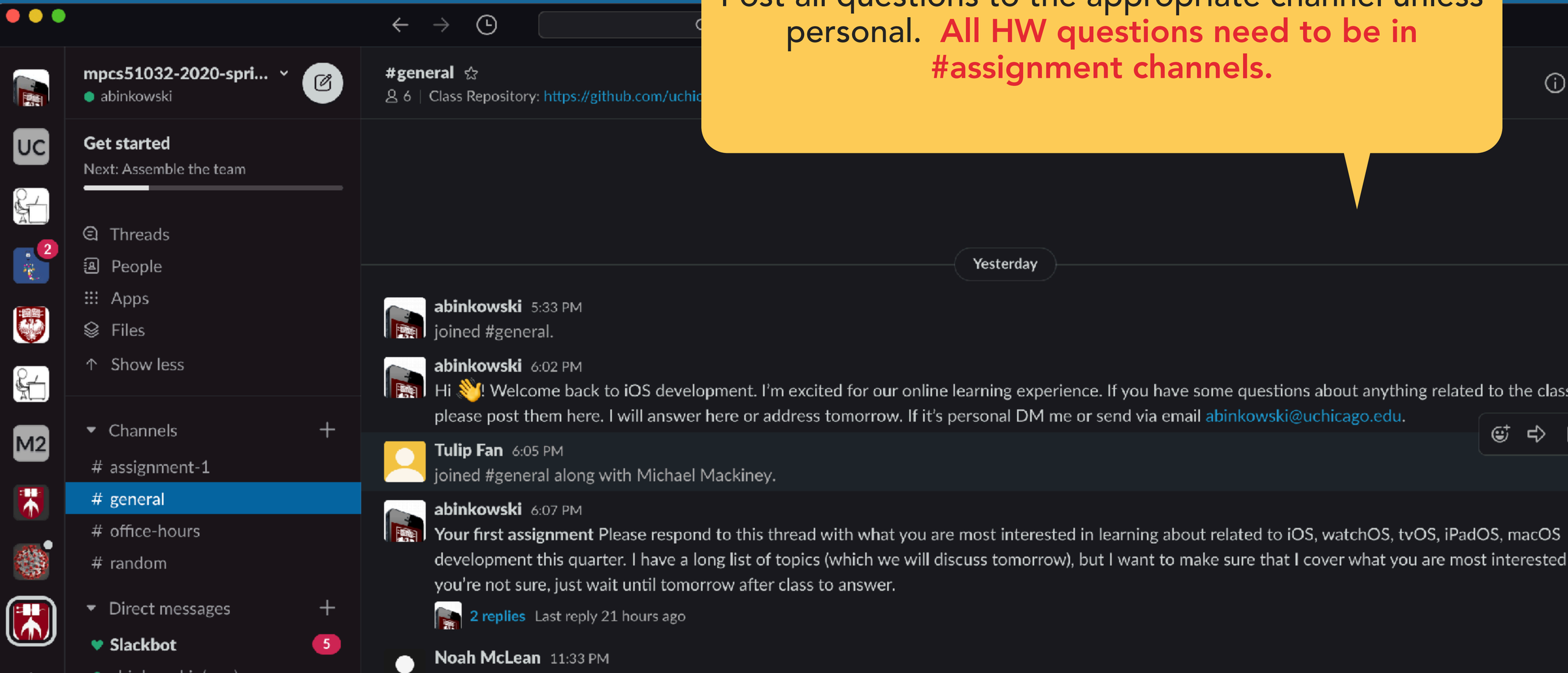
Find a Page...

Home

Assignment 1

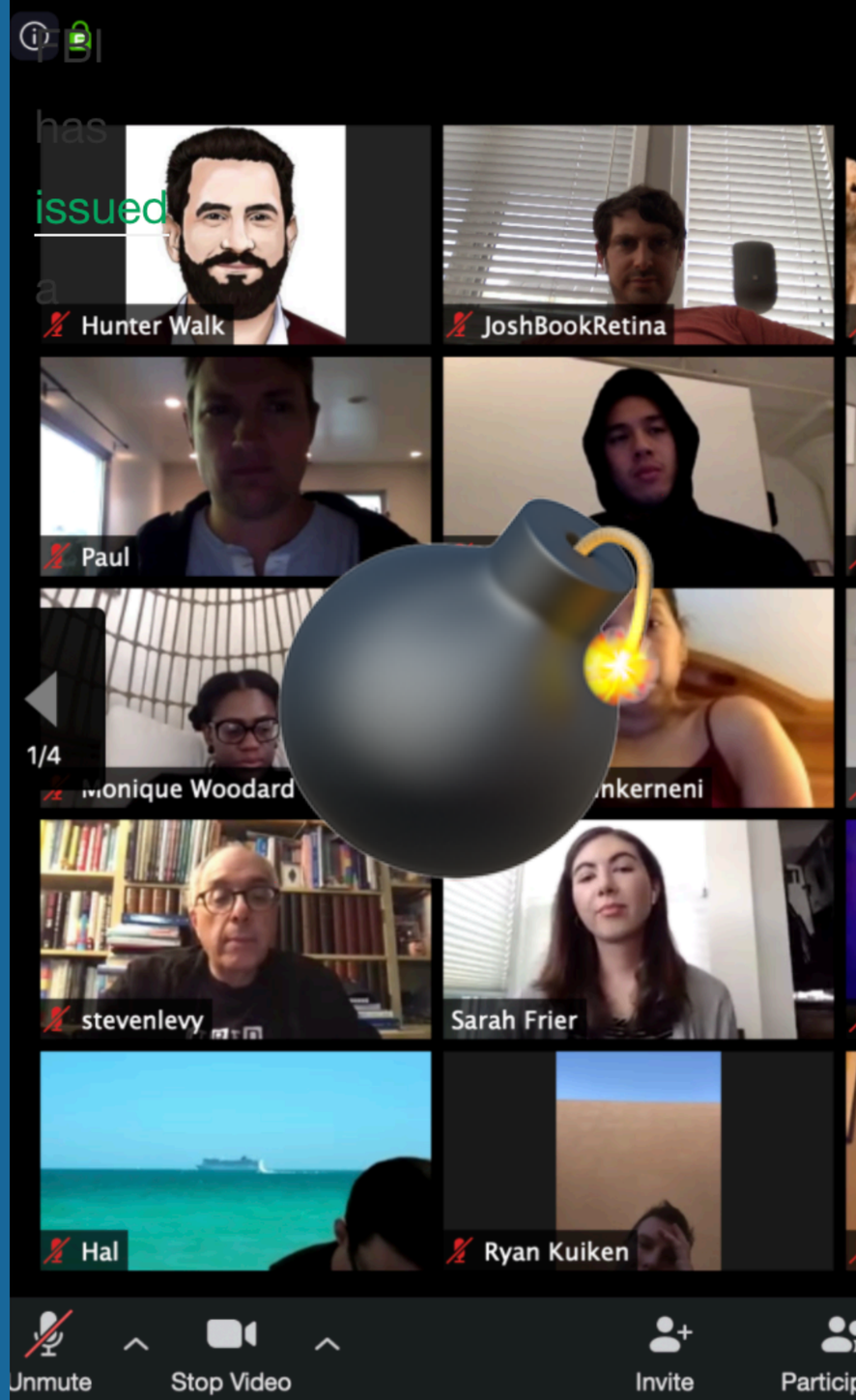
COURSE RESOURCES

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



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

Edit | New Page

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

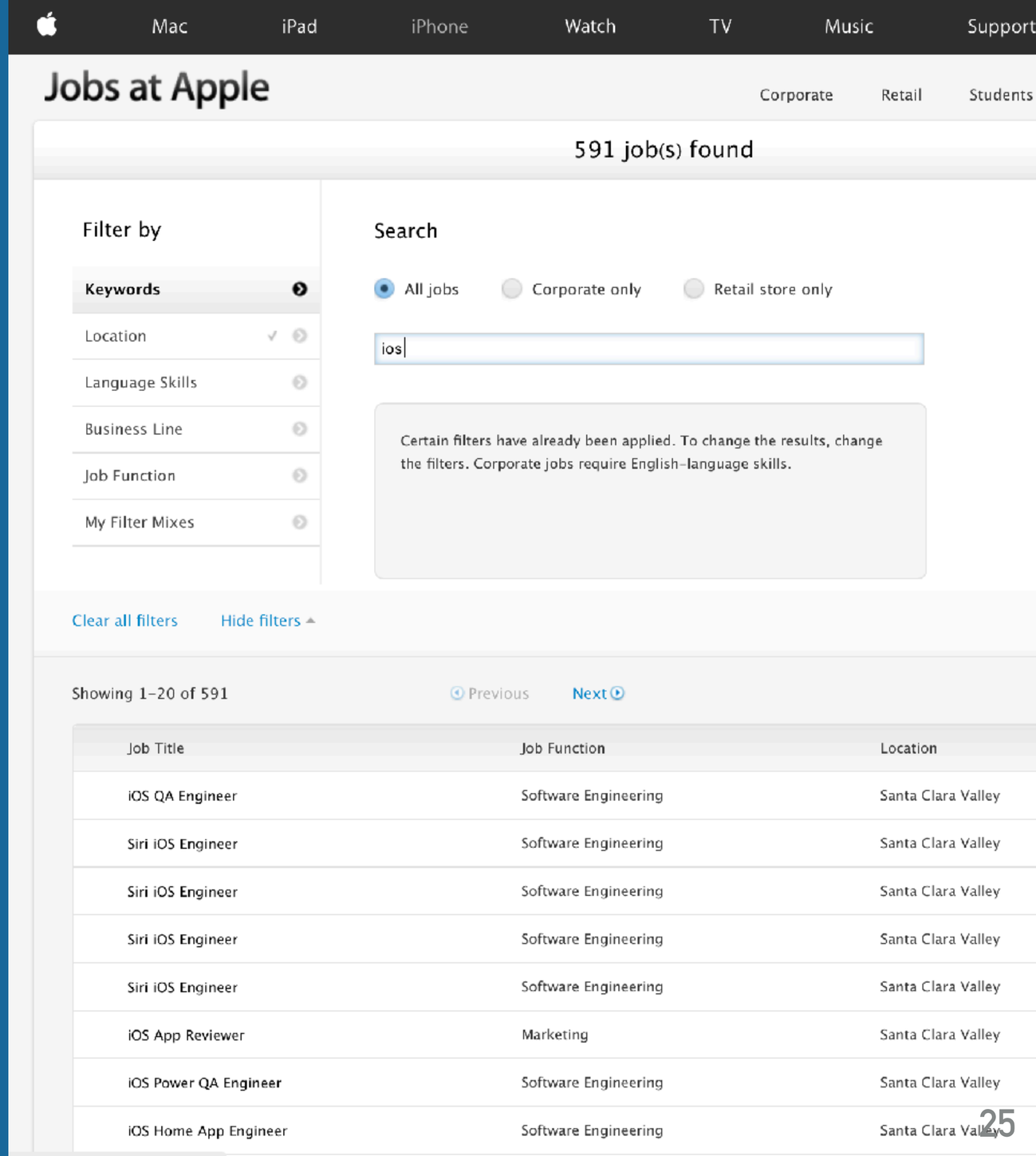
+ Add a custom sidebar

Clone this wiki locally

https://github.com/uch:

COURSE DESIGN

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



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

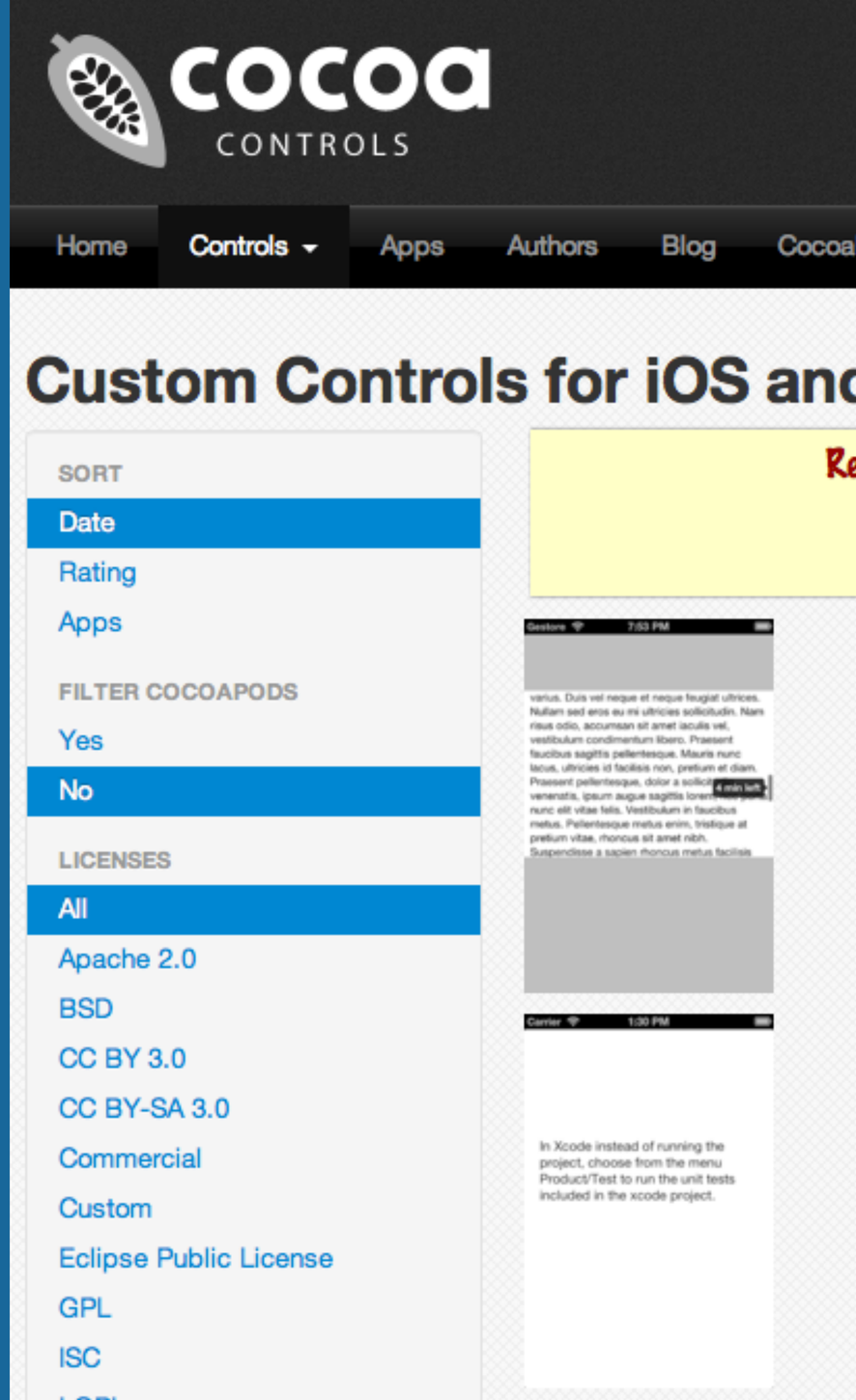
- Third Party Frameworks
  - You may **not** use 3rd party frameworks for homework unless noted or specific permission is granted
  - You may use them for your final project.
    - Please submit them with your proposal.
    - You should be able to comfortably explain what they do.
  - Some third-party 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

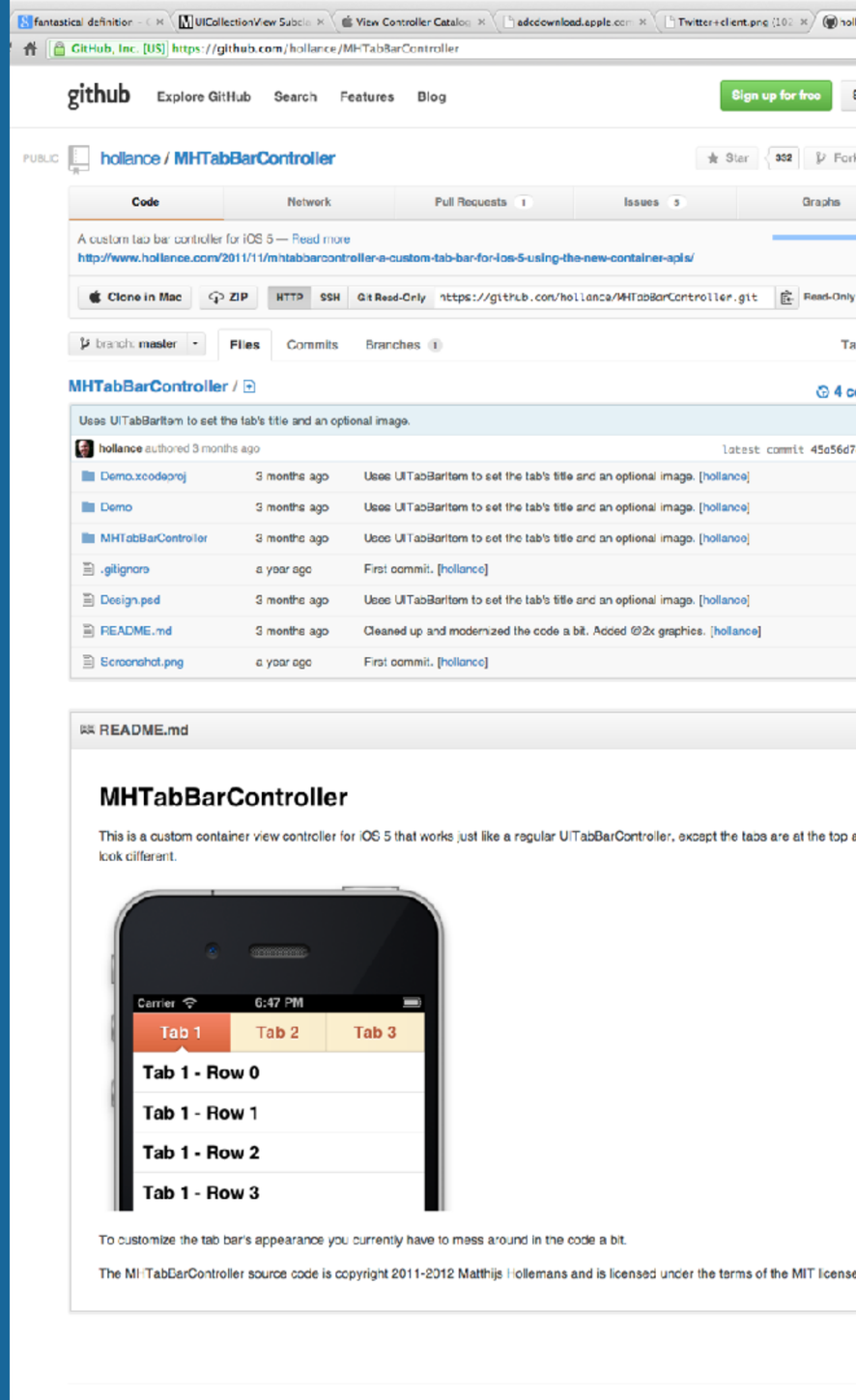
- Aims
  - Identify best practices
  - Increase your exposure (both senses of the word)
  - See under-the-hood of design patterns
  - Code review



CONTROL STUDIES

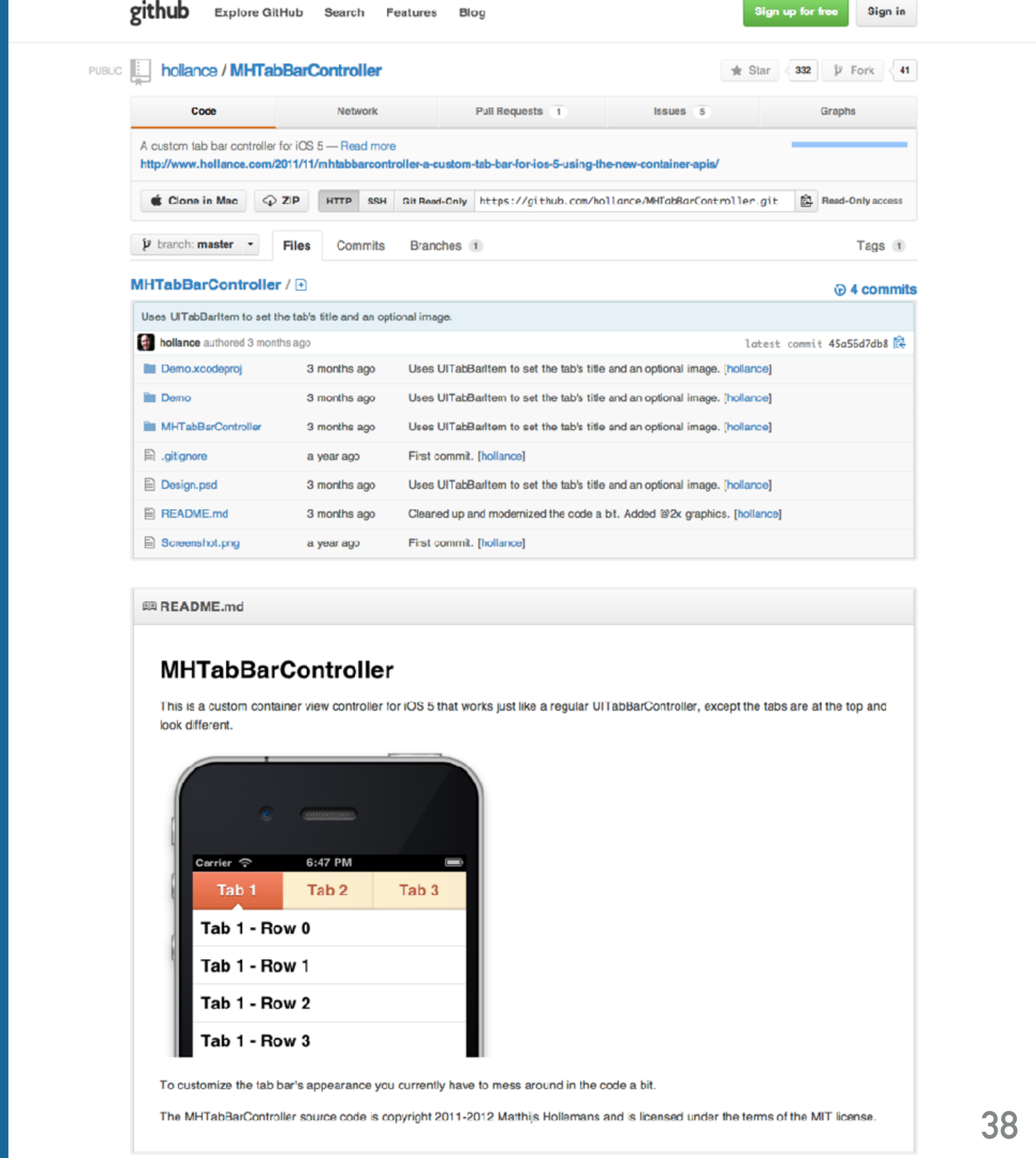
- 5% of final grade
- Requirements
  - ~15 minutes technical presentation and questions
  - Code review of control
  - Implement the control in a demo app that you will walk-through
  - Source code will be put in class playground to share
  - Submit a pull-request to the author
    - Just had one closed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

Good interview prep



CONTROL STUDIES

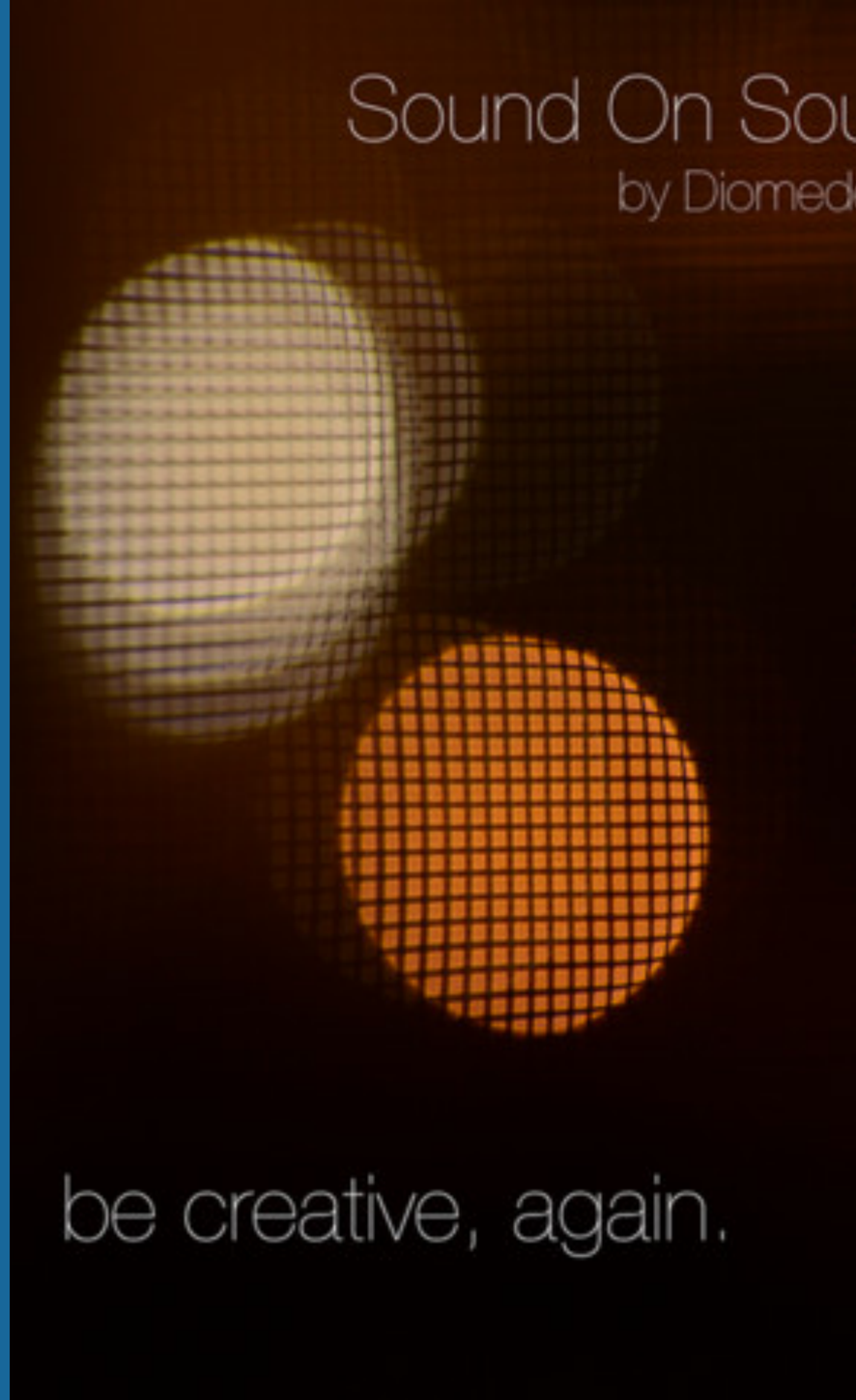
- Signups will be sent out later
- Find something to use in your final project



FINAL PROJECTS

FINAL PROJECTS

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



ADVANCED IOS

- Questions??

ASSIGNMENT 1

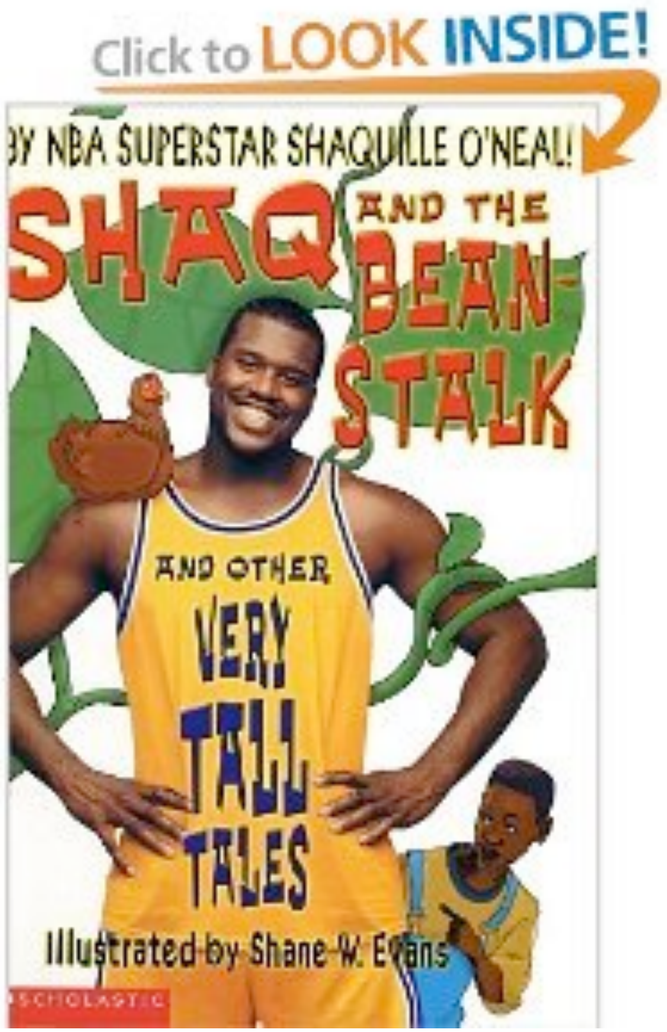
POP QUIZ

What do John Travollta, 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?

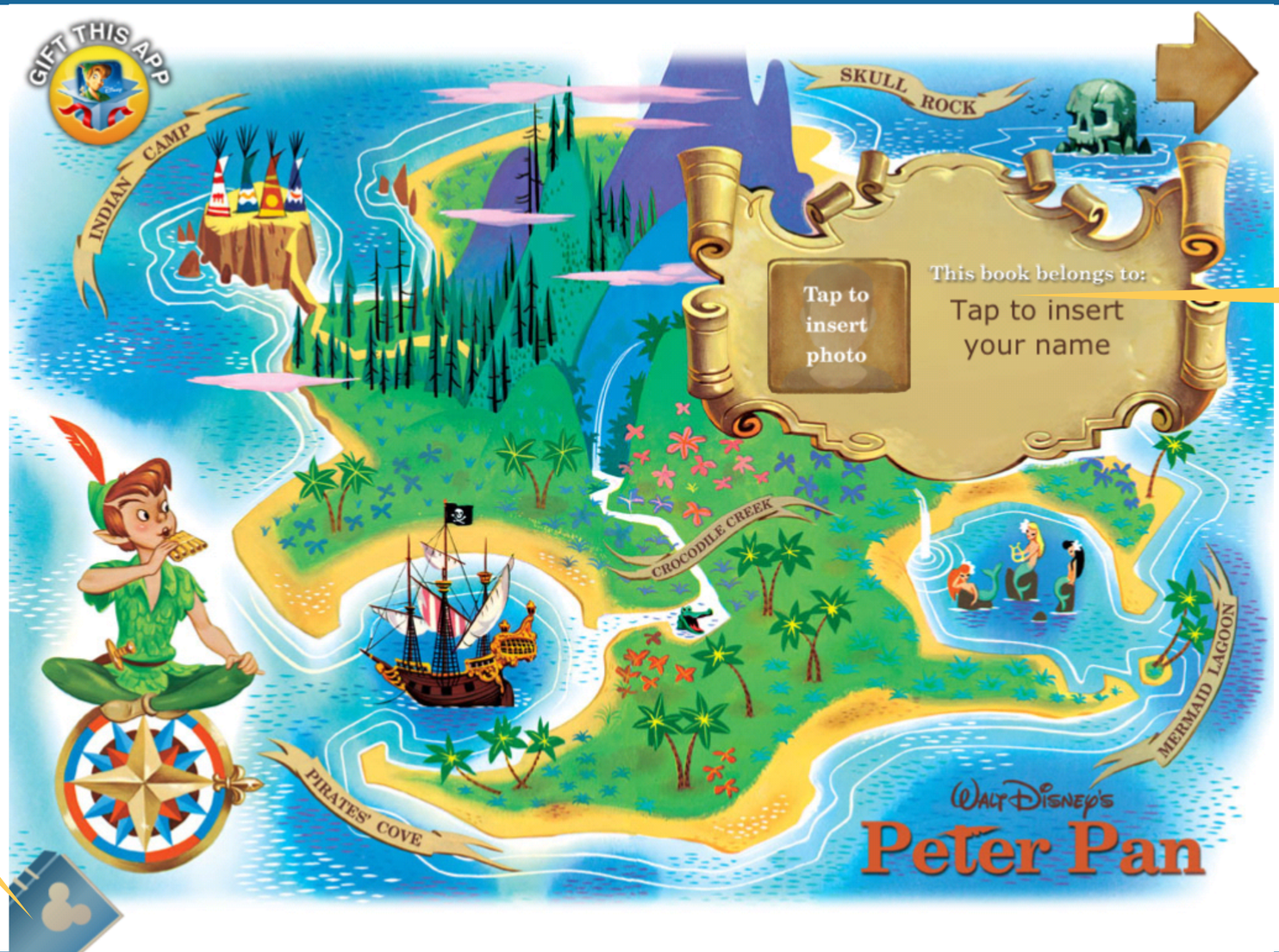
ASSIGNMENT 1

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



Settings

About the author

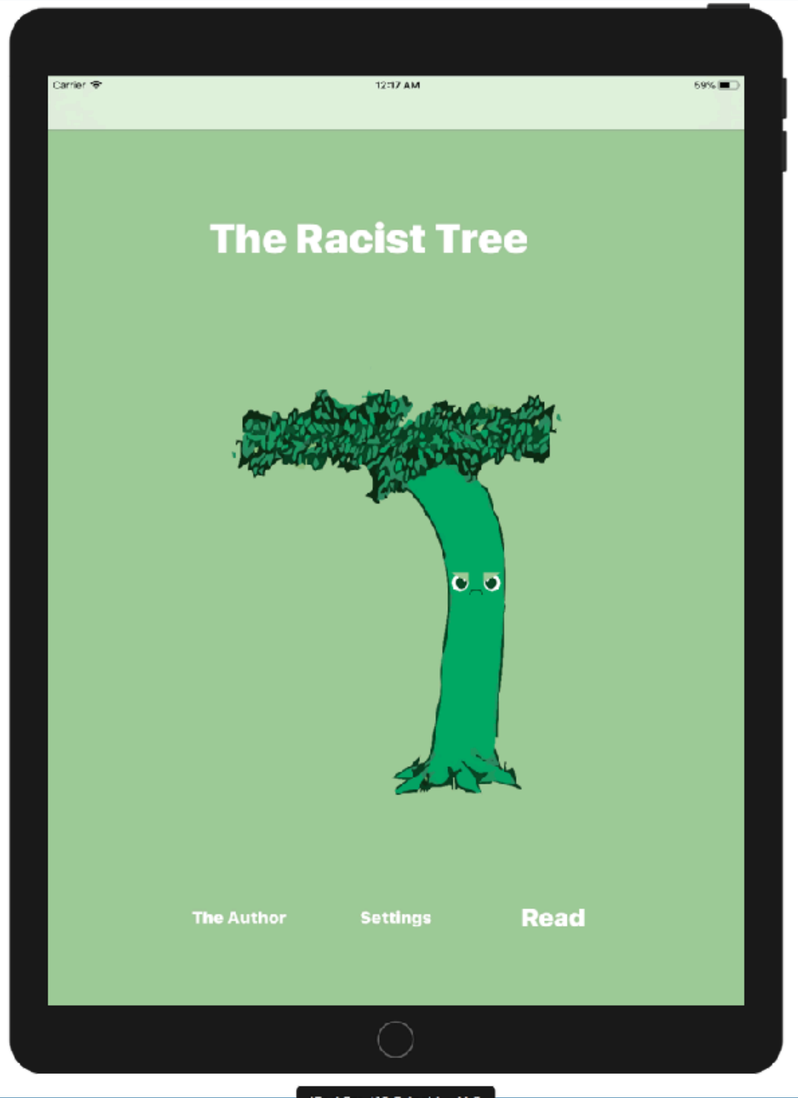
ASSIGNMENT 1

Page forward

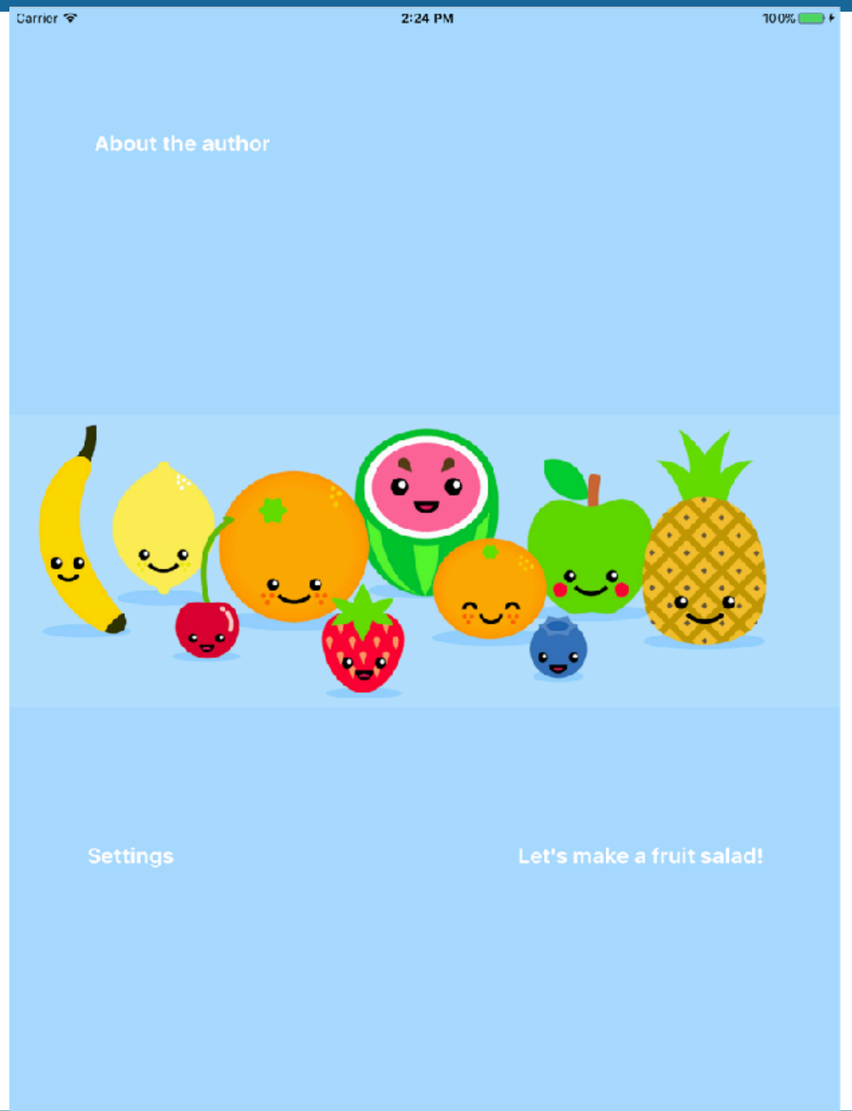
Page Backward

interactive element

ASSIGNMENT 1



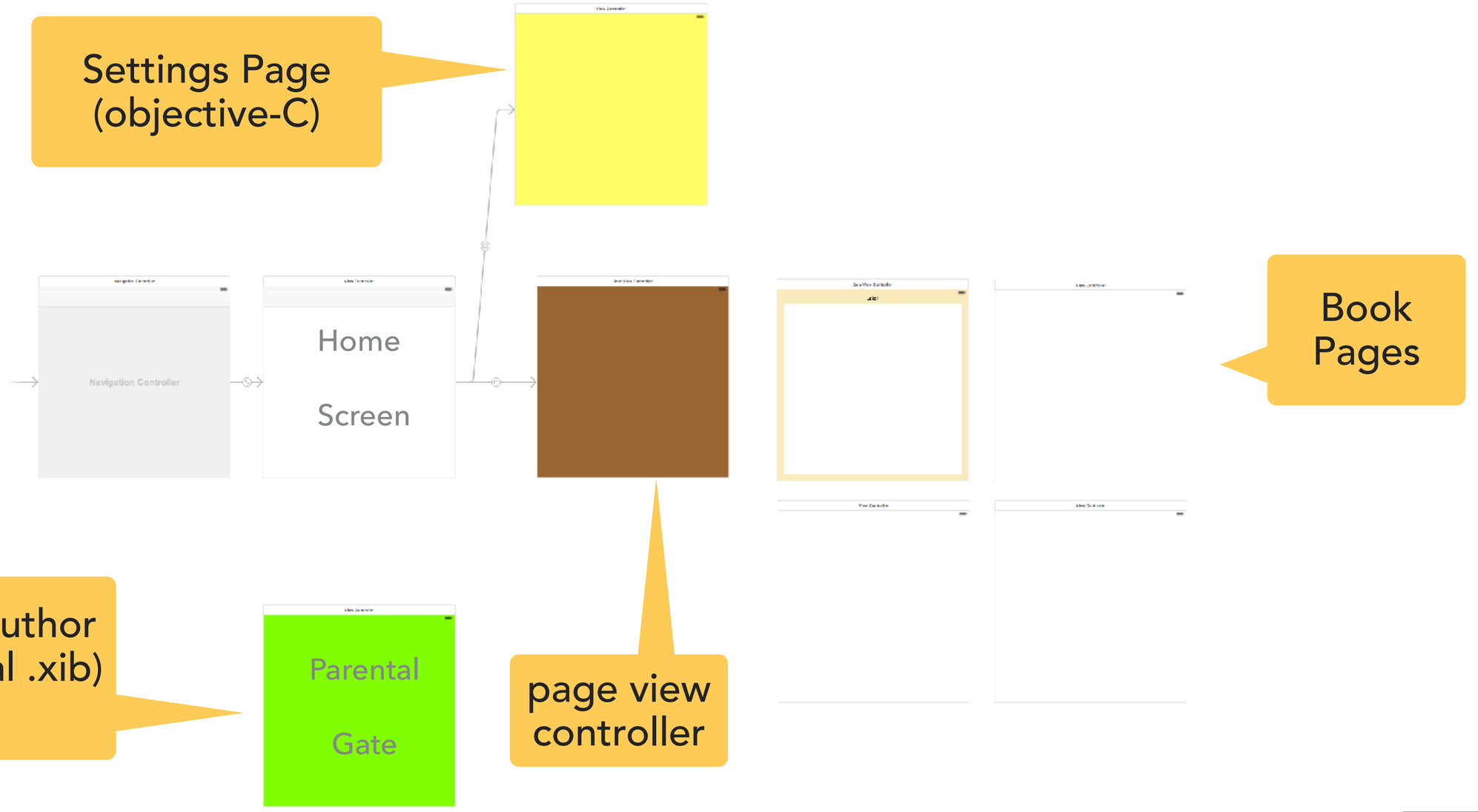
ASSIGNMENT 1



ASSIGNMENT 1


Carrier
5:23 PM
100%
Children_Book

OVERALL REQUIREMENTS

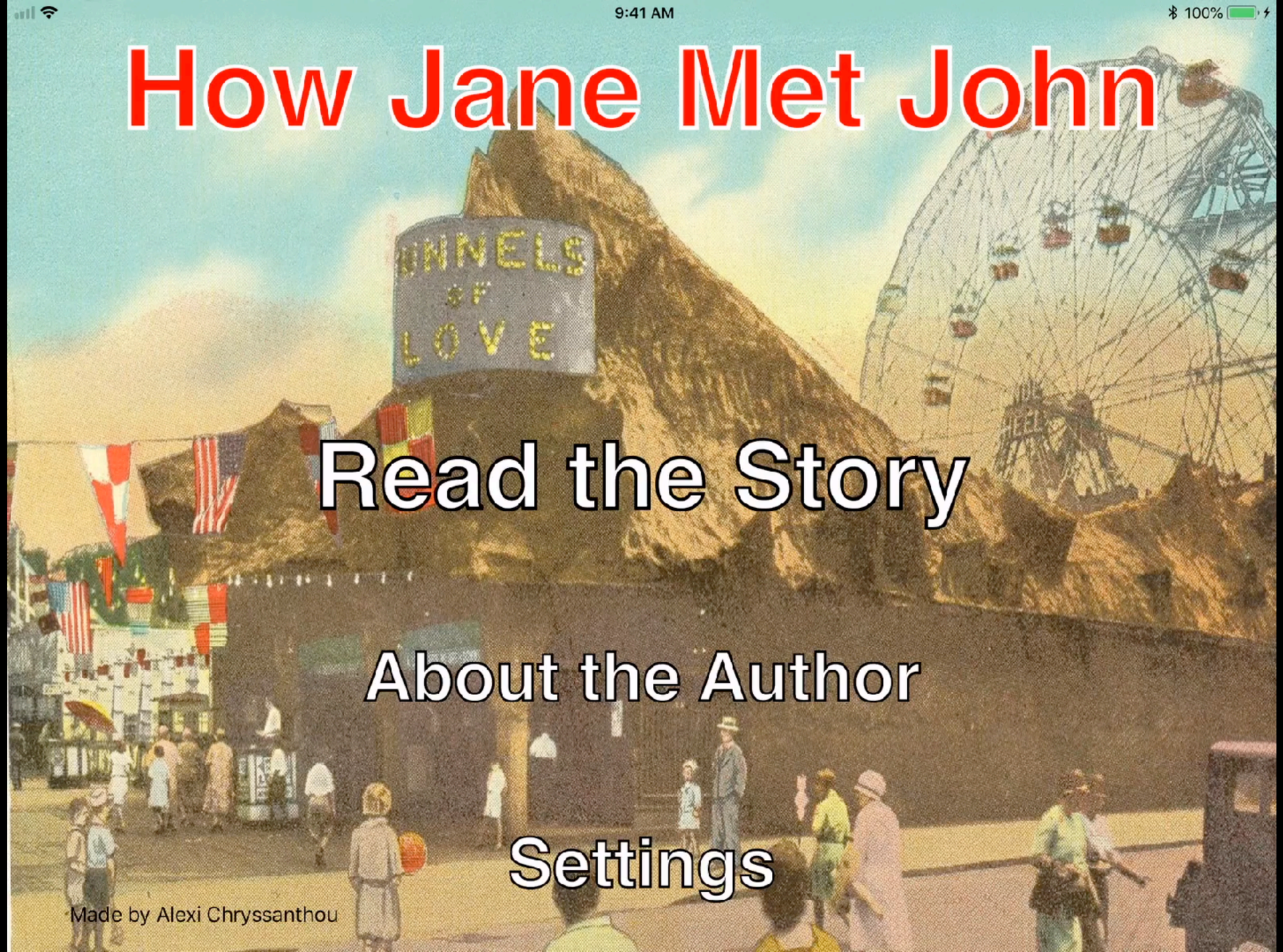


OVERALL REQUIREMENTS

- “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

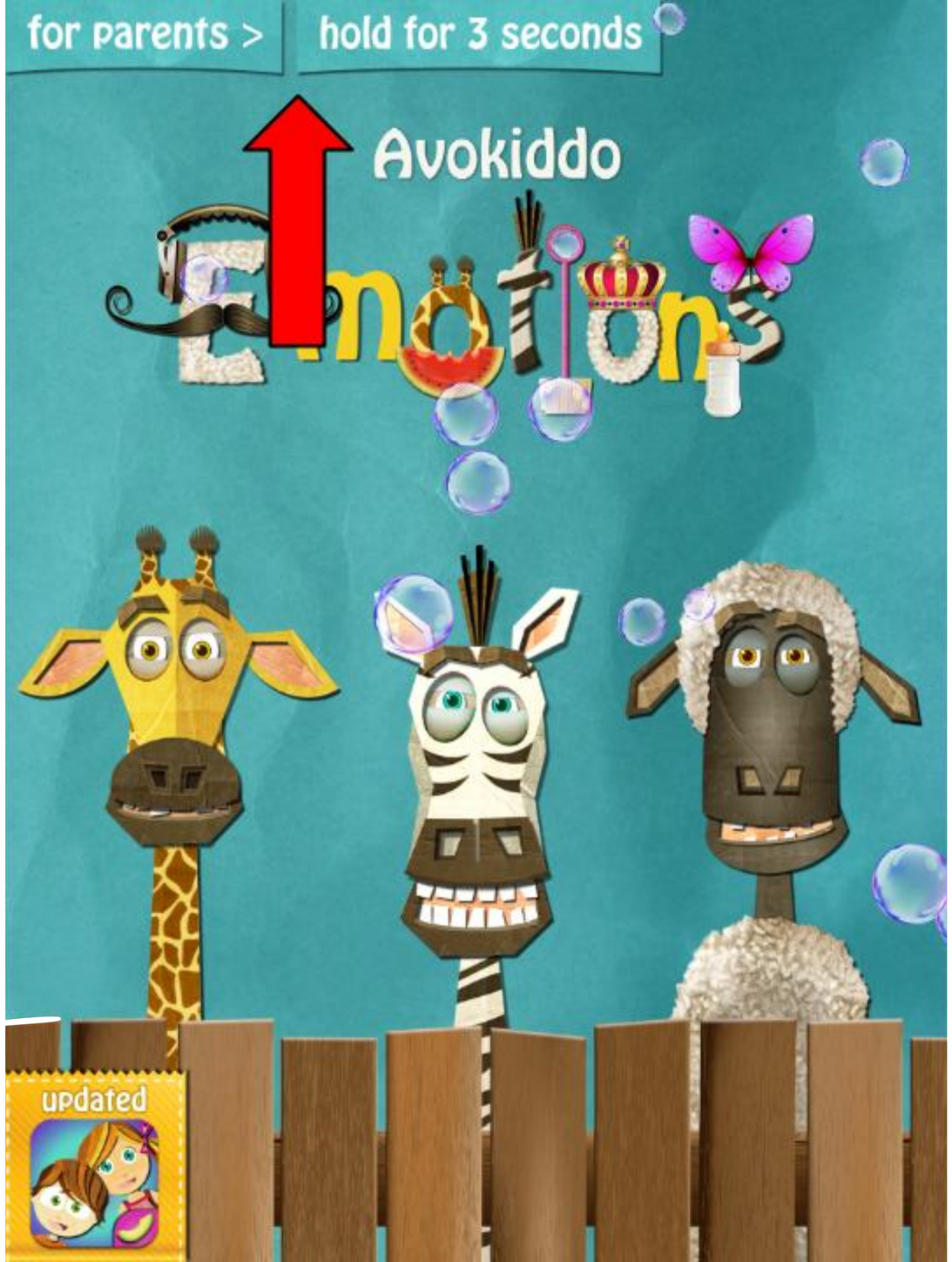
- Every content page must implement:
  - “Read to me” functionality
    - Record sound using Audacity, Garage Band, iPhone and email to self (UITextKit,AVSpeechSynthesis, next week)
    - Words should highlight as they are read
    - Start automatically or by button tap (depending on user preference 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


9:41 AM
100%
How Jane Met John
ONNELS
OF
LOVE
Read the Story
About the Author
Settings
Made by Alexi Chryssanthou

PARENTAL GATE

SUBTITLE

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



A Hummingbird's Heart

Start | About | Settings

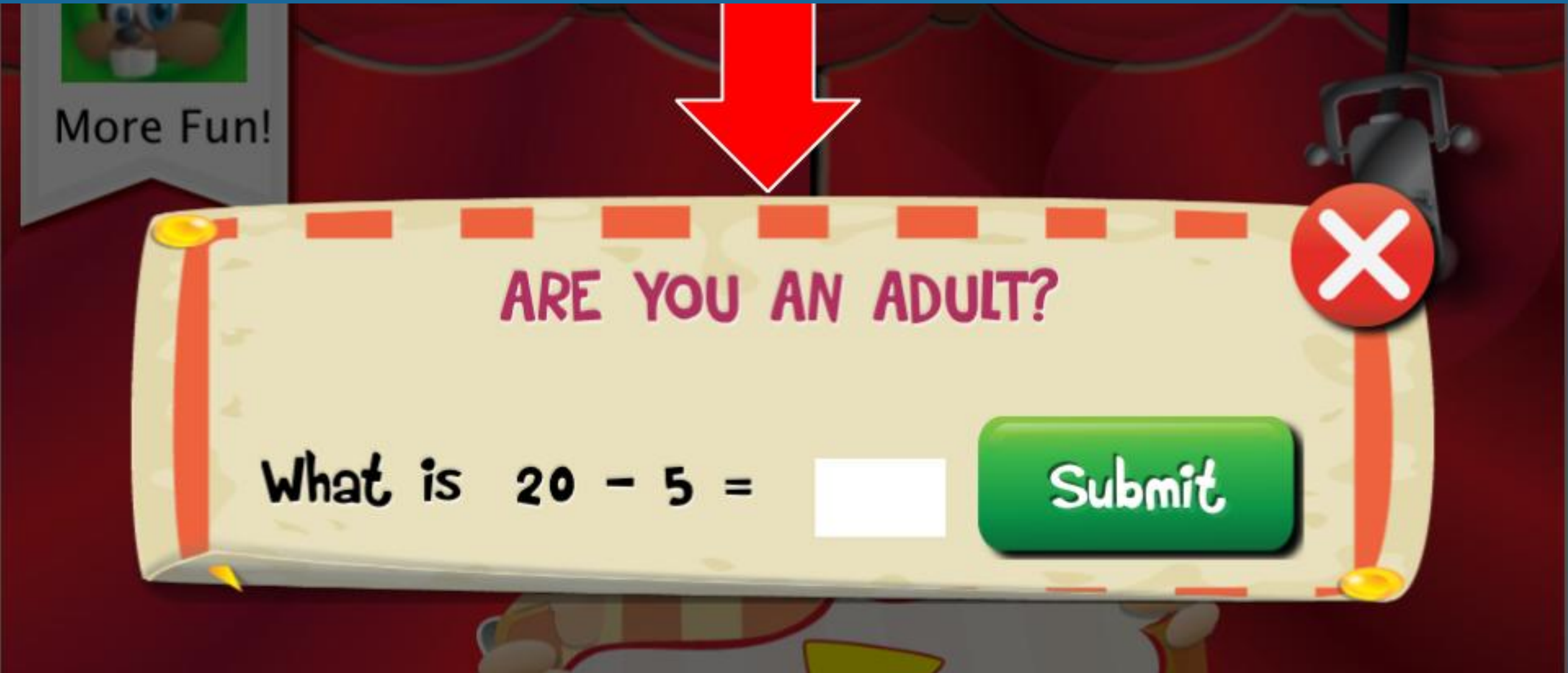
PARENTAL GATE

- Implementation used to be a best practice, now it is required

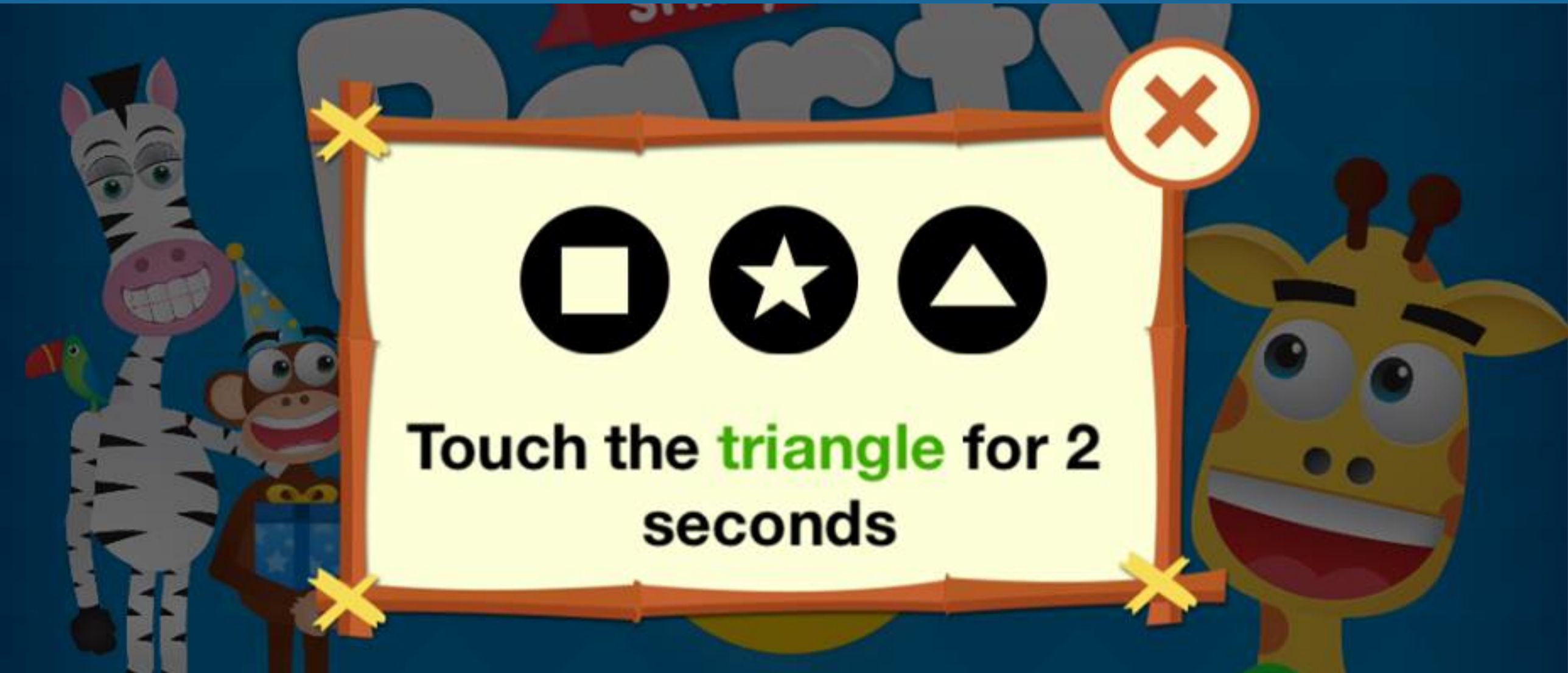
> 24. Kids Apps
>
> - 24.3- Apps primarily intended for use by kids under 13 must get parental permission or use a parental gate before allowing the user to link out of the app or engage in commerce
>
> This requires gating when linking out of the app or engaging in commerce such as selling in-app items.

-

PARENTAL GATE



PARENTAL GATE



PARENTAL GATE

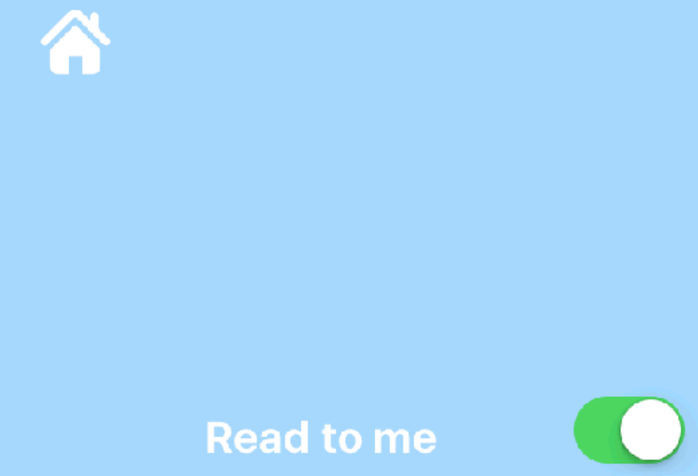
REQUIREMENTS

- 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

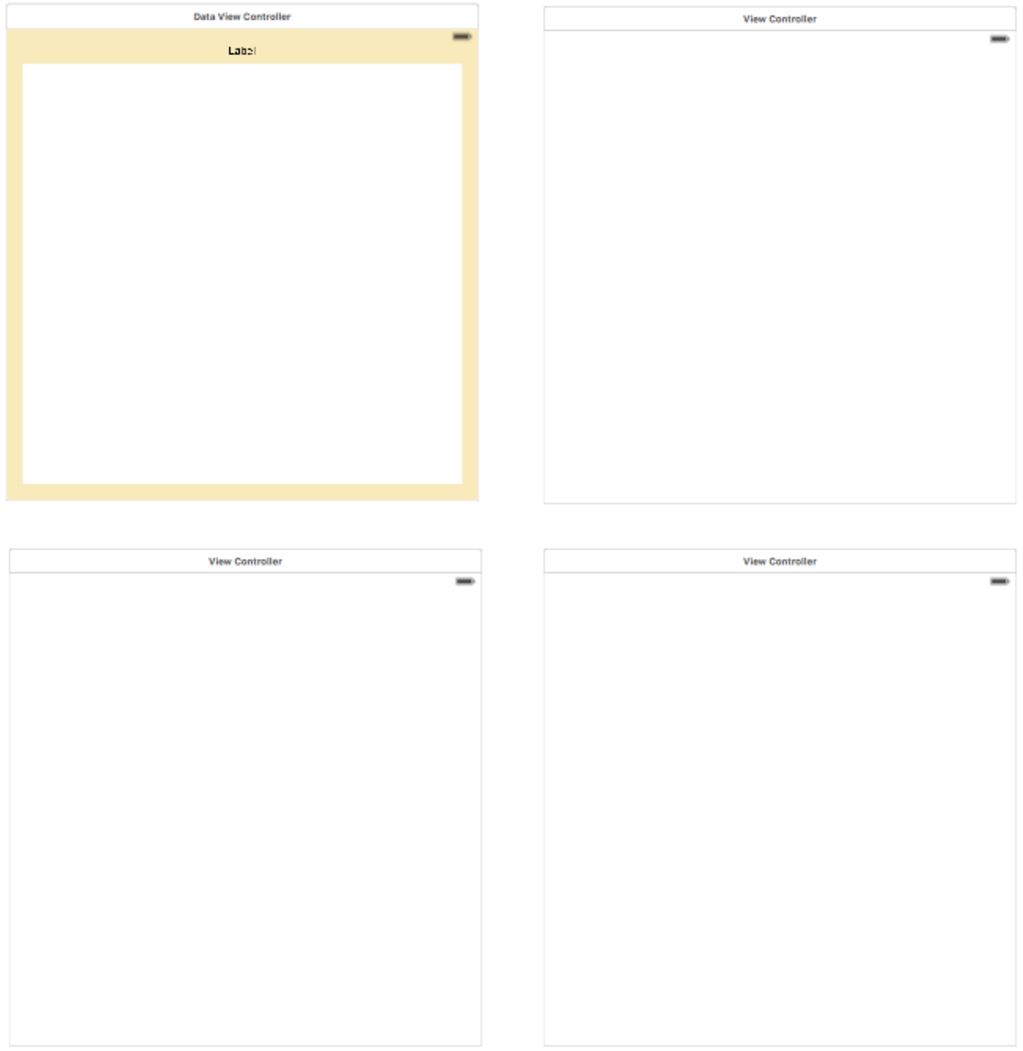
- Settings page in objective-c 😳
- Allow user to select preference to automatically read the page
    - Default should be "on"



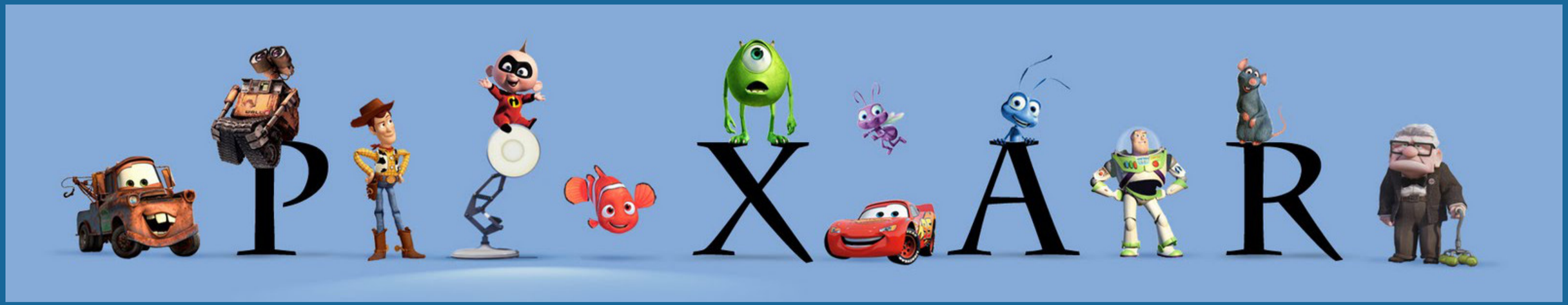
BOOK PAGE OPTIONS

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

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1A

VIEW CONTROLLER EVOLUTION

VIEW CONTROLLER EVOLUTION

THE VIEW CONTROLLER

THE VIEW CONTROLLER

IN THE BEGINNING...

- UIViewControllers 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

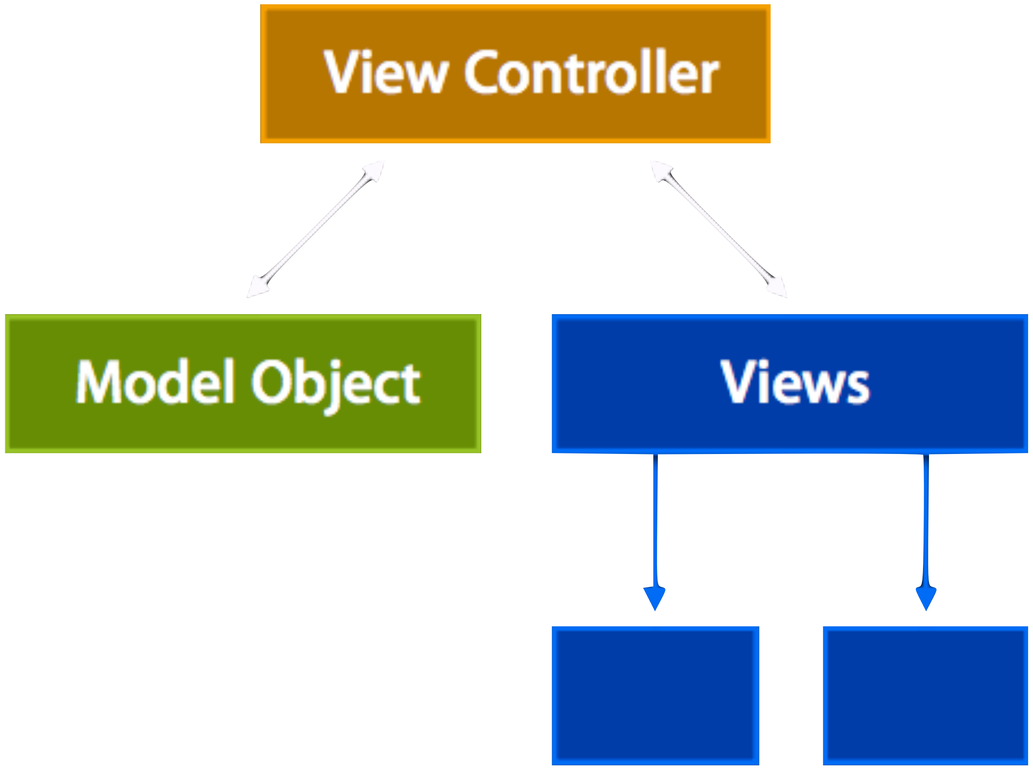


Image via Apple

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

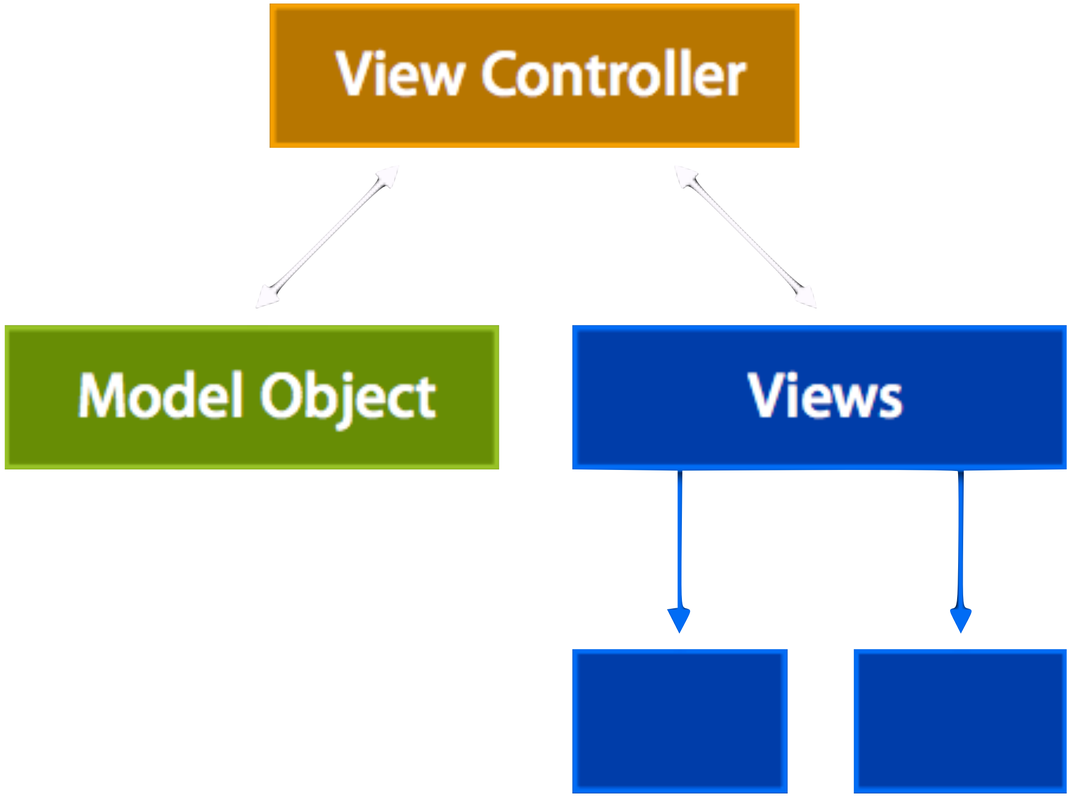
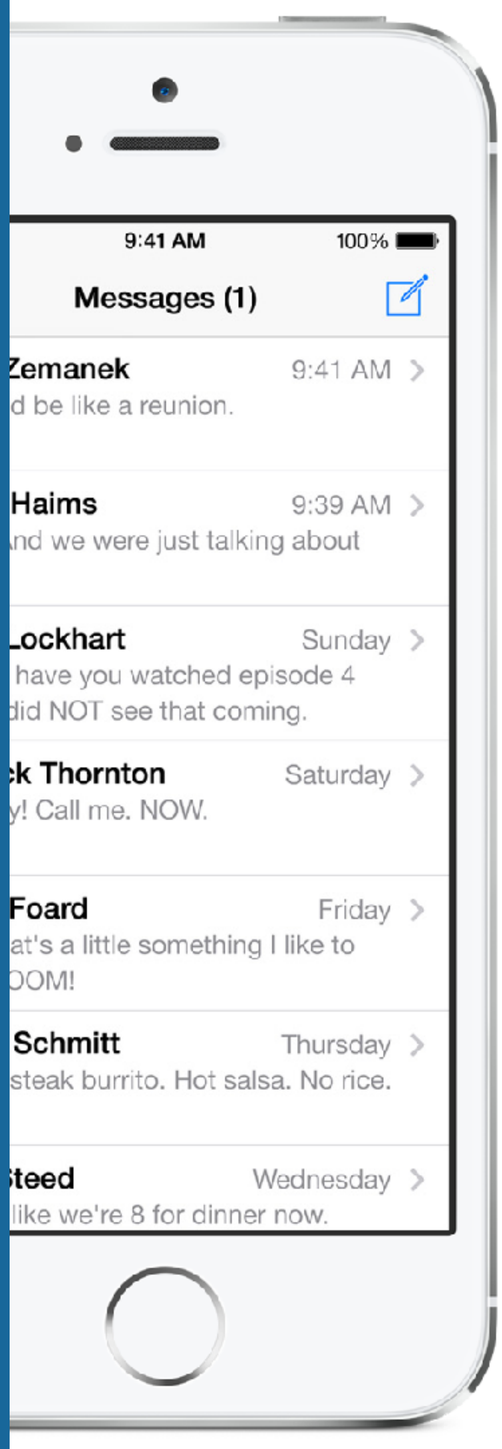
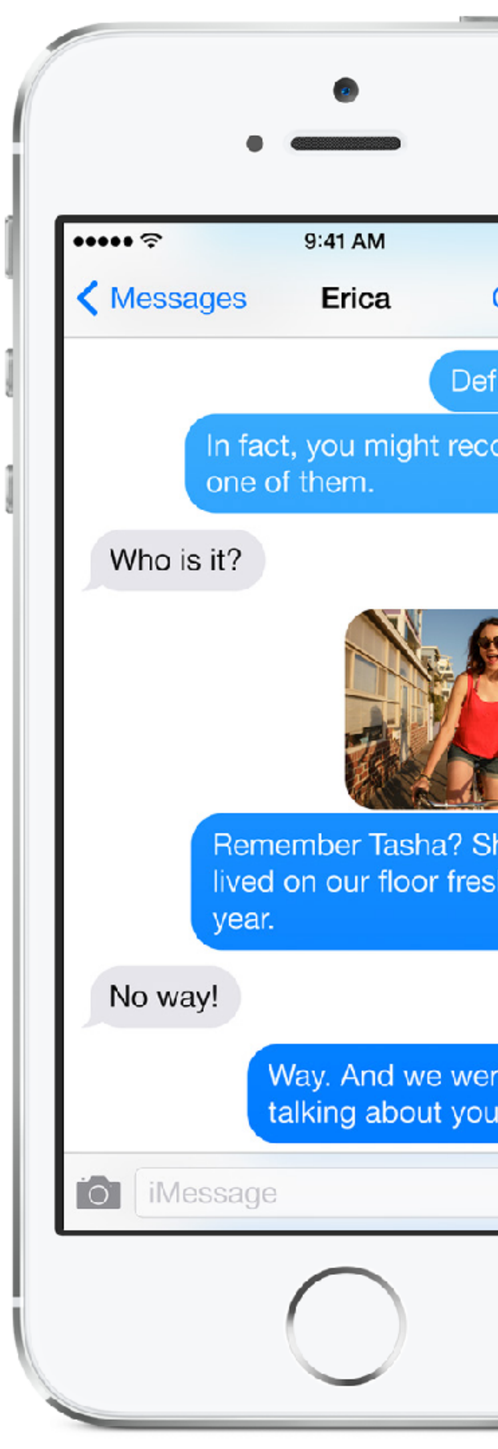


Image via Apple

VIEW CONTROLLERS

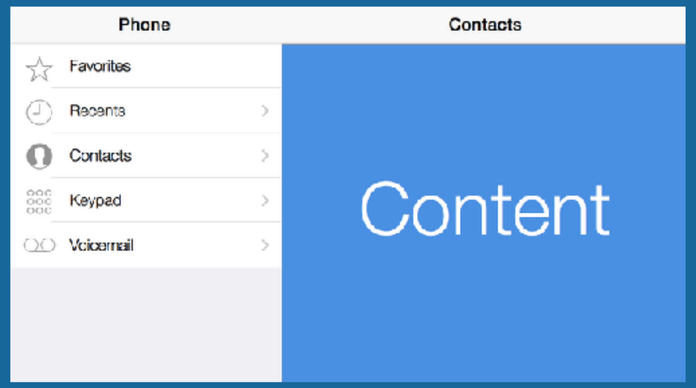
- For many interfaces it "makes sense"
- Screenful of content

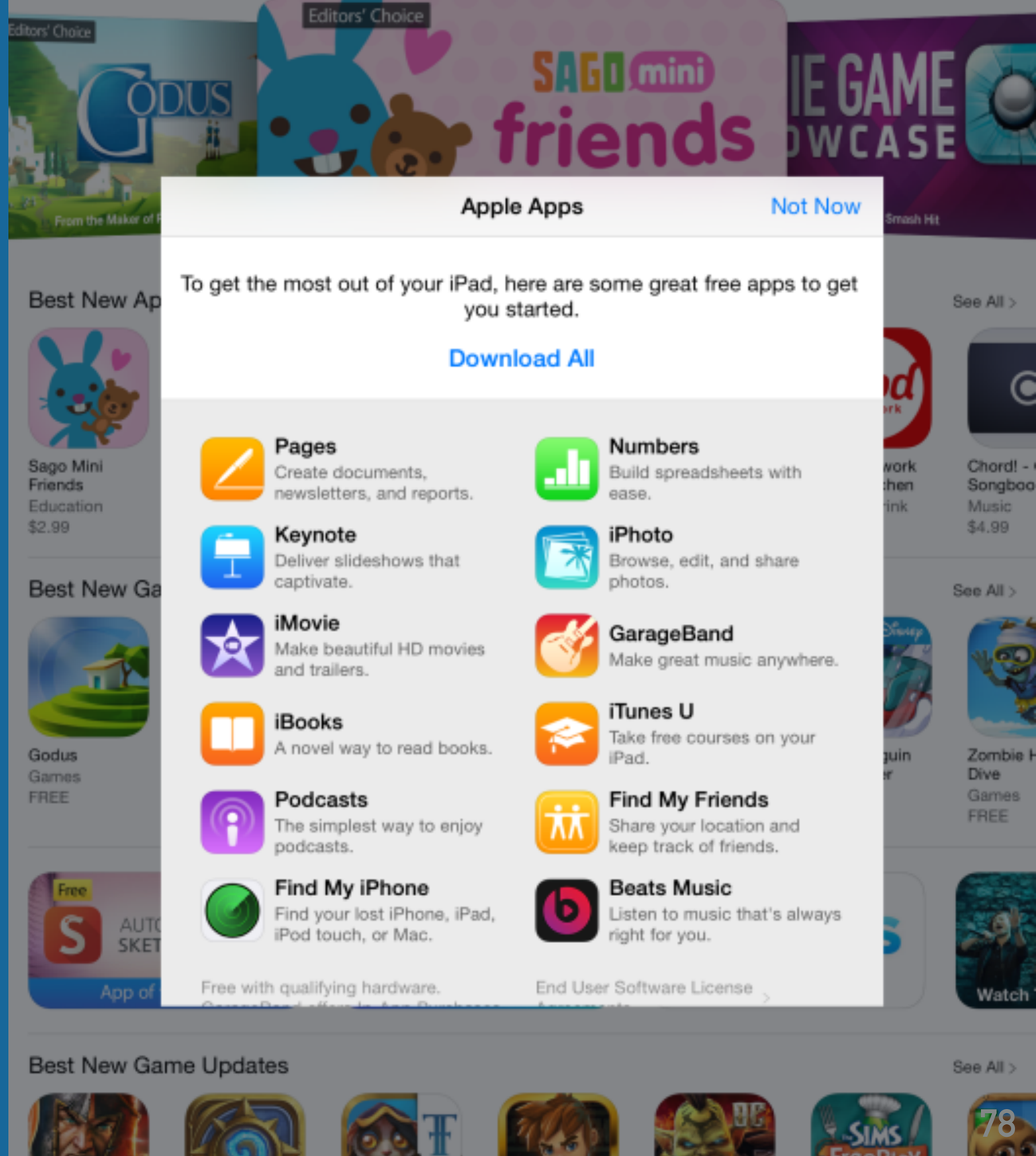




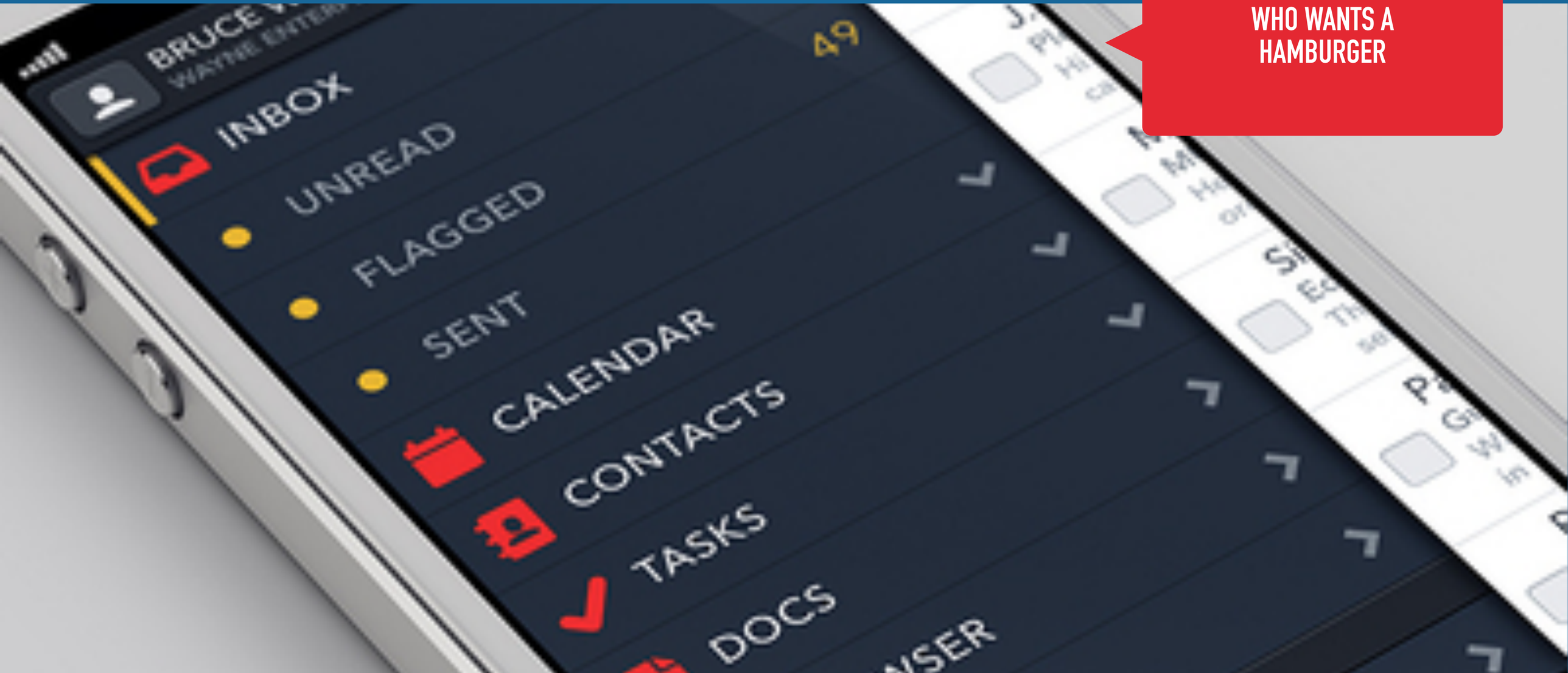
VIEW CONTROLLERS

- “Screenful of content” getting harder to define



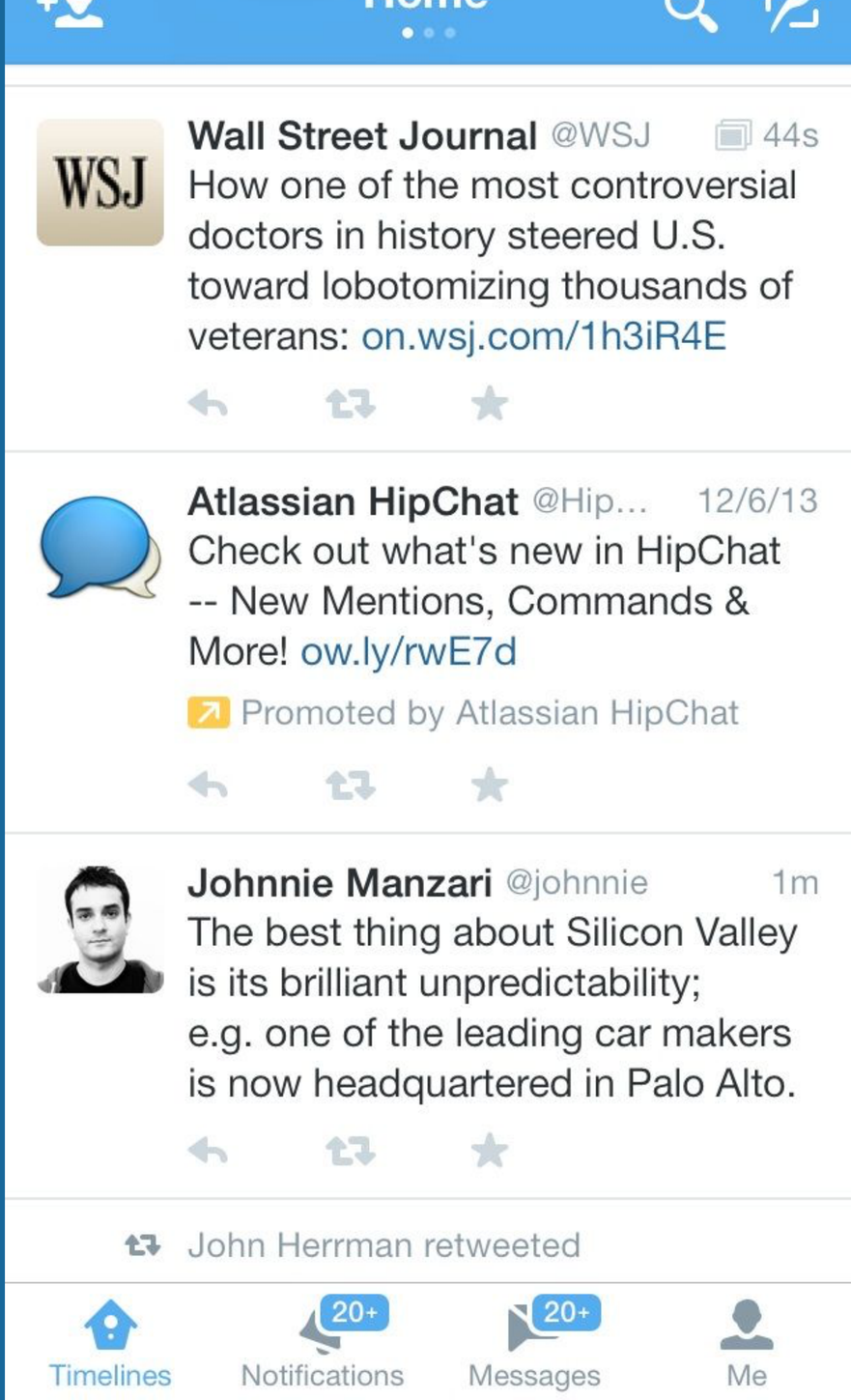


VIEW CONTROLLERS



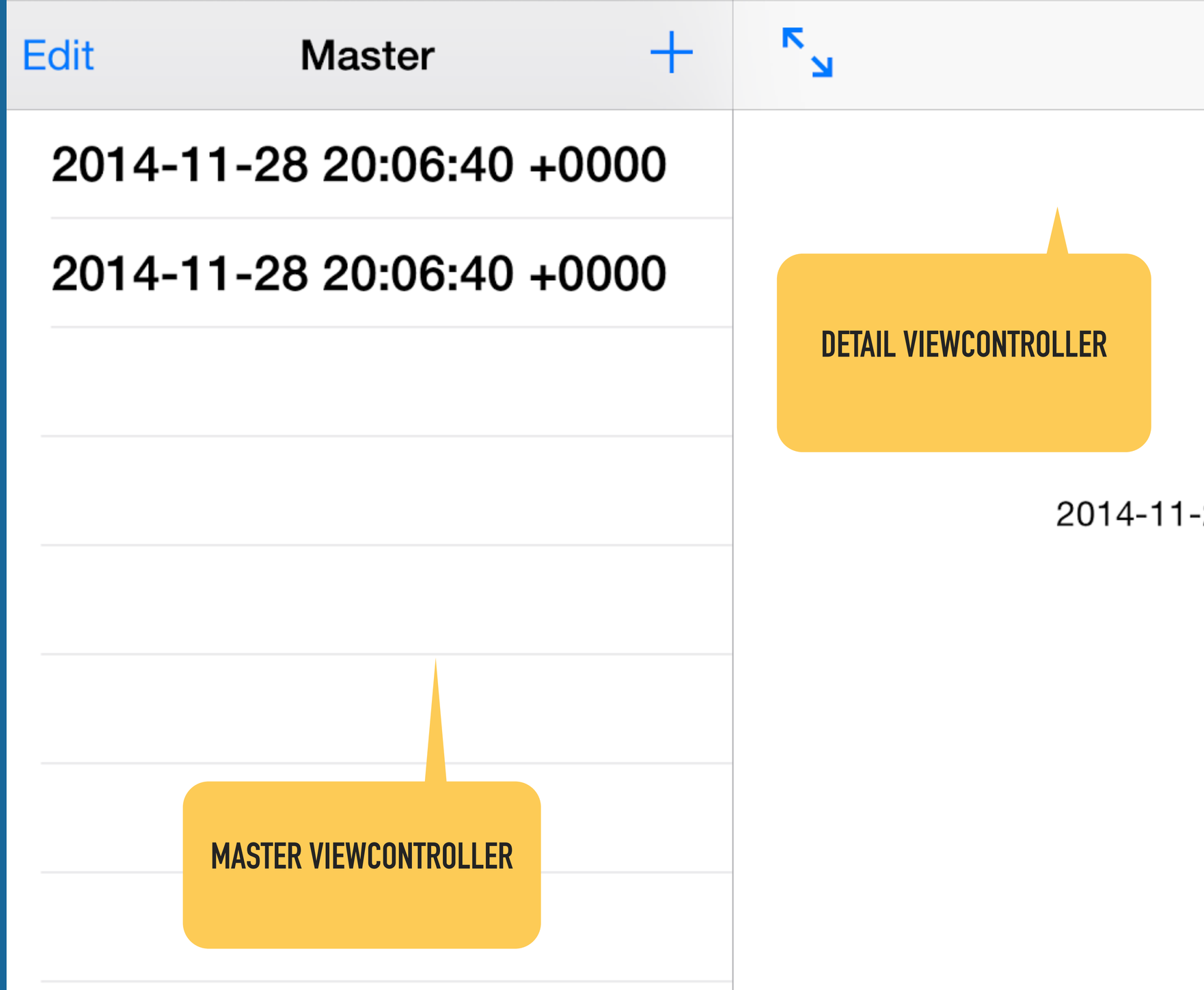
VIEW CONTROLLERS

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



AND NOW…

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



VIEW CONTROLLERS

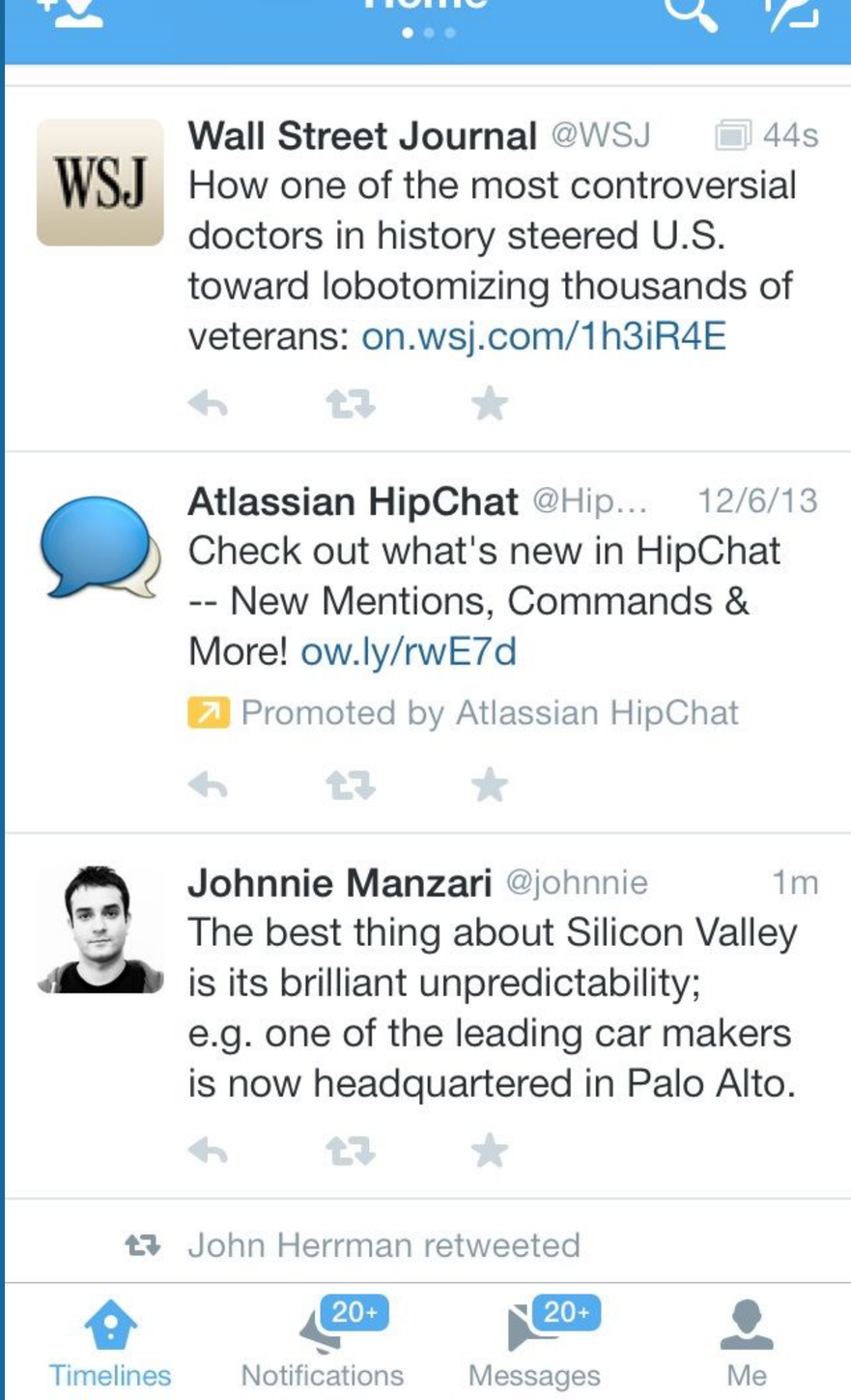
SUMMARY

- 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



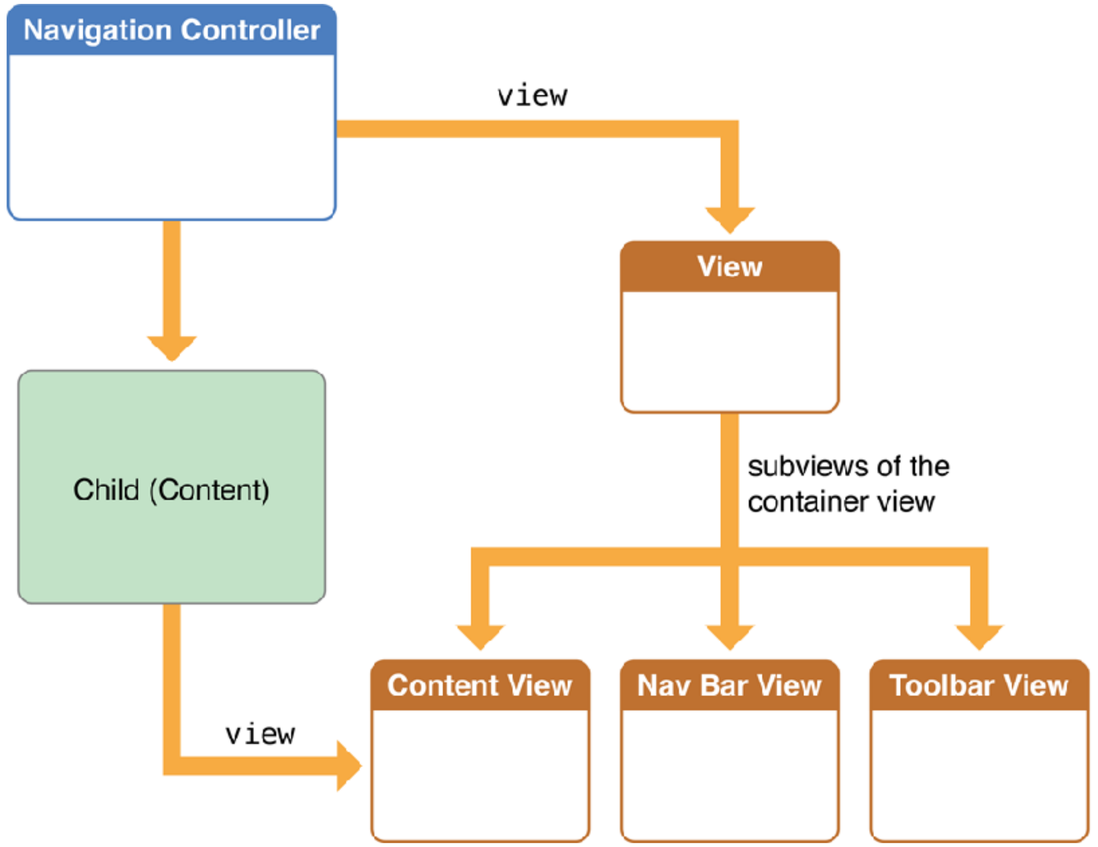
VIEW CONTROLLER CONTAINERS

- 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)

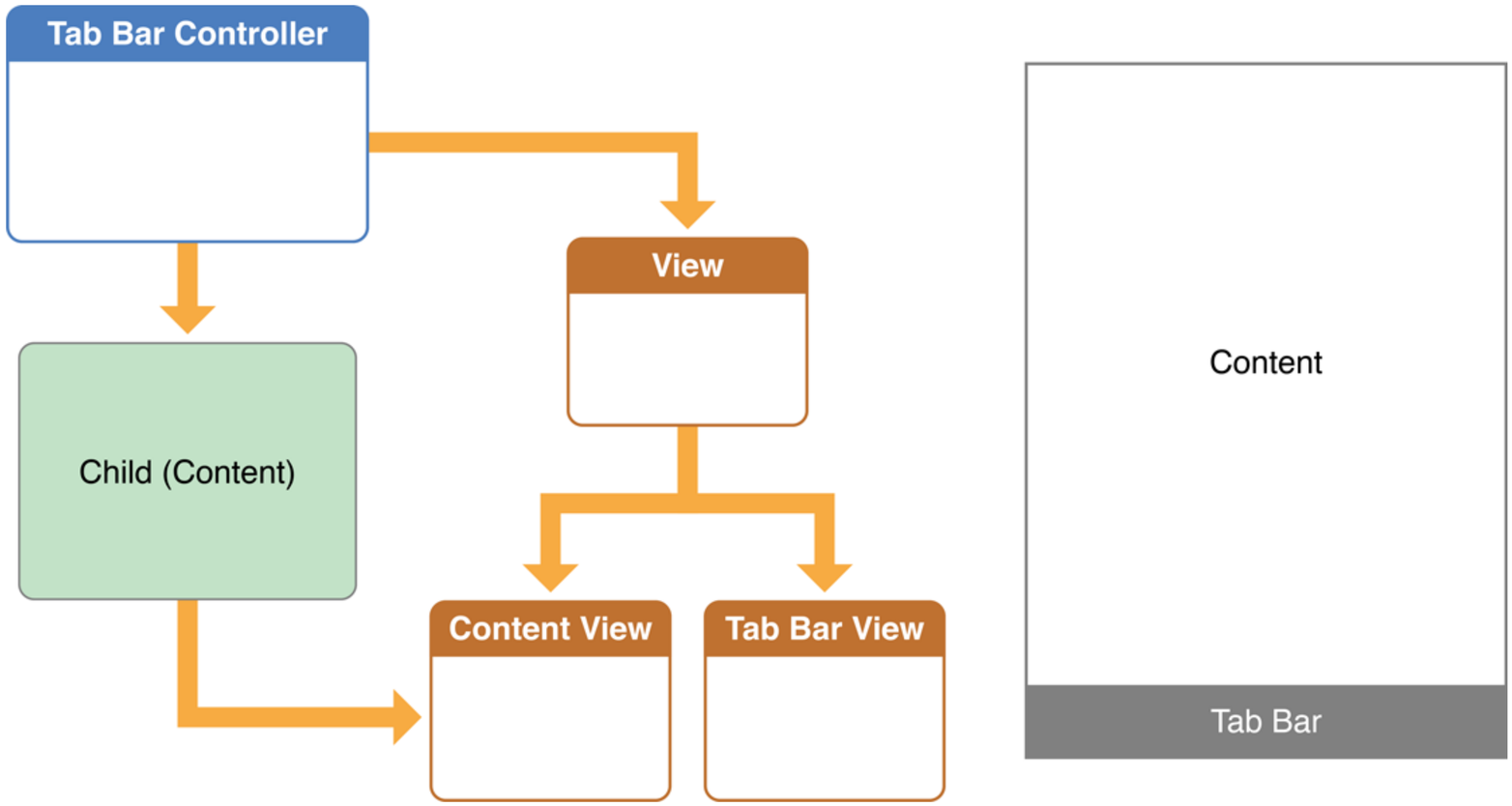
VIEW CONTROLLER CONTAINERS

- 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

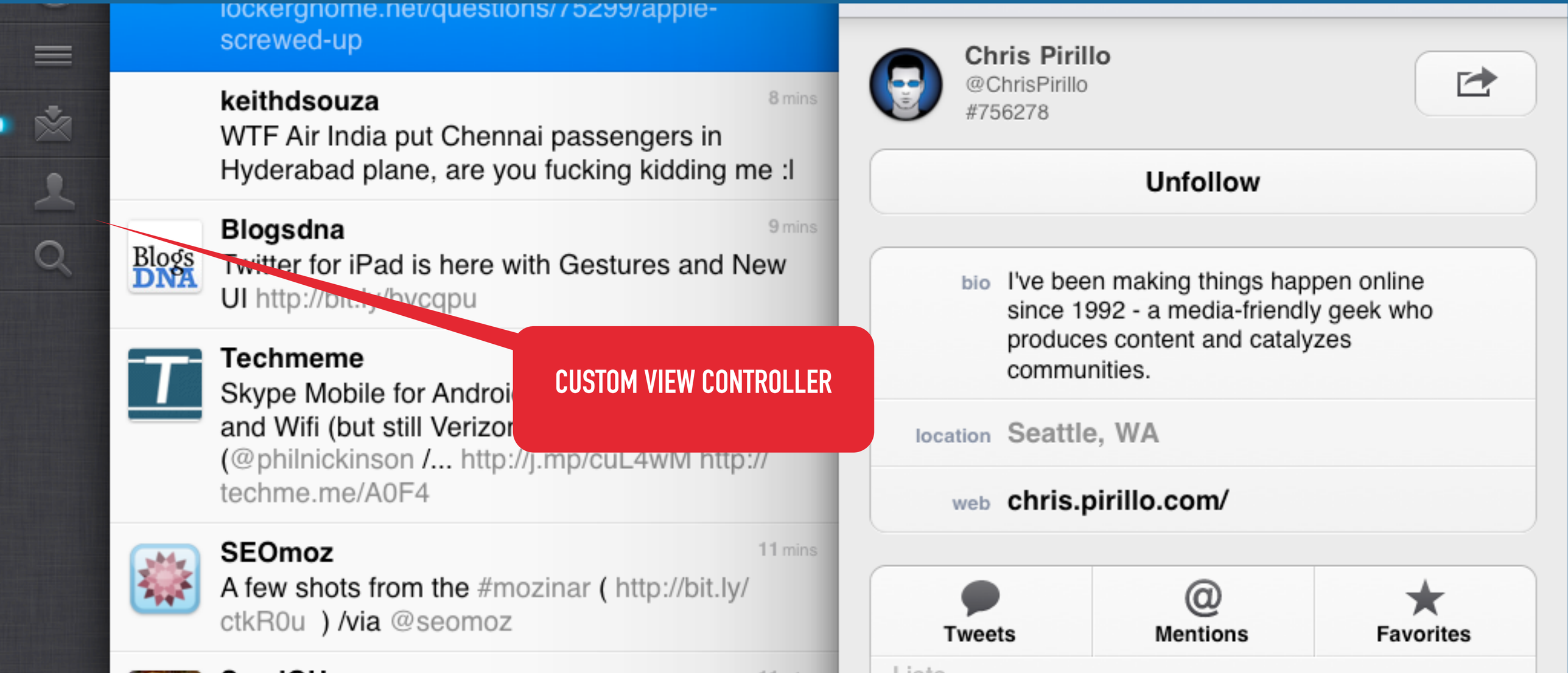


VIEW CONTROLLER CONTAINERS

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

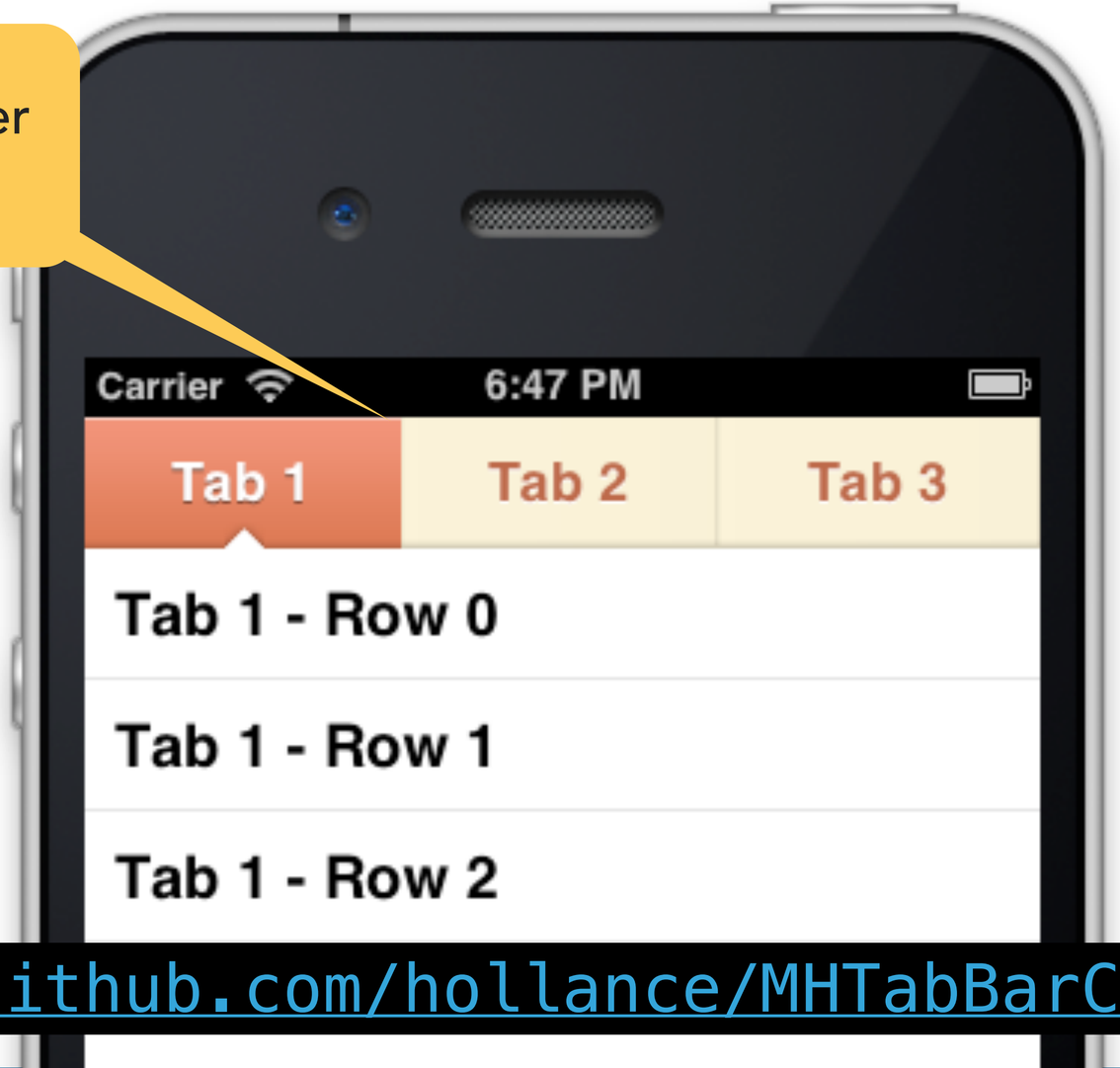


VIEW CONTROLLER CONTAINERS



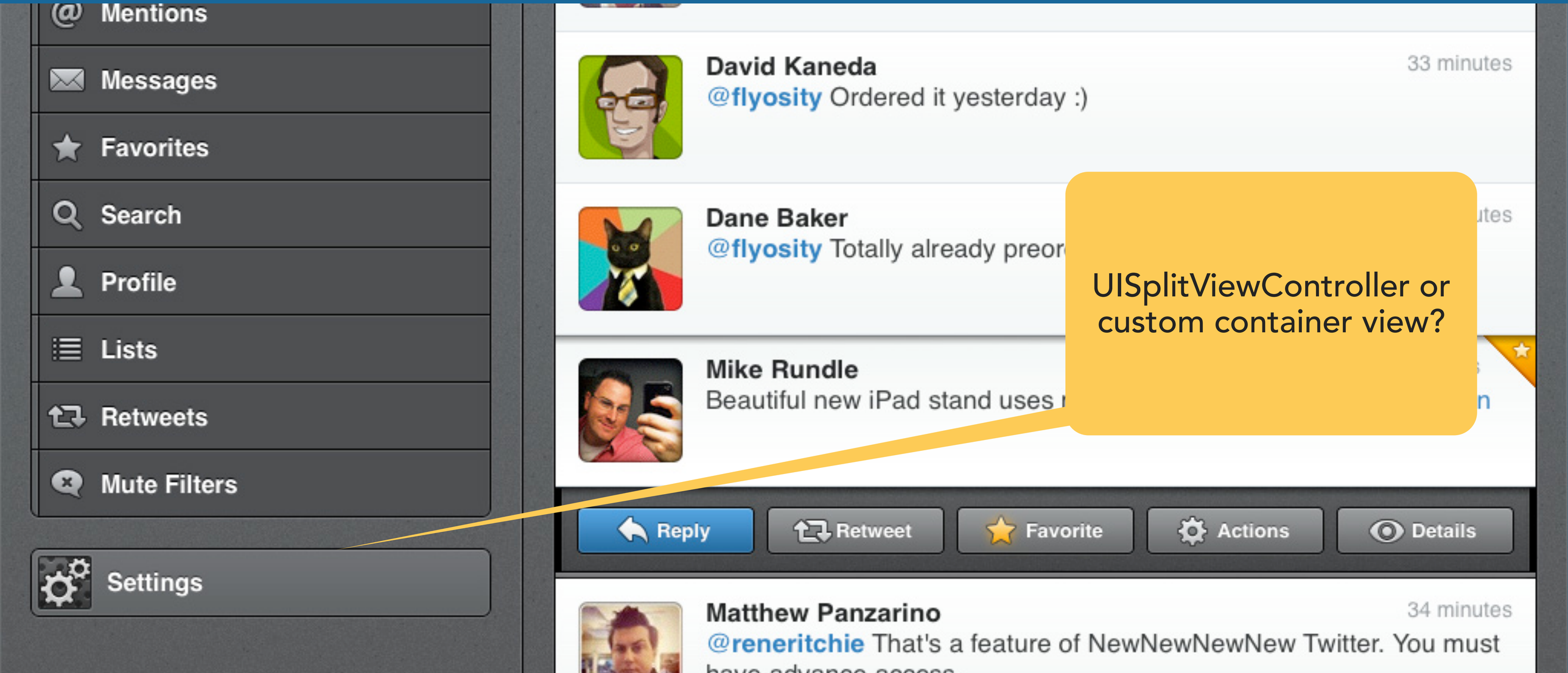
VIEW CONTROLLER CONTAINERS

custom view controller container



https://github.com/hollance/MHTabBarController

VIEW CONTROLLER CONTAINERS



VIEW CONTROLLER CONTAINMENT

VIEW CONTROLLER CONTAINERS

Blue
View color

Blue Arrow
Subview relationship

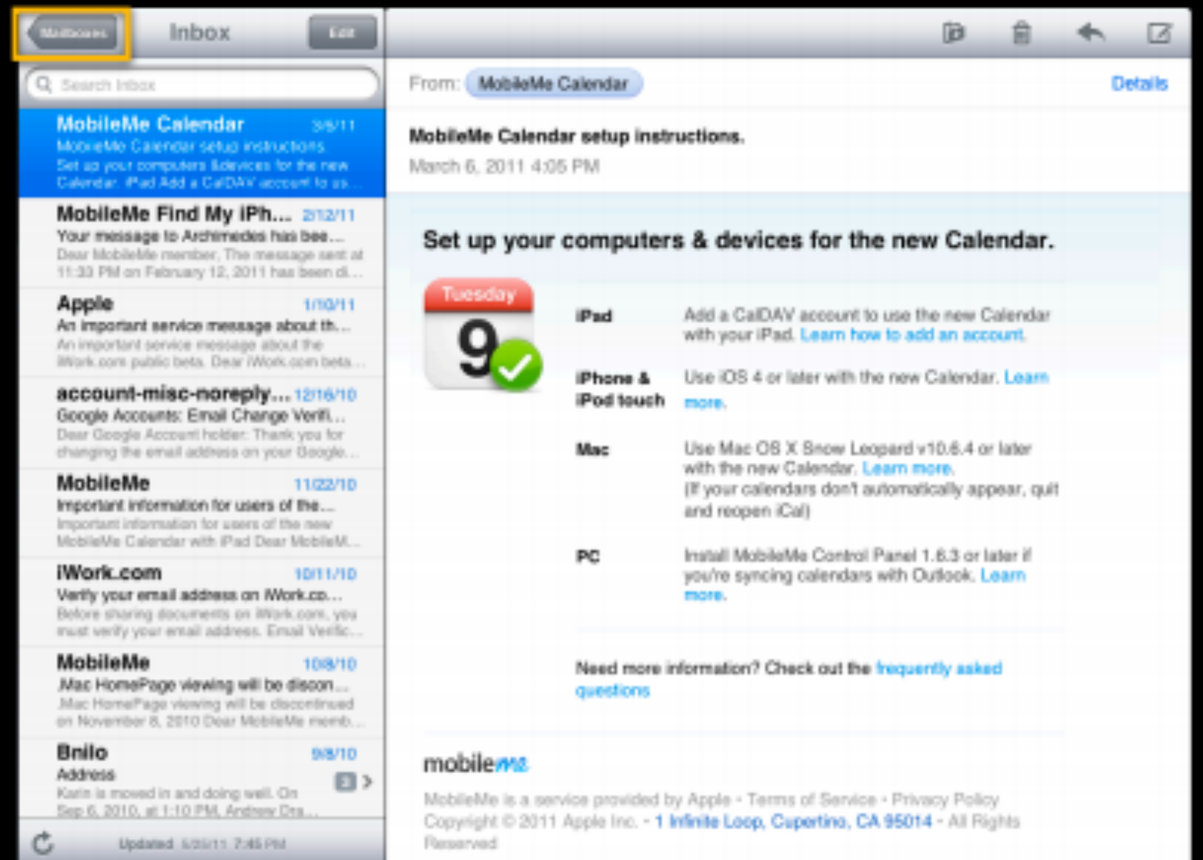
Gold
View controller color

Split View Controller

Gray Arrow
Parent view controller relationship

Image via Apple

VIEW CONTROLLER CONTAINERS



CONTENT CONTROLLER

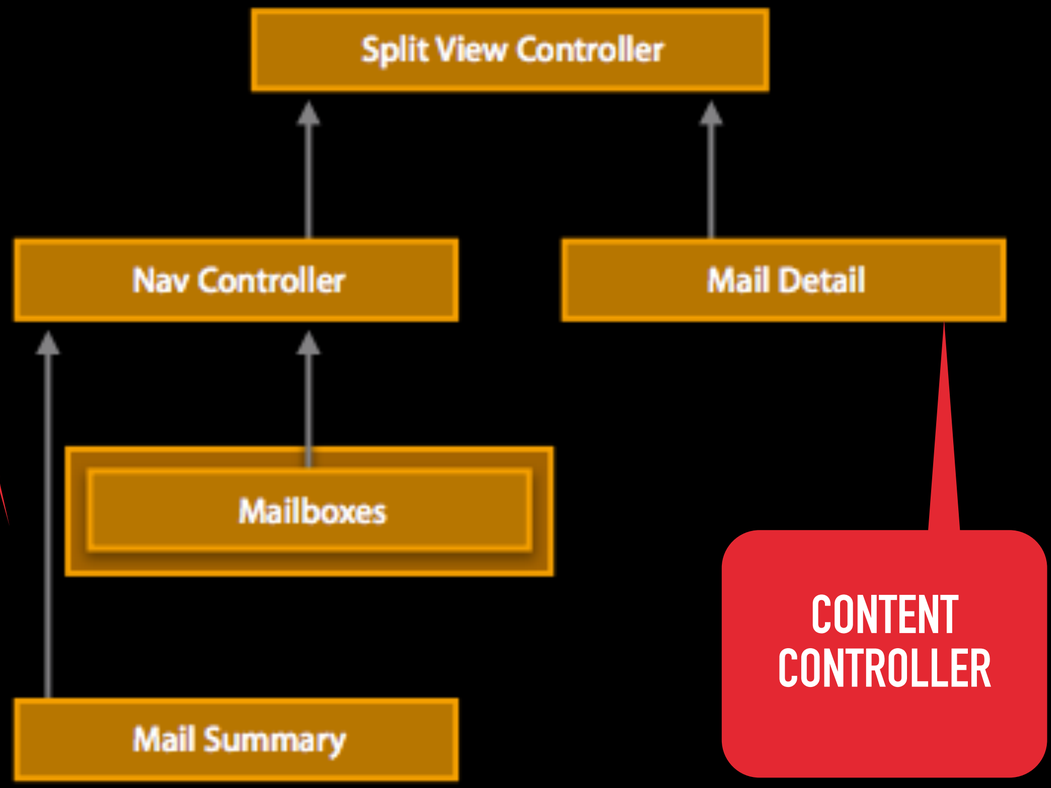


Image via Apple

VIEW CONTROLLER CONTAINERS

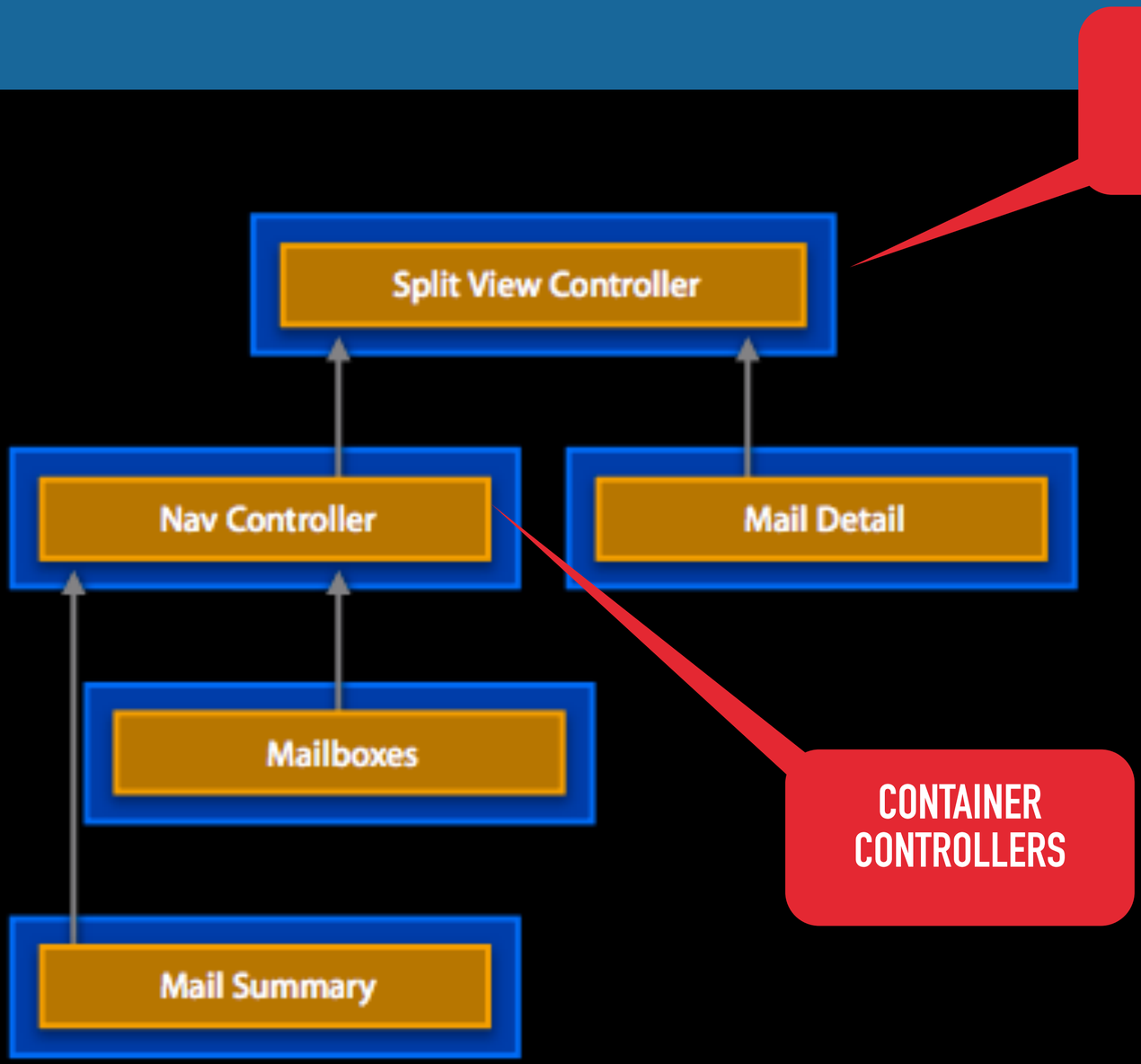
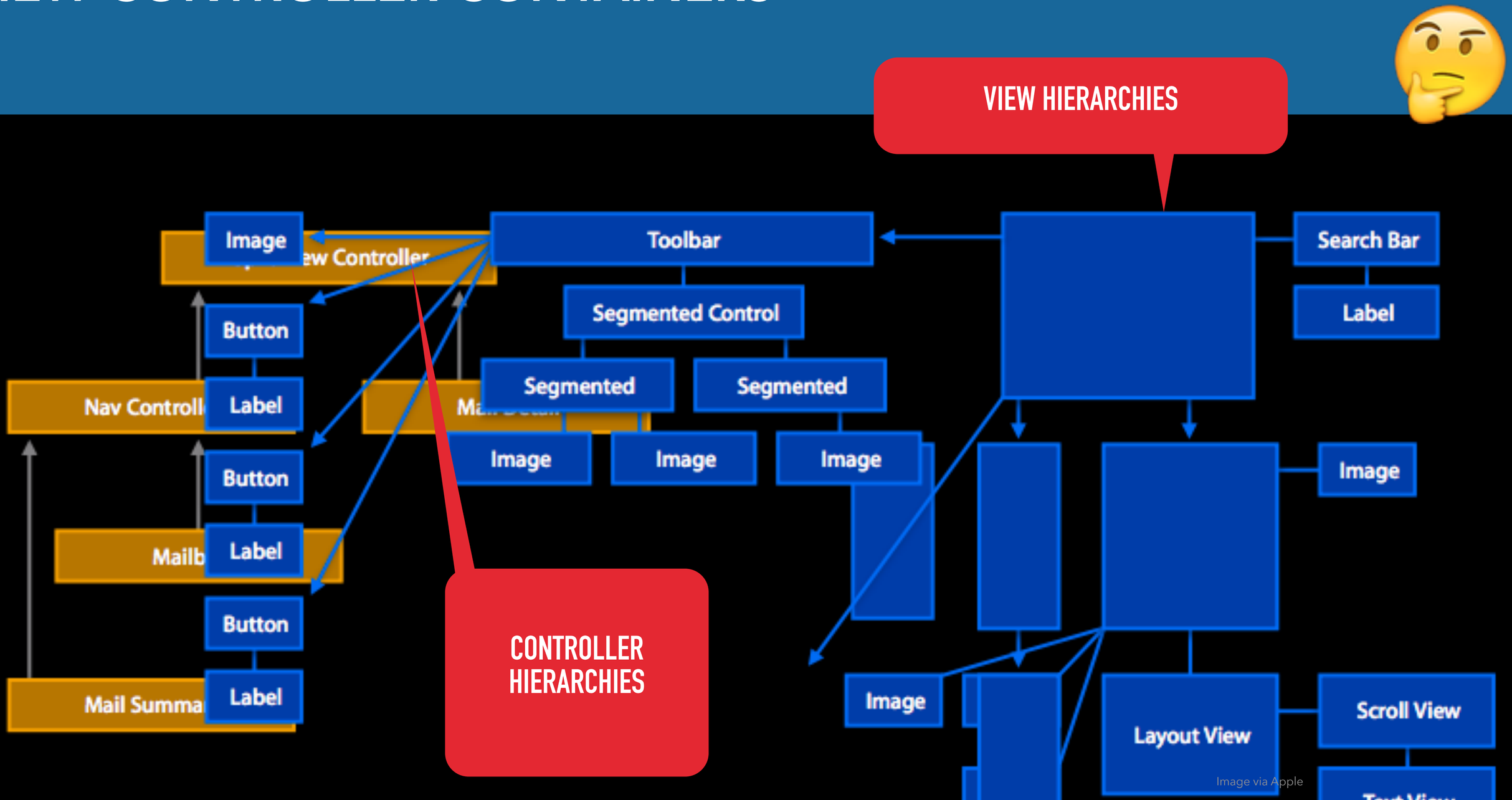


Image via Apple

VIEW CONTROLLER CONTAINERS



VIEW CONTROLLER CONTAINERS

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

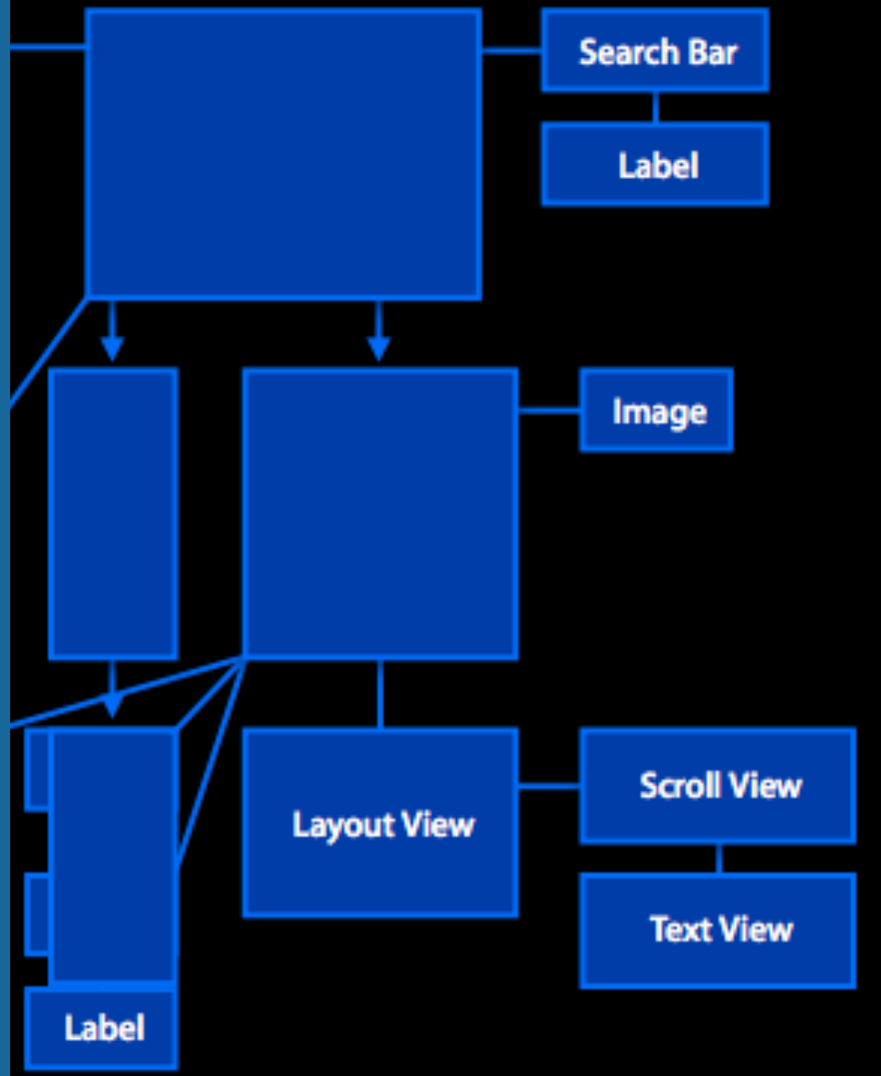


Image via Apple

VIEW CONTROLLER CONTAINERS

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

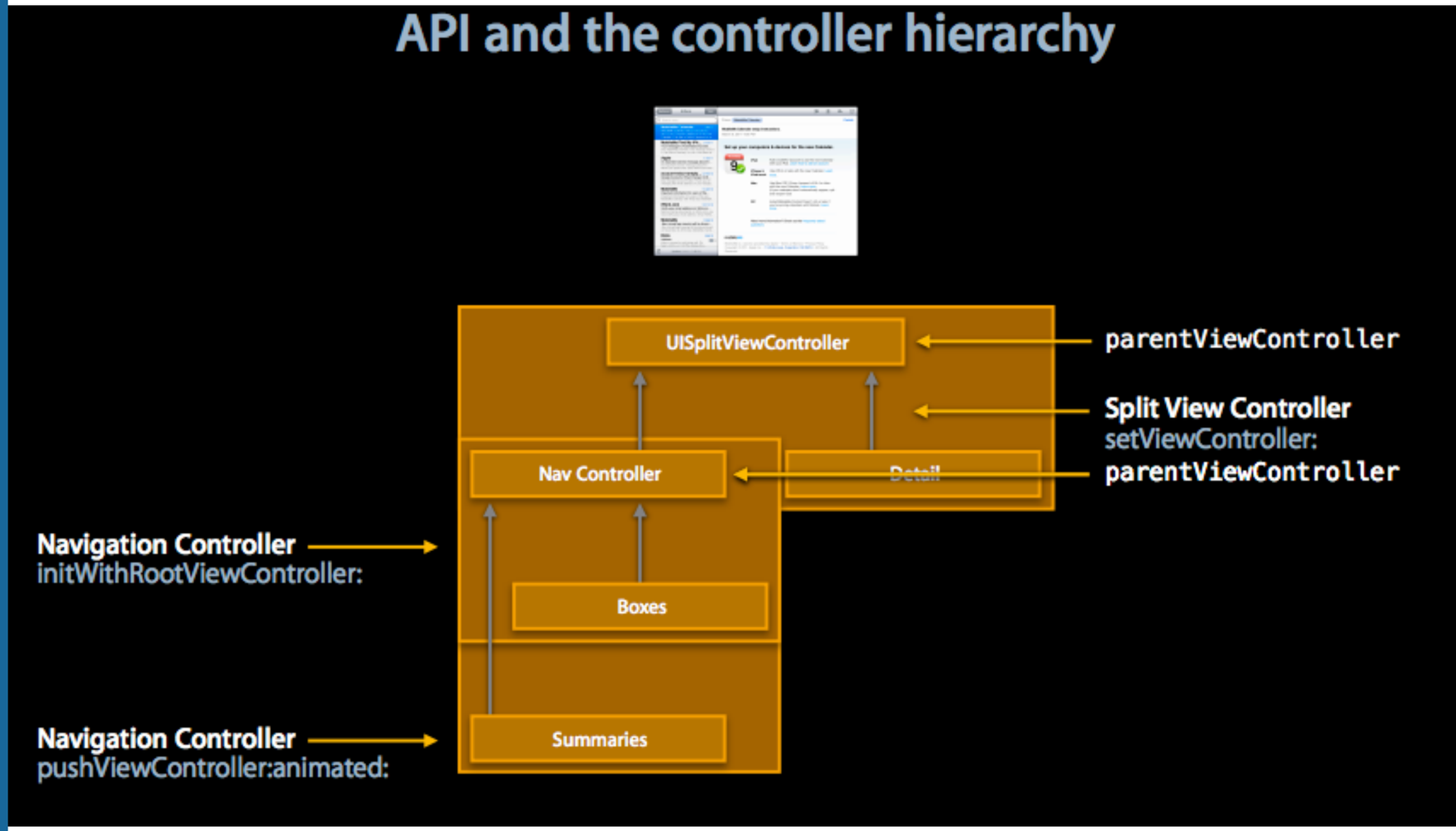


Image via Apple

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

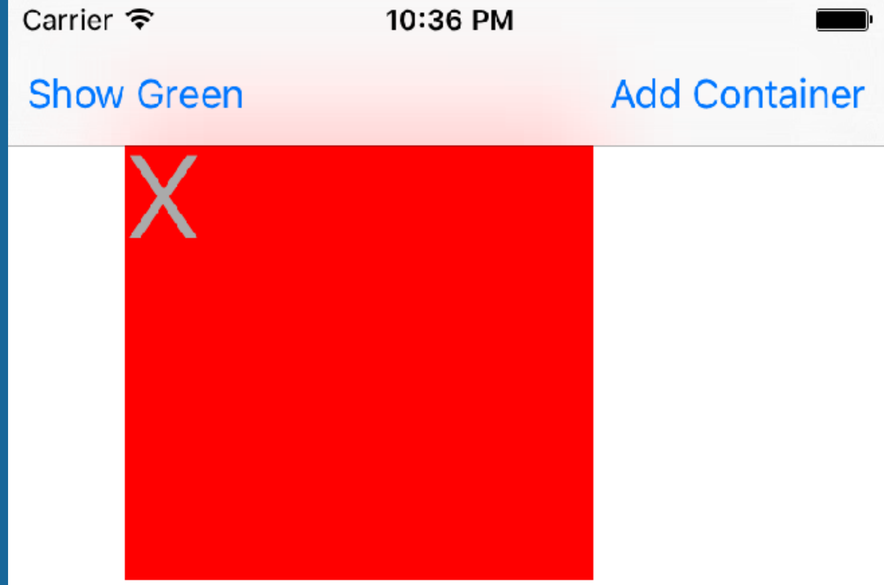
3. Call the child's didMoveToParentViewController: method to signal that the operation is complete

```
// 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)
```



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

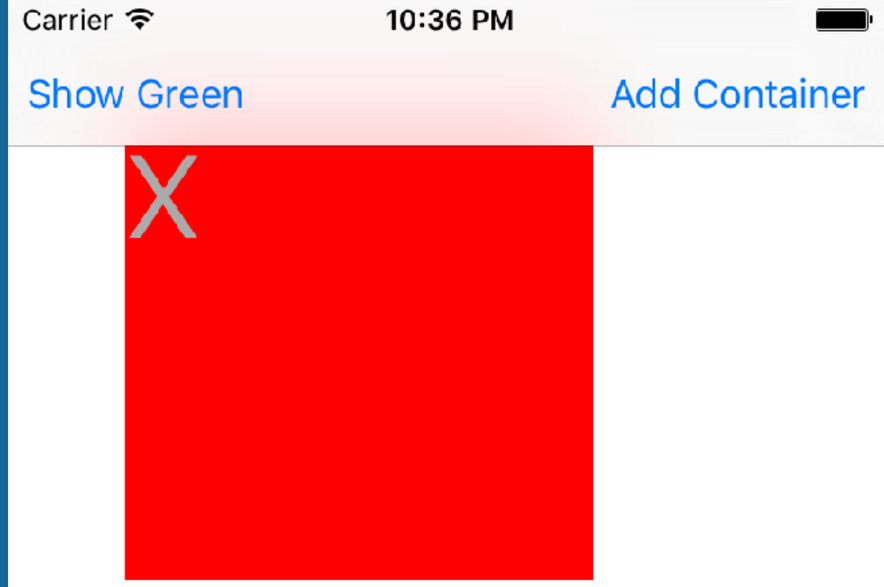
```
// 1. Calls the child's willMoveToParentViewController: method
// parameter of nil to tell the child that it is being removed
sender.willMoveToParentViewController(nil)

// 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



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: - Life
  //
```

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

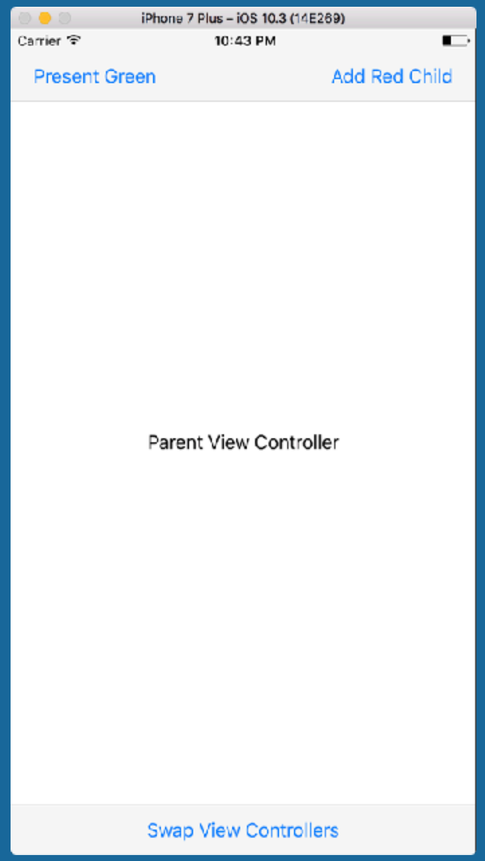
- More versatile implementation
- Determine if under containment or content view controller

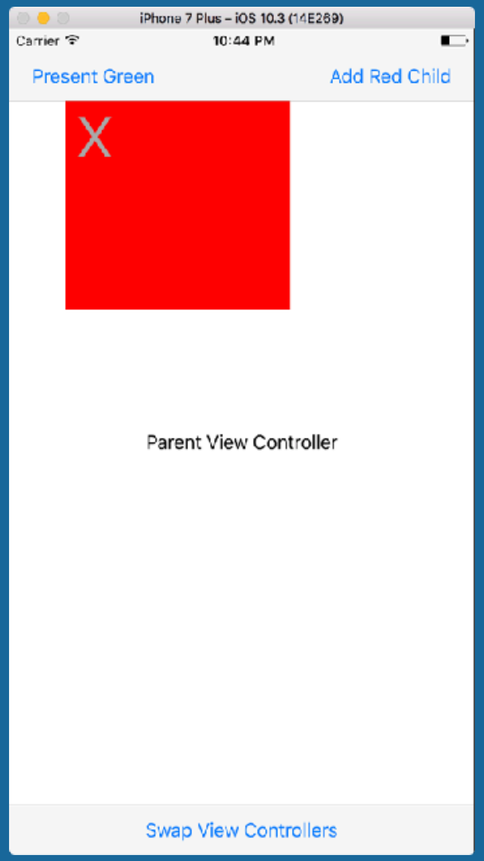
```
///
protocol ColorViewControllerDelegate: class {...}


///
/// 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() {...}



}
```

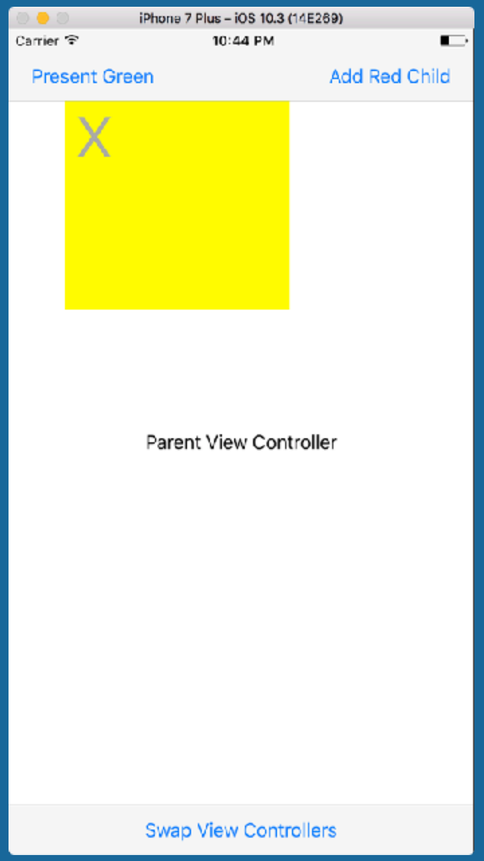
CHILD VIEW CONTROLLER VIEW TRANSITIONS

VIEW CONTROLLER CONTAINERS

- Transitioning between children view controllers

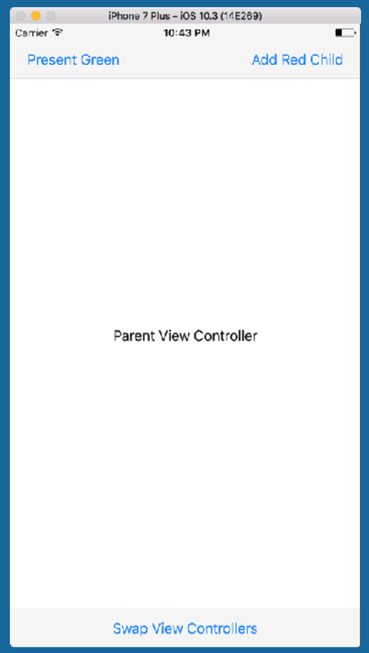






VIEW CONTROLLER CONTAINERS

- Transitioning between children view controllers

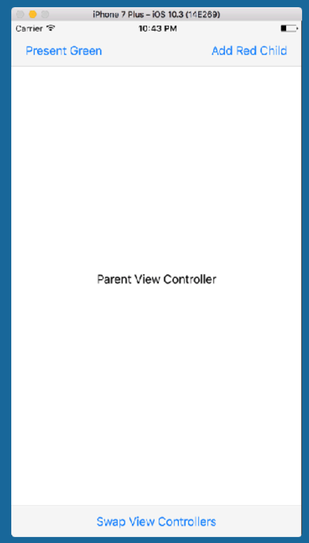


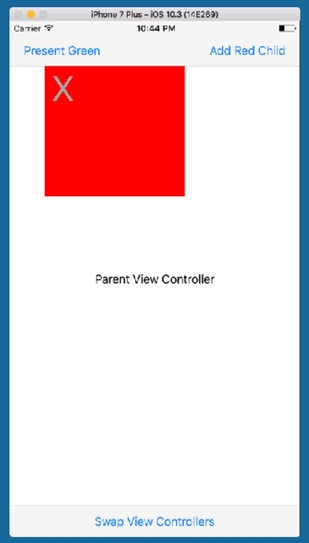

```
@IBAction func tapSwapViewControllers(_ sender: UIBarButtonItem) {
  // Create a new view controller (yellow color)
  let gvc = ColorViewController(nibName: "GreenViewController", bundle: nil)
  gvc.delegate = self
  gvc.view.backgroundColor = UIColor.yellow
  gvc.view.frame = CGRect(x: 50, y: 50, 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")
  }
}
```

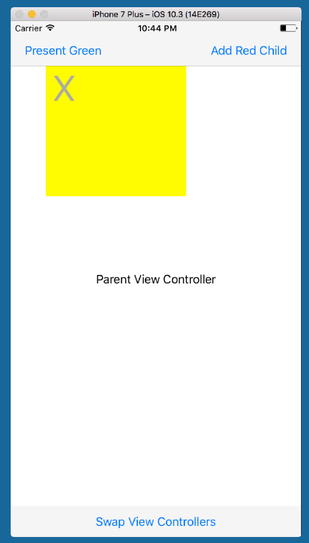
VIEW CONTROLLER CONTAINERS

CONTAINER VIEW

- Make you own tab bar







-

BUTTONS

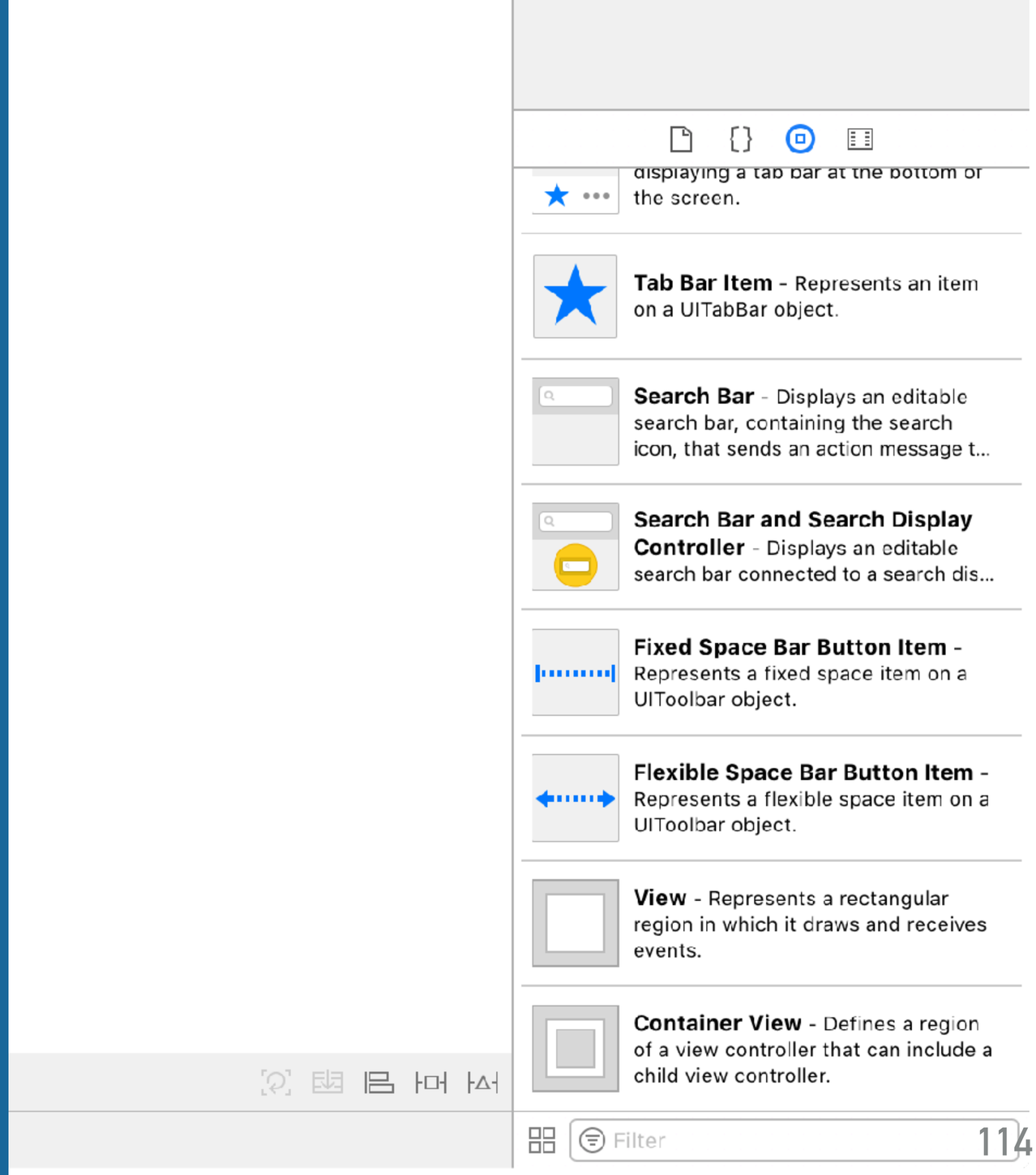
HOME. USERS. SETTINGS.

VIEW CONTROLLER CONTAINMENT USING INTERFACE BUILDER

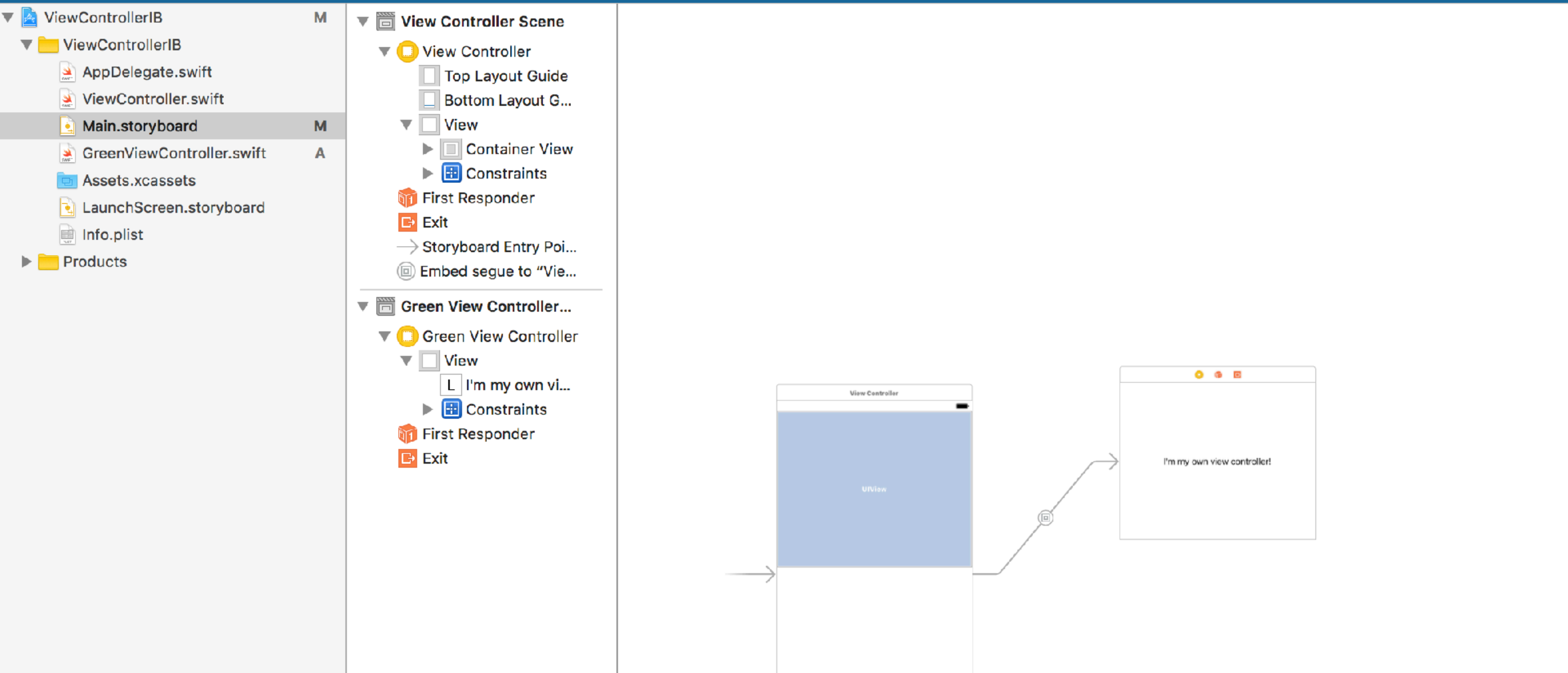
VIEW CONTROLLER CONTAINMENT USING INTERFACE BUILDER

SUBTITLE

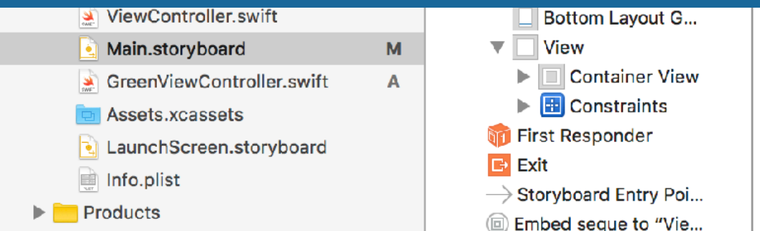
- Add Container Views to a View Controller in Interface Builder
- Automatically performs the parent-child dance for you

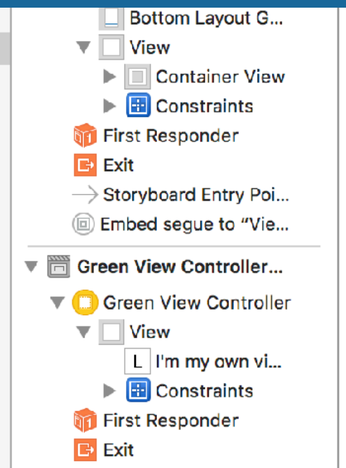


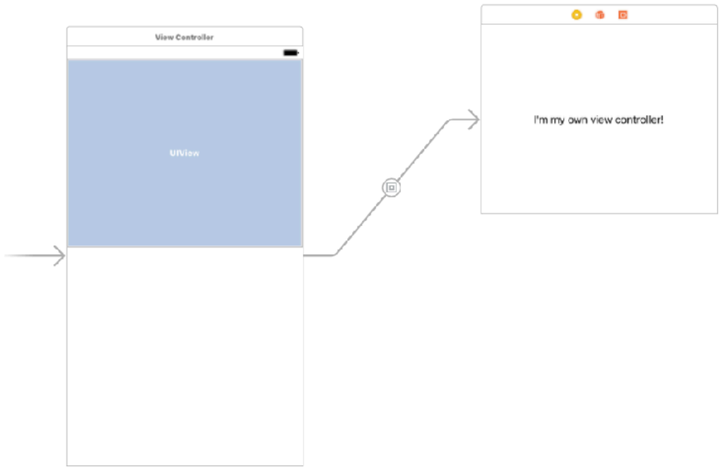
VIEW CONTROLLER CONTAINMENT USING INTERFACE BUILDER

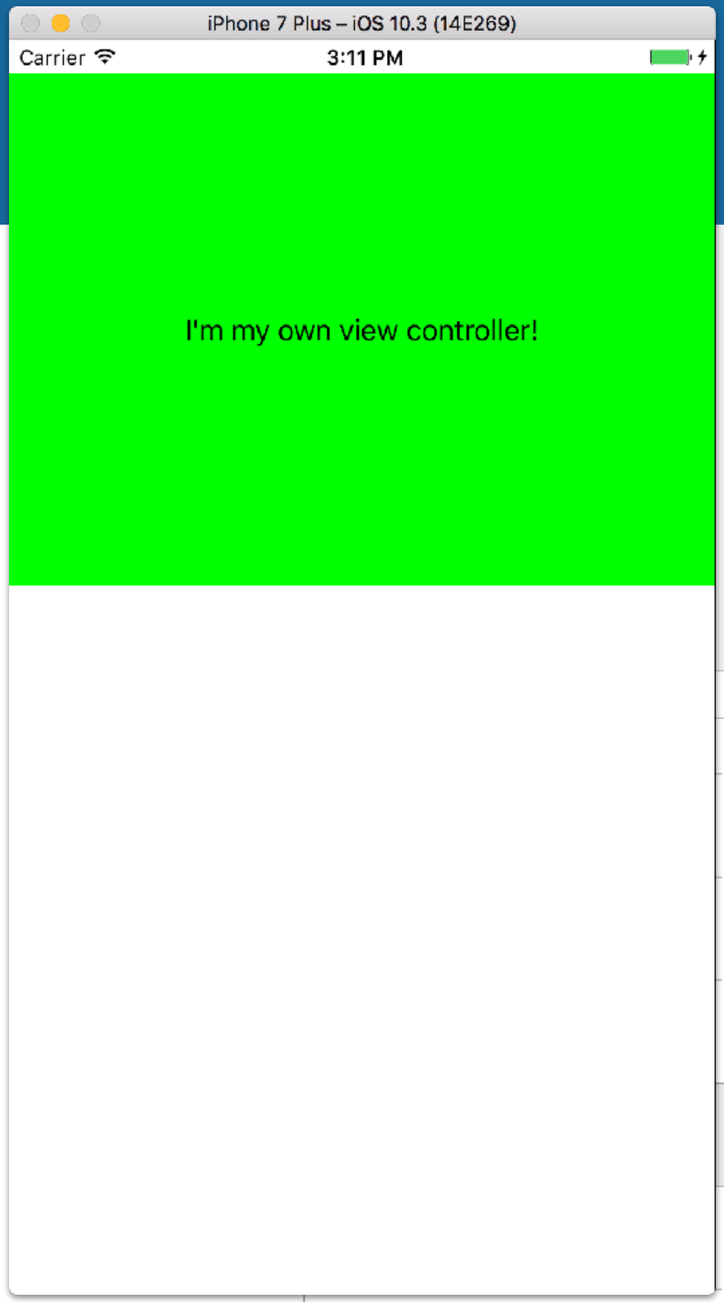


VIEW CONTROLLER CONTAINMENT USING INTERFACE BUILDER









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 is 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

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1A

SESSION 1B

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1A

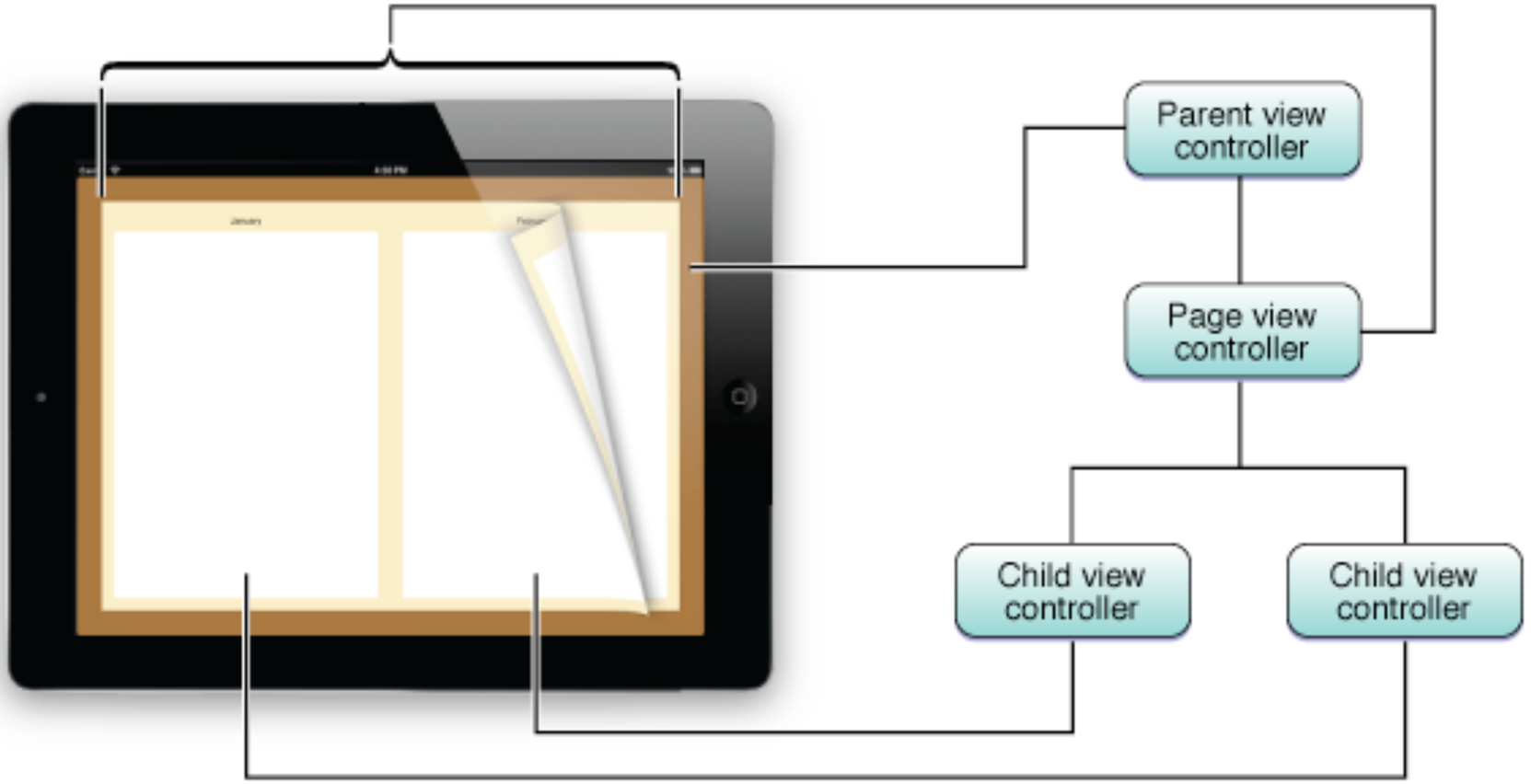
PAGE VIEW CONTROLLER

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

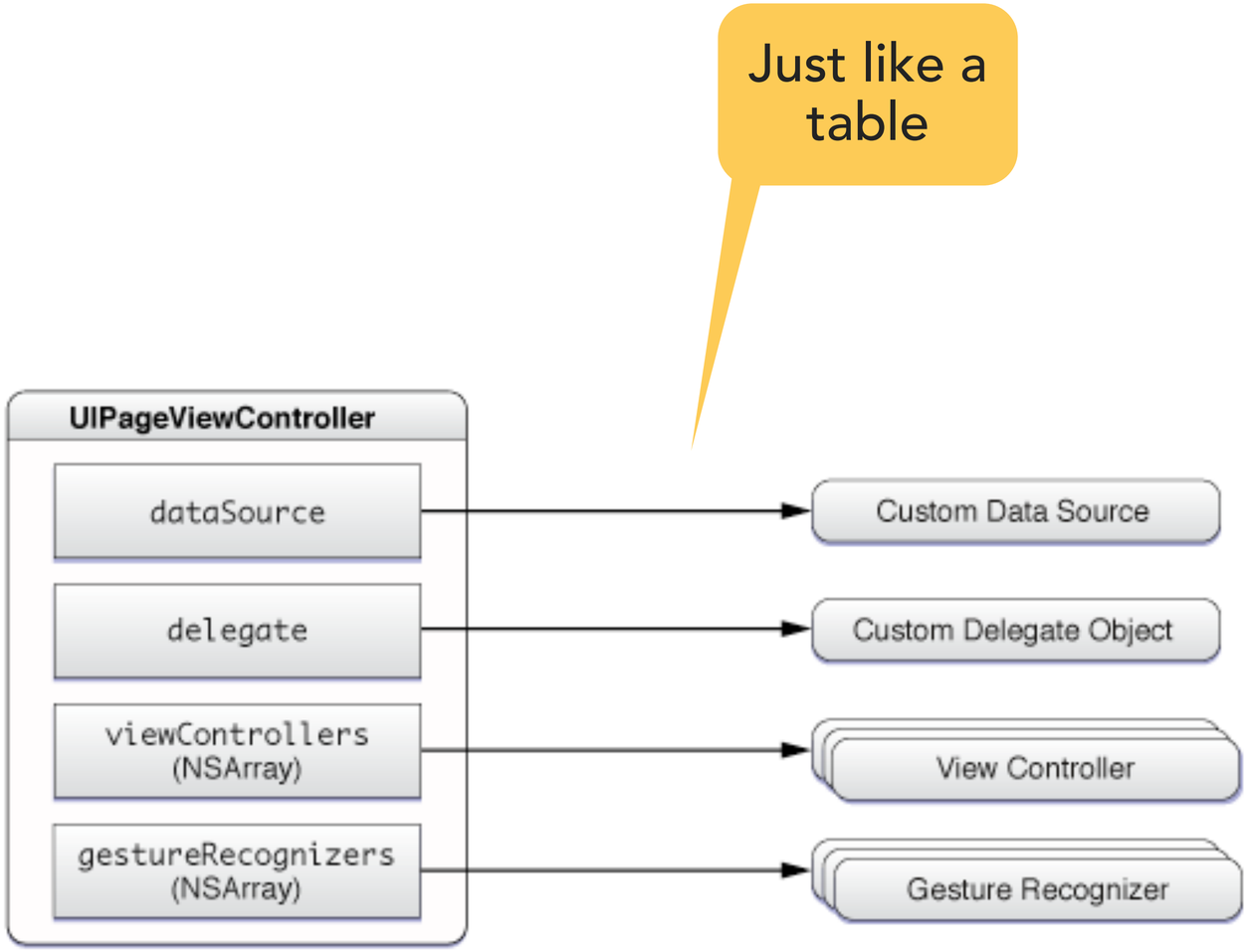
PAGE VIEW CONTROLLER

- 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

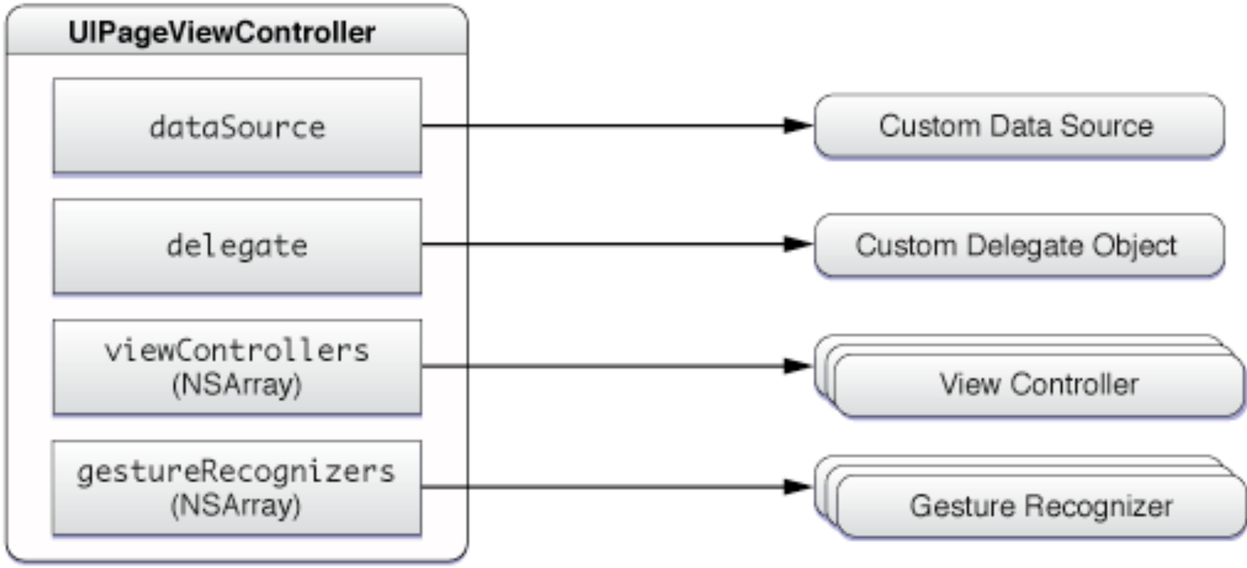
- A page view interface consists of the following objects
  - An optional delegate
  - An optional data source
  - An array of the current view controllers
  - An array of gesture recognizers



PAGE VIEW CONTROLLER

UIPAGEVIEWCONTROLLER API

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

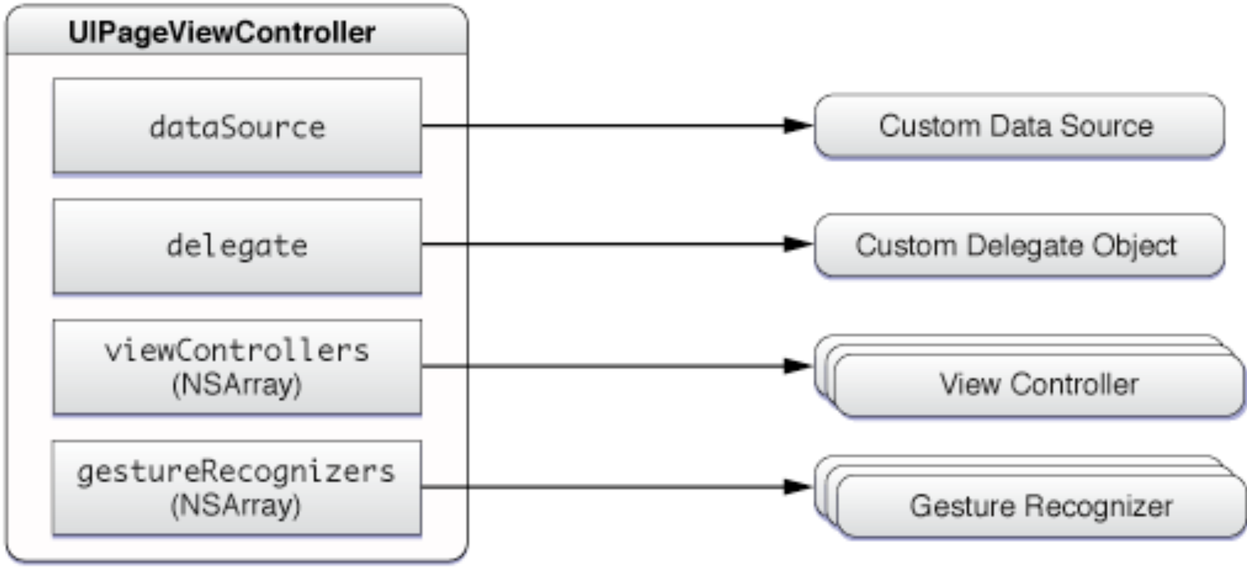


Reusing view controllers similar to table cells

PAGE VIEW CONTROLLER

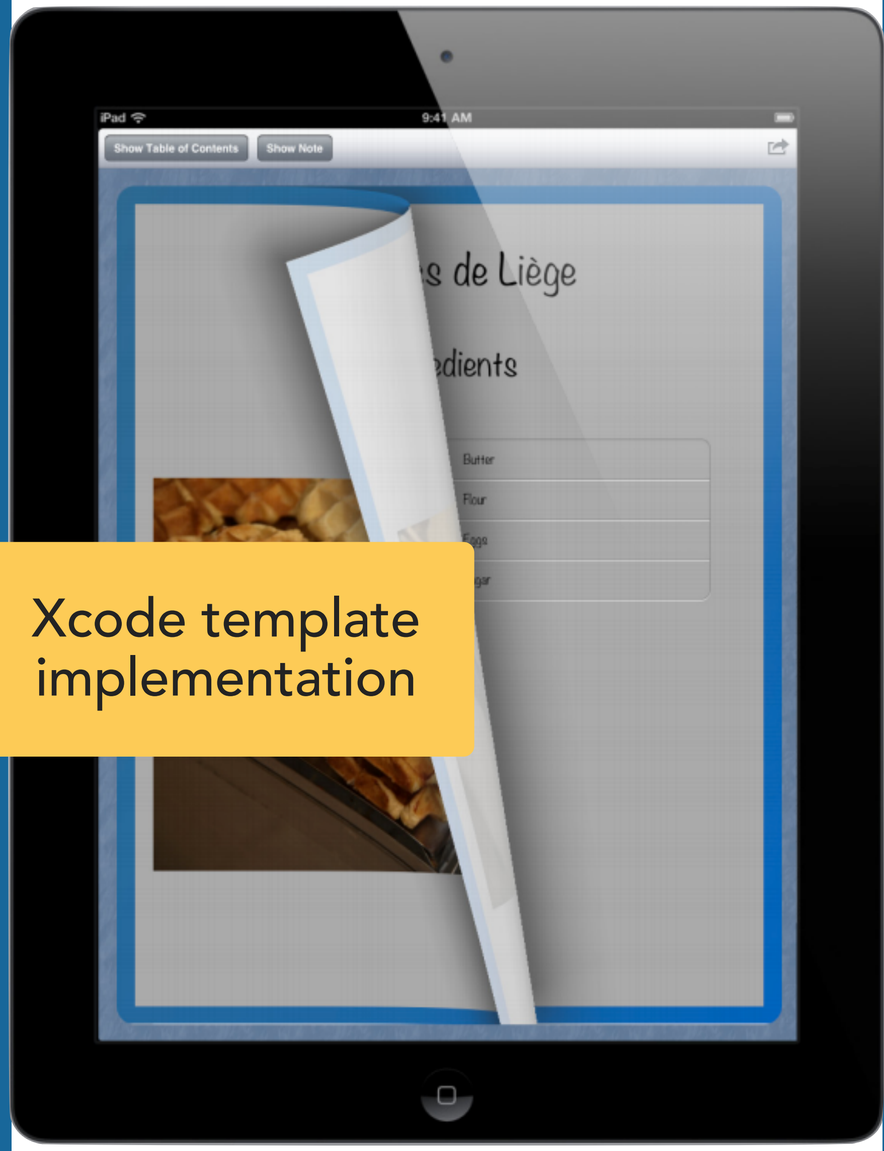
UIPAGEVIEWCONTROLLER API

- 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
```

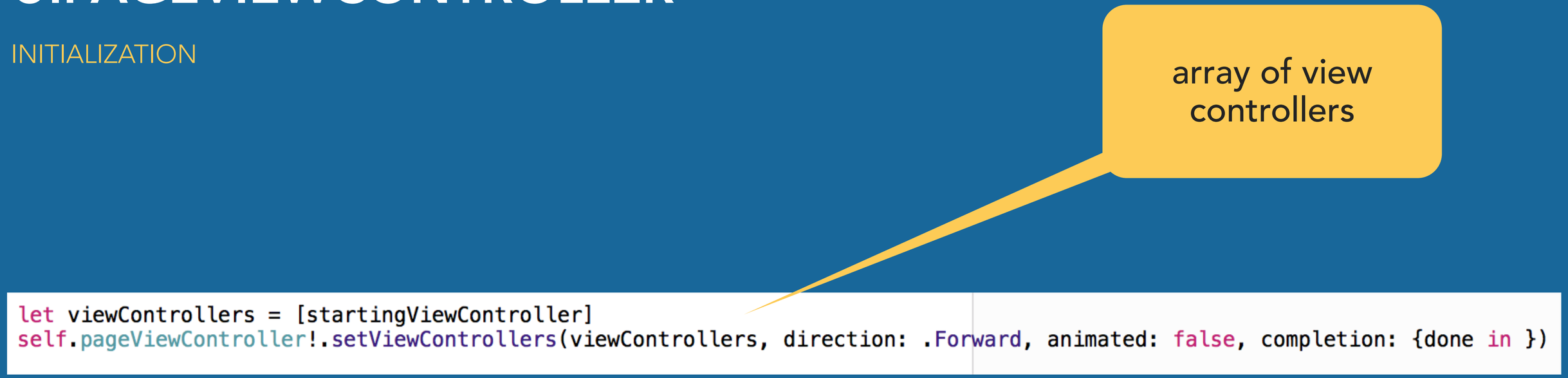
page flip direction

page spine

- Initialization

UIPAGEVIEWCONTROLLER

INITIALIZATION



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

UIPAGEVIEWCONTROLLER

INITIALIZATION

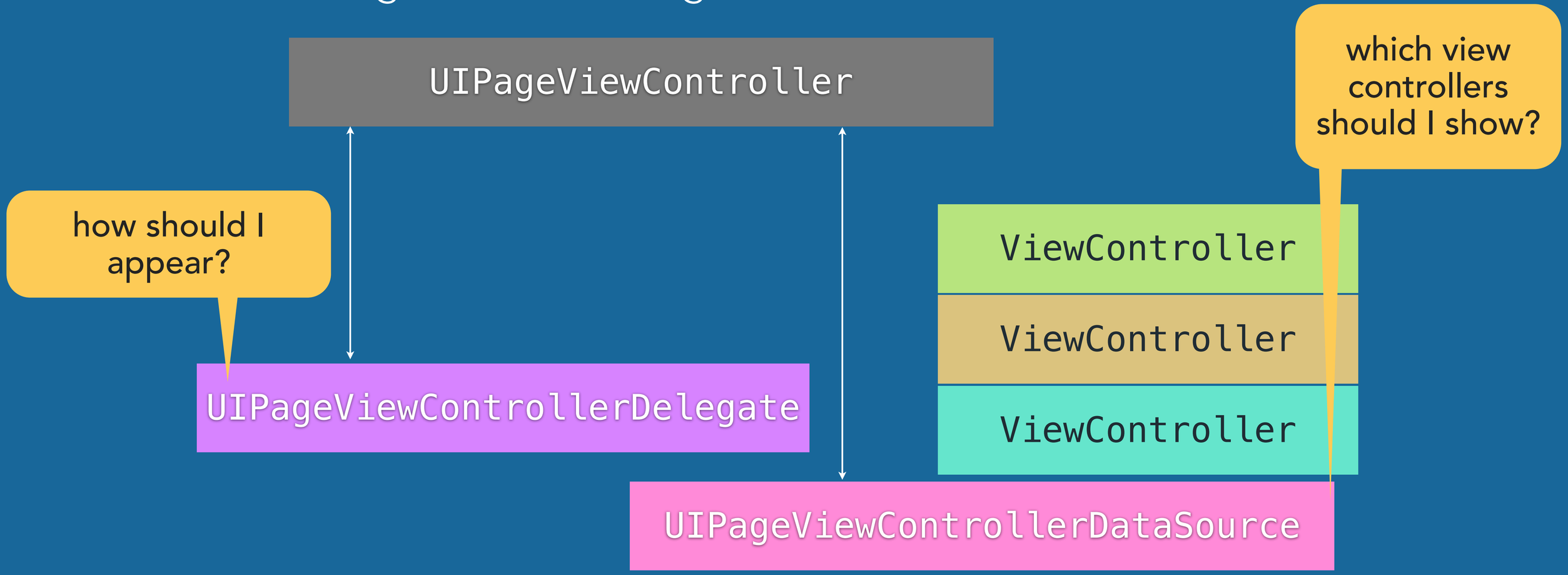
```
let viewControllers = [currentViewController]
self.pageViewController!.setViewControllers(viewControllers, direction: .Forward, animated: true, completion: {done in })
```

- 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

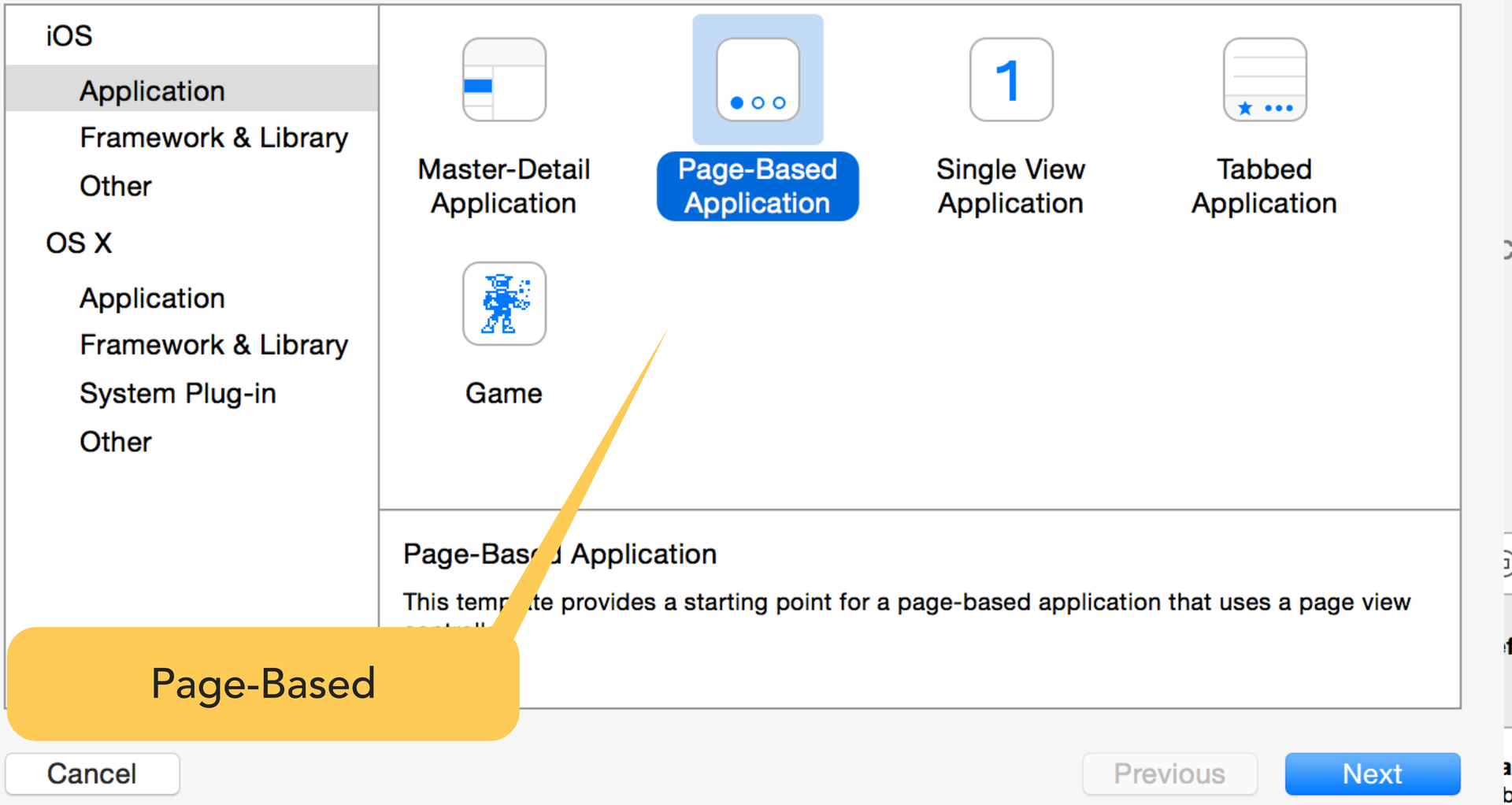


A SIMPLE PAGE VIEW CONTROLLER EXAMPLE

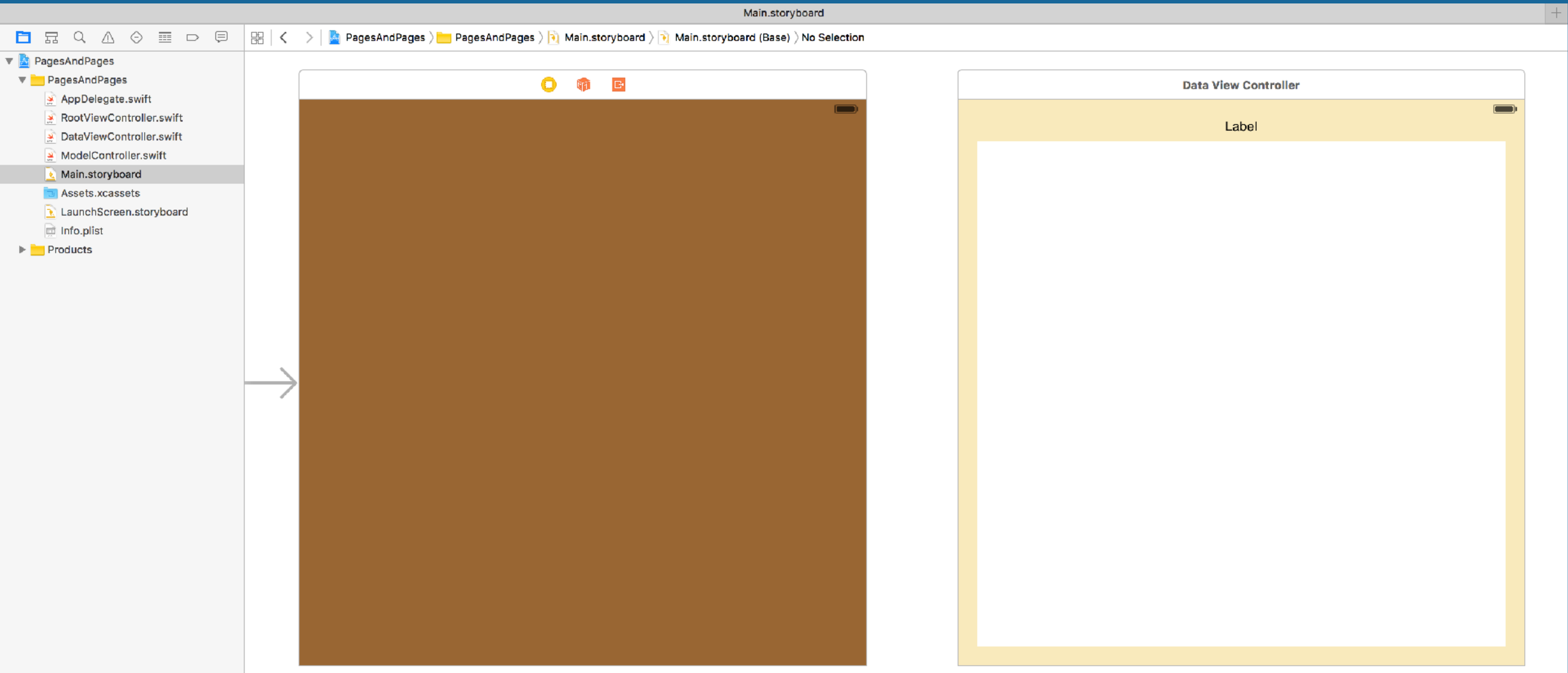
ONCE UPON A TIME...

PAGE VIEW CONTROLLER TEMPLATE

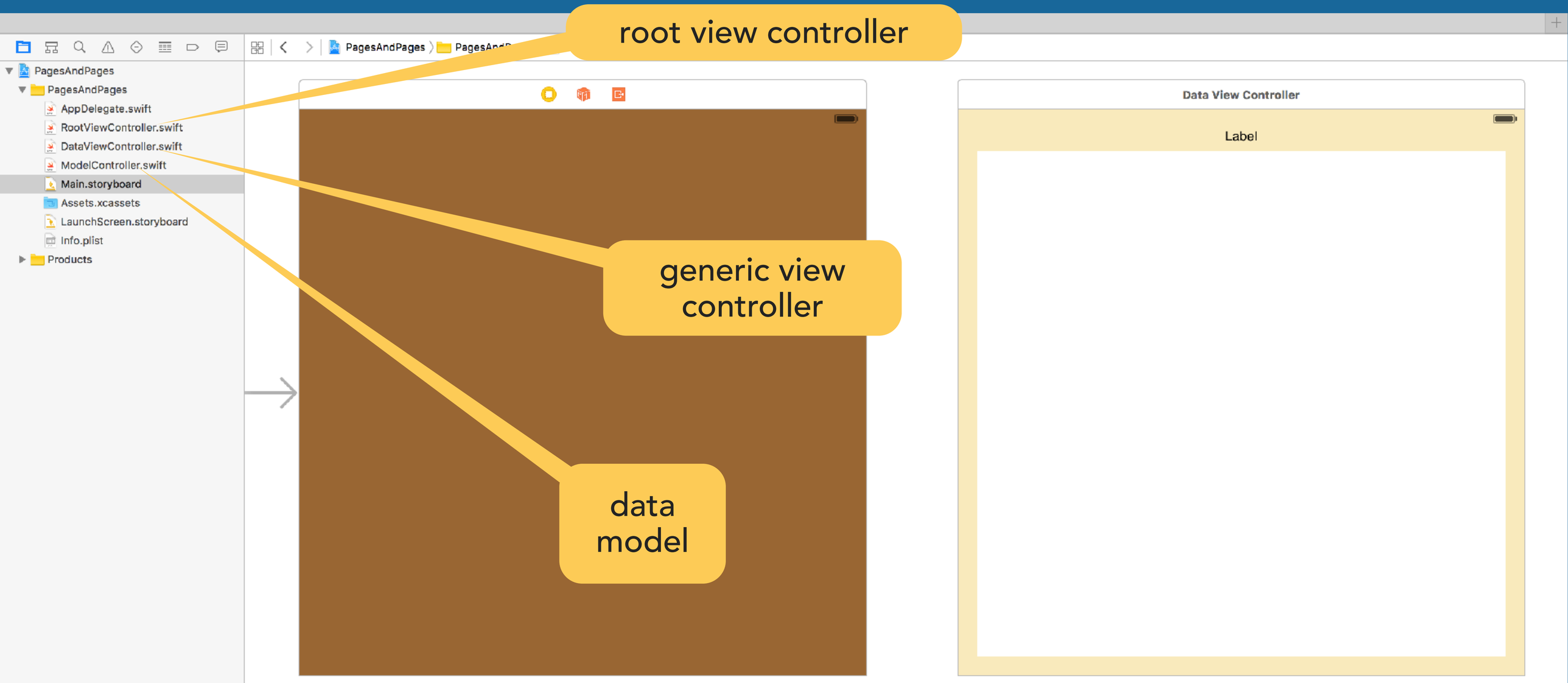
PAGE VIEW CONTROLLER TEMPLATE



PAGE VIEW CONTROLLER TEMPLATE

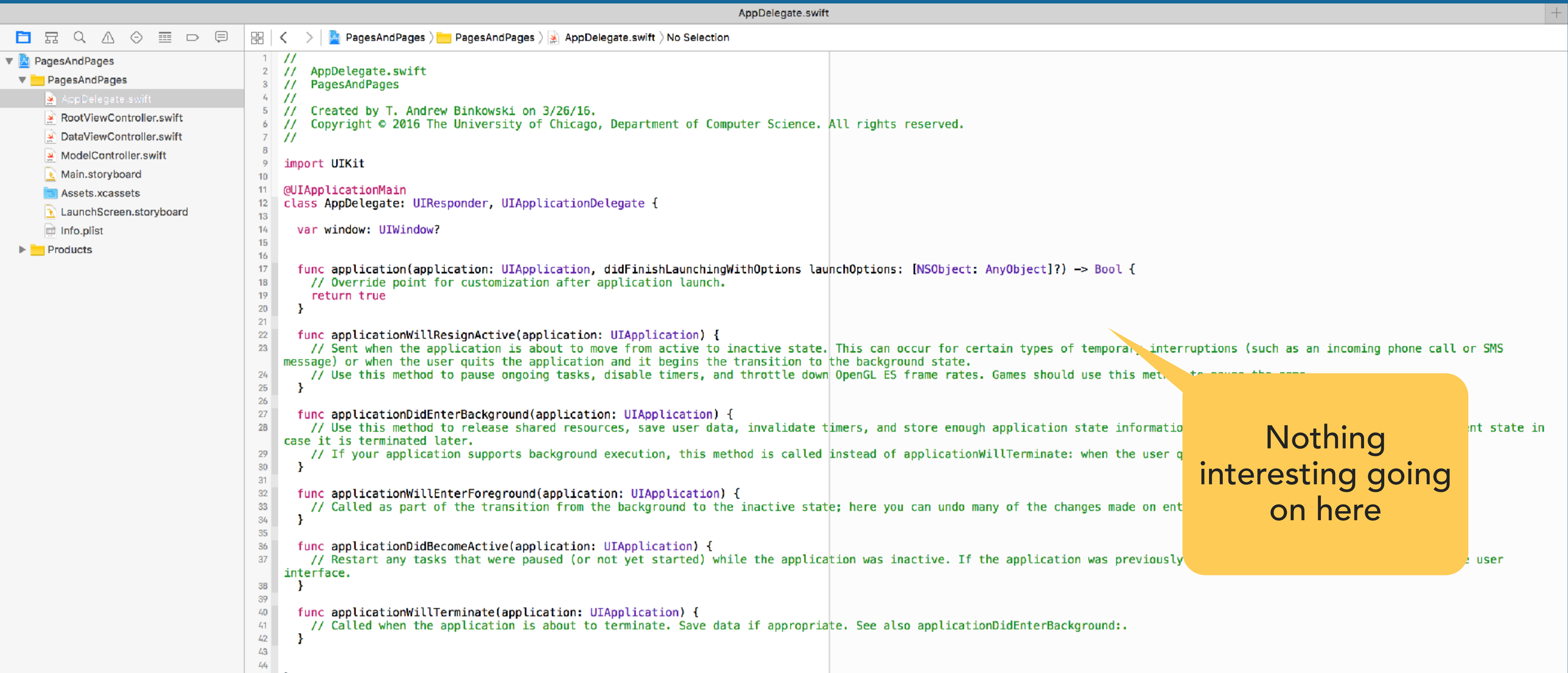


PAGE VIEW CONTROLLER TEMPLATE



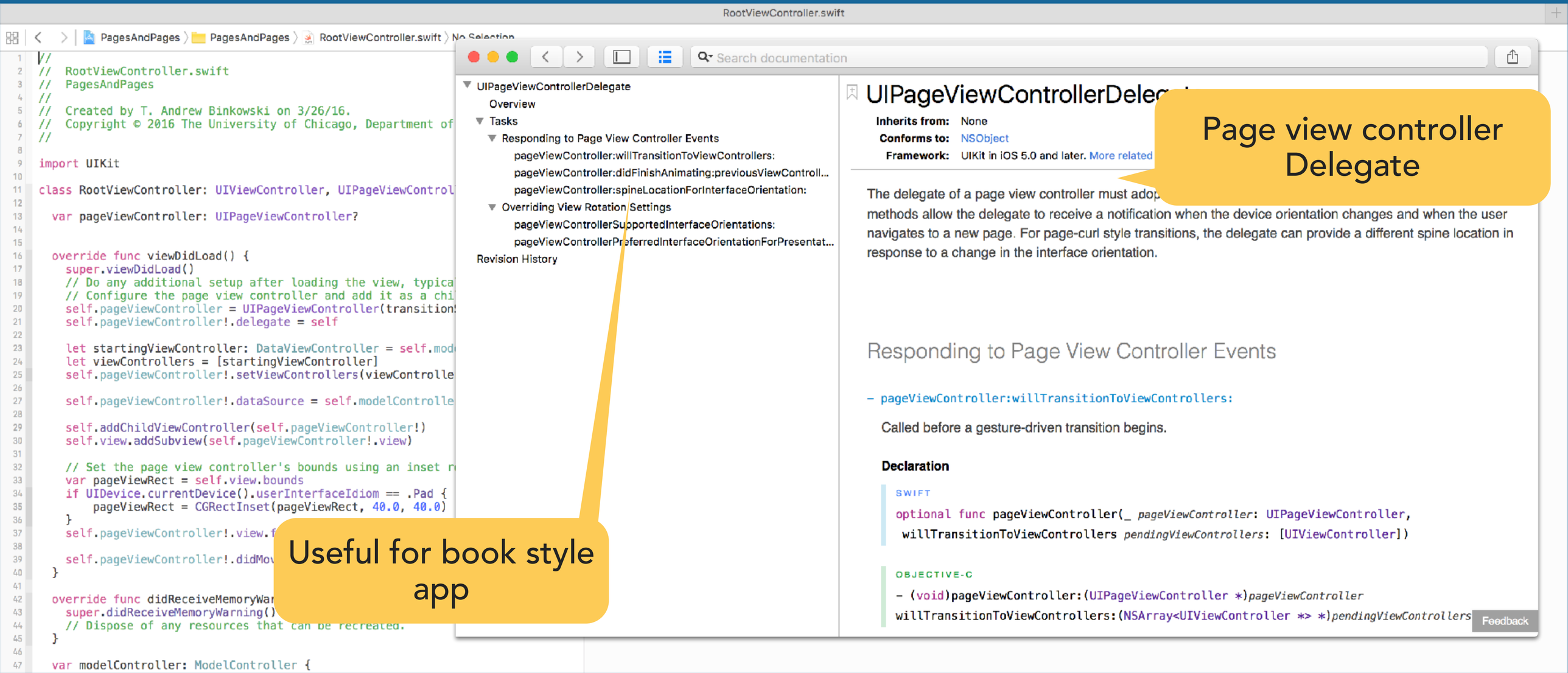
PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER



PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER



PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER

Initialize the page 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)
}
```

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
  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

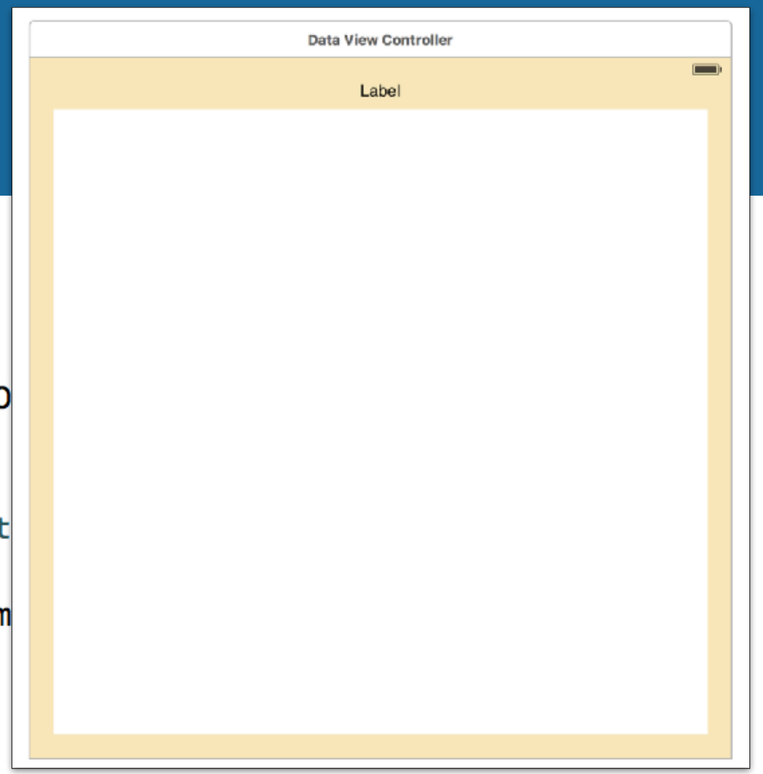
```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
    // Configure the page view controller and add it as a child view controlle
    self.pageViewController = UIPageViewController(transitionStyle: .pageCurl,        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

  let startingViewController: DataViewControll                    rollerAt
  let viewControllers = [startingViewControlle
  self.pageViewController!.setViewControllers(                    rd, anim

  self.pageViewController!.dataSource = self.m

  self.addChildViewController(self.pageViewCo troller!)
  self.view.addSubview(self.pageViewControll r!.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)
}
```

create colored inset
(eye candy)

PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER

```
func pageViewController(pageViewController: UIPageViewController, spineLocationForInterfaceOrientation orientation: UIInterfaceOrientation) -> UIPageViewControlle

if (orientation == .Portrait) || (orientation == .PortraitUpsideDown) || (UIDevice.currentDevice().userInterfaceIdiom == .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 currentViewController = self.pageViewController!.viewControllers![0]
    let viewControllers = [currentViewController]
    self.pageViewController!.setViewControllers(viewControllers, direction: .Forward, animated: true, completion: {done in })

    self.pageViewController!.doubleSided = false
    return .Min
}

// In landscape orientation: Set 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 currentViewController = 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
}
```

show only one page (portrait)

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 pageViewCo
 It also implements a custom method, viewControllerAtIndex: which is useful in the implementation of the d                    he initial configuration of the

 There is no need to actually create view controllers for each page in advance -- indeed doing so incurs u                    he data model, these methods cr
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 = NSDateFormatter()
    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                                                    obje
    return pageData.indexOf(viewController.dataObject) ?? NSNotFound
  }

  // MARK: - Page View Controller Data Source

  func pageViewController(pageViewController: UIPageViewController, viewControllerBeforeViewController viewController: UIViewControlle                                                    {...}

  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) ->
    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) -> U
    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() {...}

  func viewControllerAtIndex(index: Int, storyboard:                    troller? {

  func indexOfViewController(viewController: DataView

  // 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
      }
```

EXTERNAL NAME

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

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

PagesAndPages 〉 Pa AndPages 〉 RootViewController.swift 〉 RootViewController

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

import UIKit

class RootViewController: UIViewController, UIPageViewControllerDelegate {

  var pageViewController: UIPageViewController?


  override func viewDidLoad() {...}

  override func didReceiveMemoryWarning() {...}

  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


  //
  // MARK: - UIPageViewController delegate methods
  //

  func pageViewController(pageViewController: UIPageViewController, spineLocationForInterfaceOrien


}
```

```swift
//
//  Created by T. Andrew Binkowski on 3/26/16.
//  Copyright © 2016 The University of Chicago, Department of Com
//

import UIKit

class RootViewController: UIViewController, UIPageViewControllerD

  var pageViewController: UIPageViewController?


  override func viewDidLoad() {...}

  override func didReceiveMemoryWarning() {...}

  lazy var modelController: ModelController = ModelController()


  //
  // MARK: - UIPageViewController delegate methods
  //

  func pageViewController(pageViewController: UIPageViewControlle


}
```

LAZY

Main.storyboard

PagesAndPages 〉 PagesAndPages 〉 RootViewController.swift 〉 RootViewController

```
//
//  RootViewController.swift
//  PagesAndPages
//
//  Created by T. Andrew Binkowski on 3/26/16
//  Copyright © 2016 T                                    All rights reserve
//

import UIKit

class RootViewControll

  var pageViewControll


  override func viewDid

  override func didReceiveMemoryWarning() {...}

  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


  //
  // MARK: - UIPageViewController delegate methods
  //

  func pageViewController(pageViewController: UIPageViewController, spineLocationForInterfaceOrien

}
```

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, UIPageViewControllerD

  var pageViewController: UIPageViewController?


  override func viewDidLoad() {...}

  override func didReceiveMemoryWarning() {...}

  lazy var modelController: ModelController = ModelController()


  //
  // MARK: - UIPageViewController delegate methods
  //

  func pageViewCo                                          PageViewControlle

}
```

LAZY IS ONLY EVALUATED ONCE

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1A

SESSION 1C

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1C

ASSIGNMENT 1

POP QUIZ

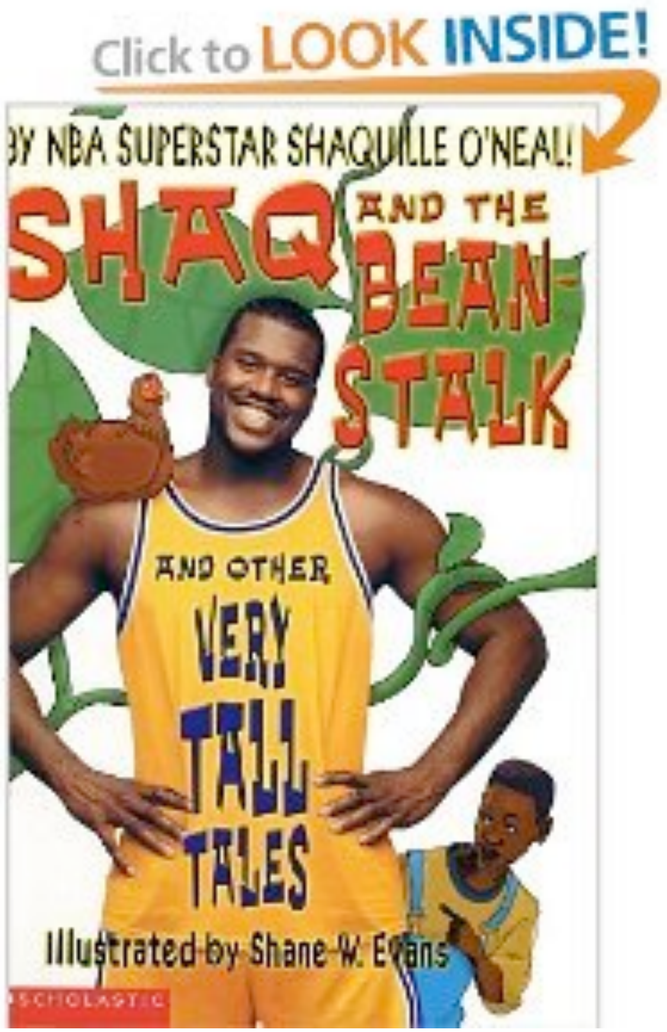
What do John Travollta, Barack Obama, Shaquille O'Neil, Whoopi Goldberg, Glenn Beck, Jay Leno, Cheech Marin, Jerry Seinfeld, Sting, Queen Latifah, Jimmy Buffett, Madonna, Spike Lee, Jerry Garcia, Paul McCartney, Jamie Lee Curtis, Bob Dylan, The Dutchess of York and Steve Martin have in common?

ASSIGNMENT 1

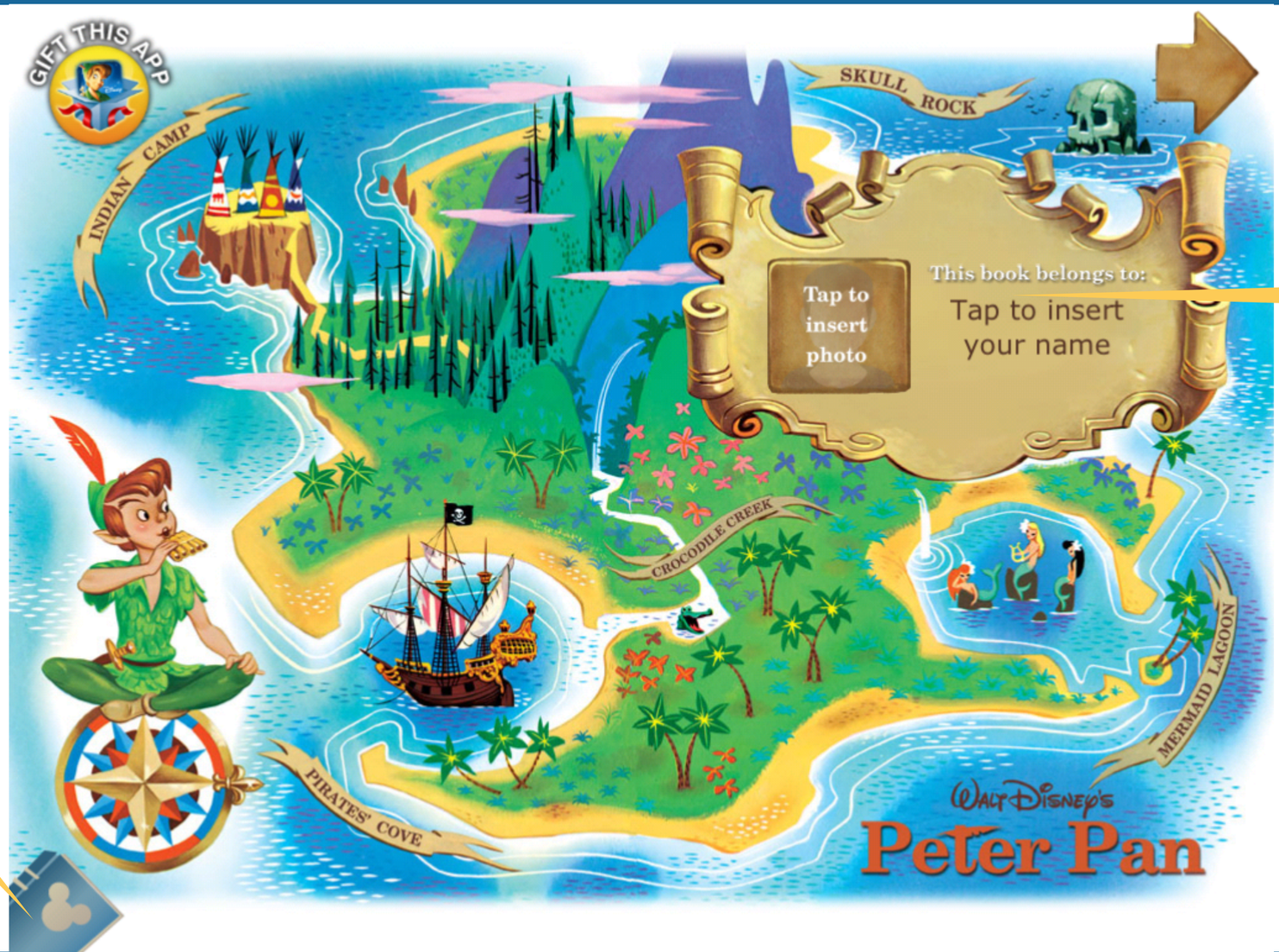
IF A CELEBRITY CAN DO IT, SO CAN YOU

SUBTITLE

- 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



Settings

About the author

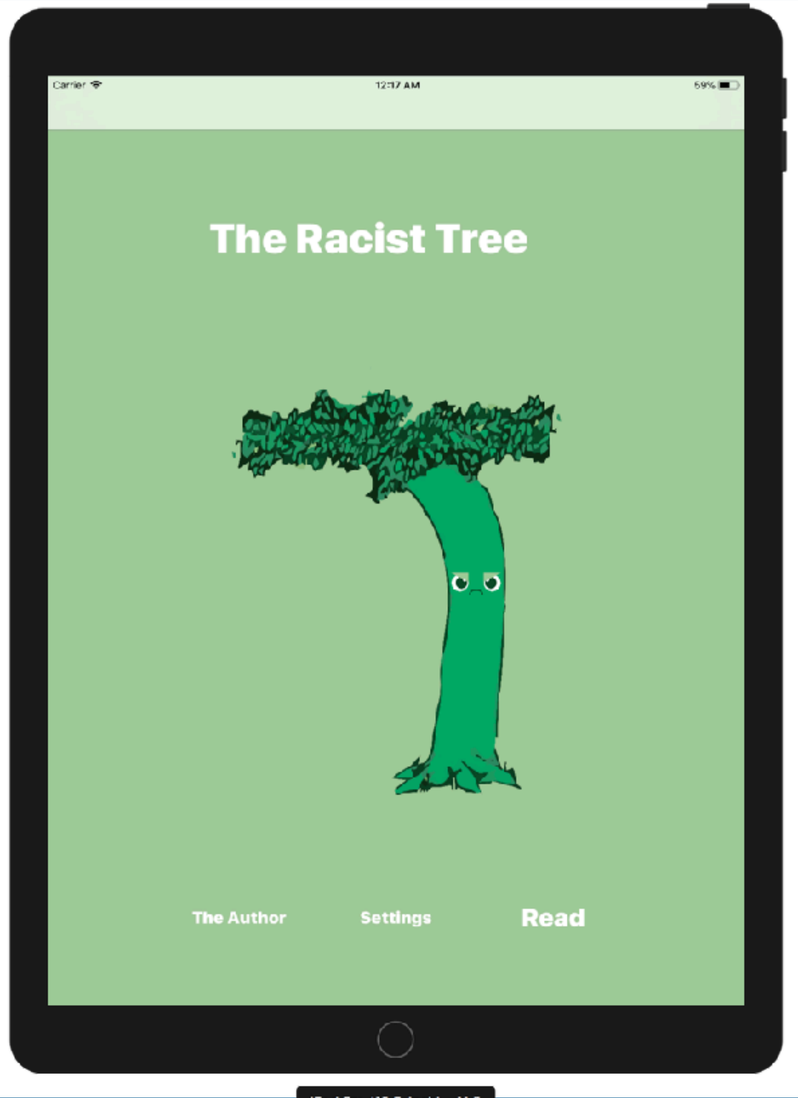
ASSIGNMENT 1

Page forward

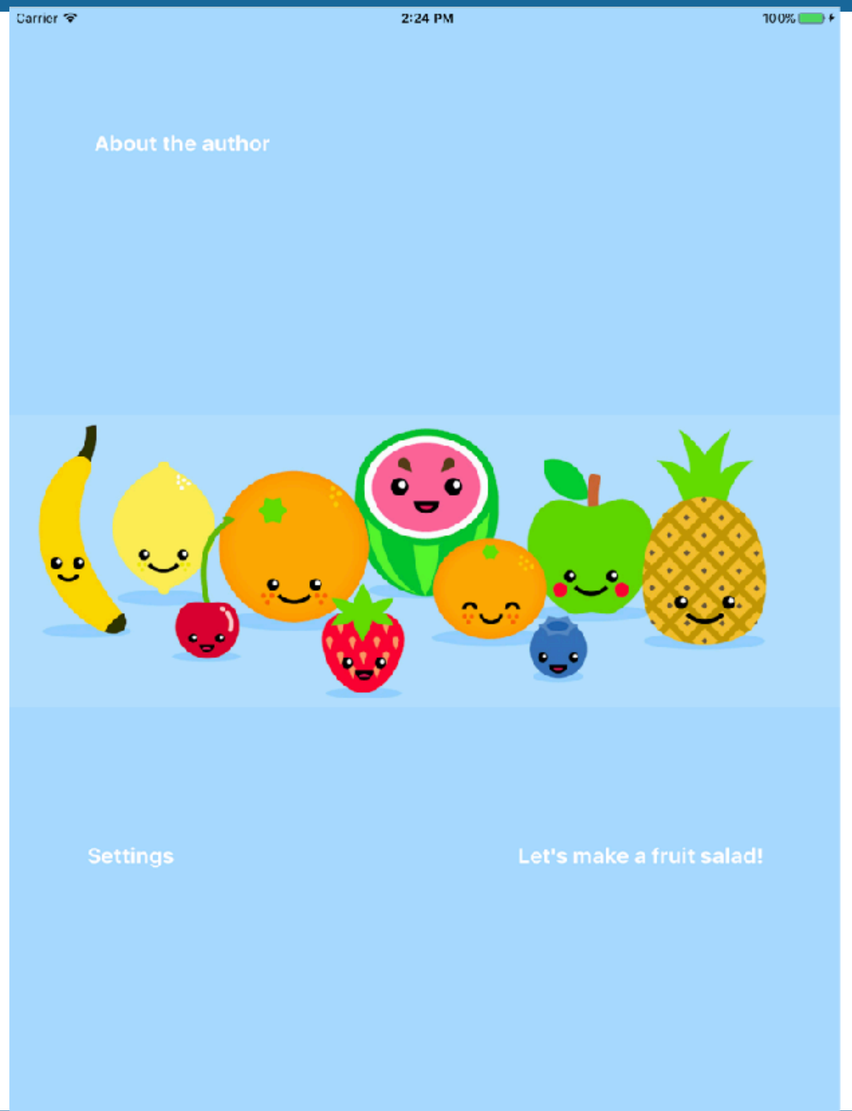
Page Backward

interactive element

ASSIGNMENT 1

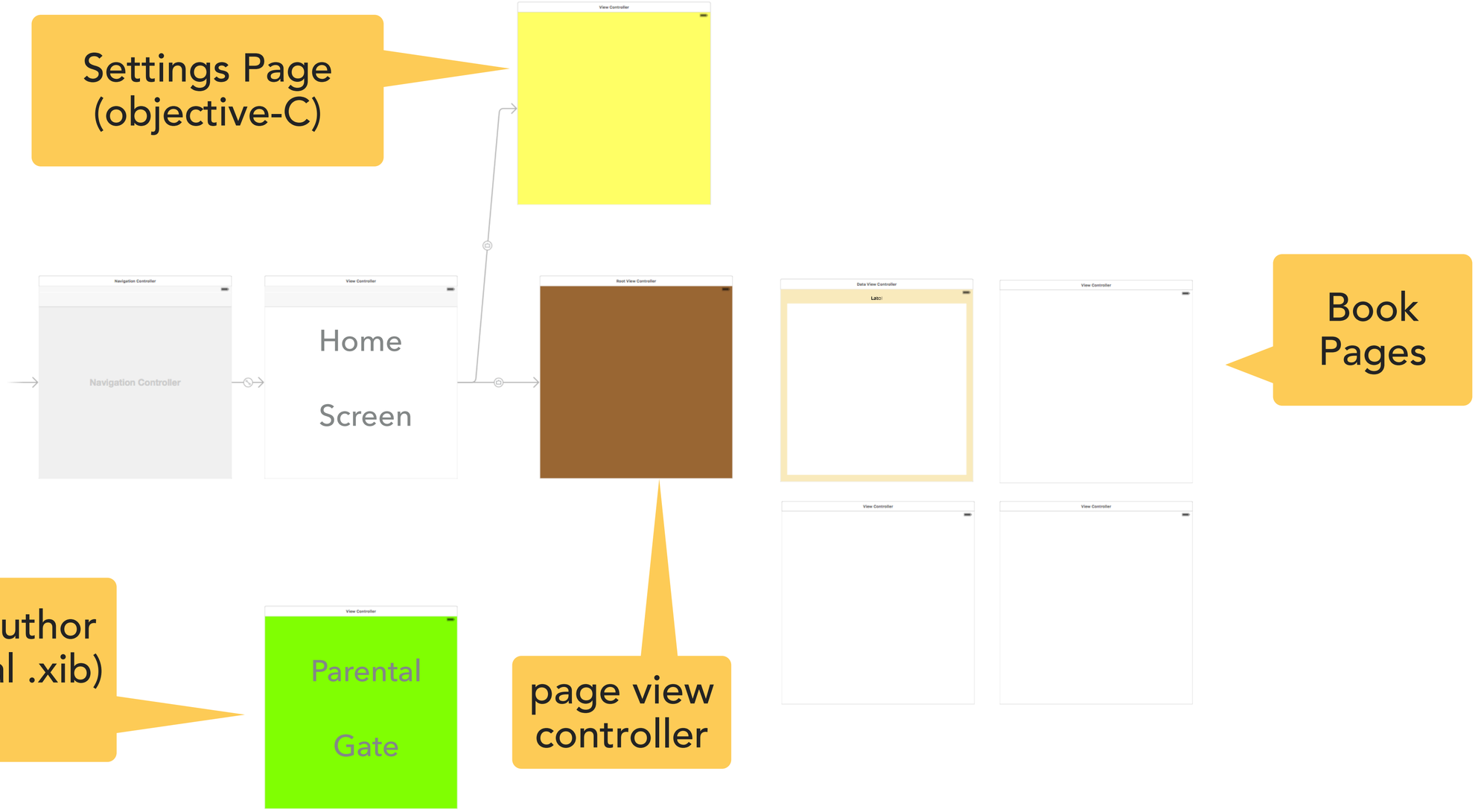


ASSIGNMENT 1



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

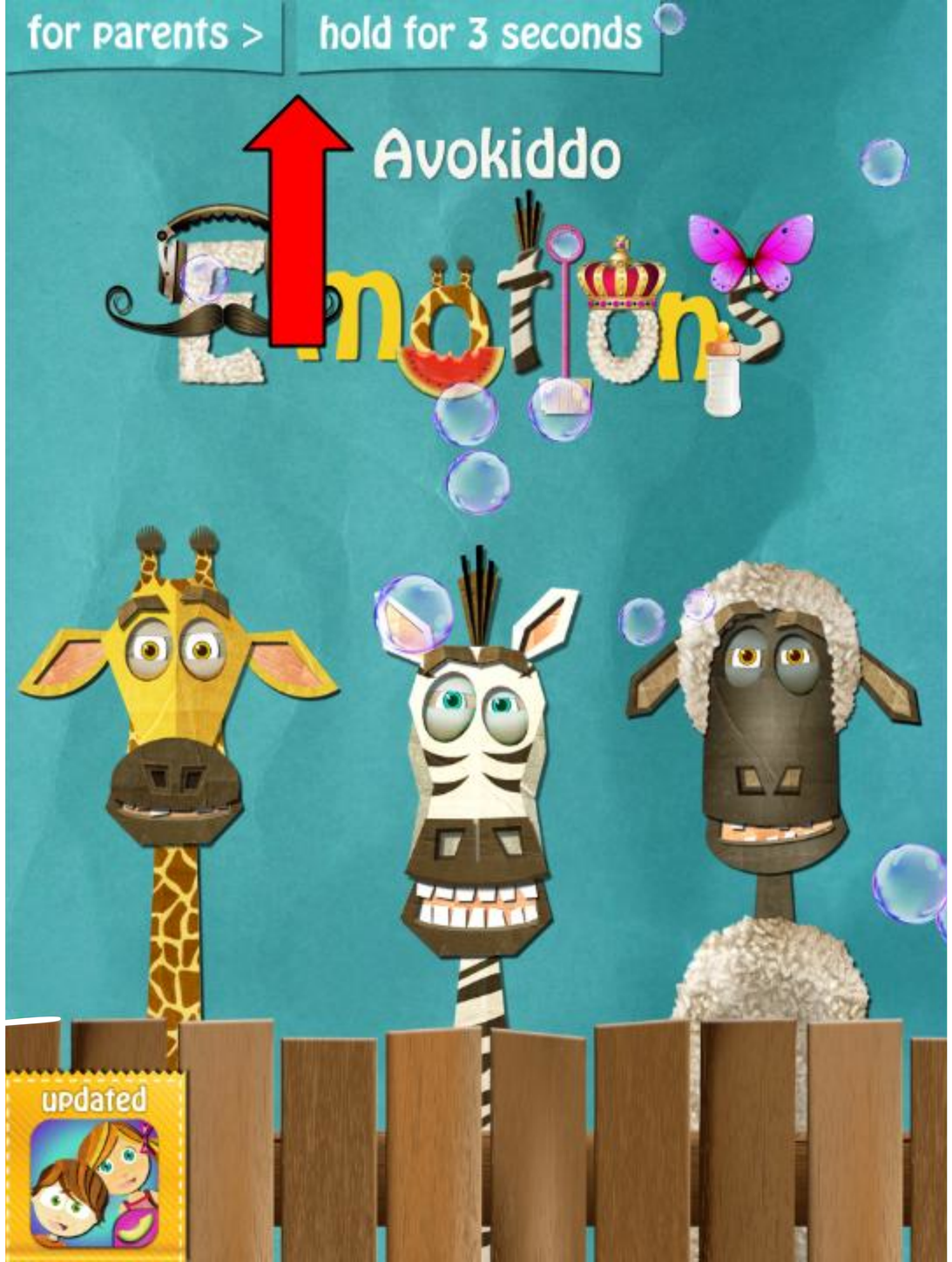
BOOK PAGES

- 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



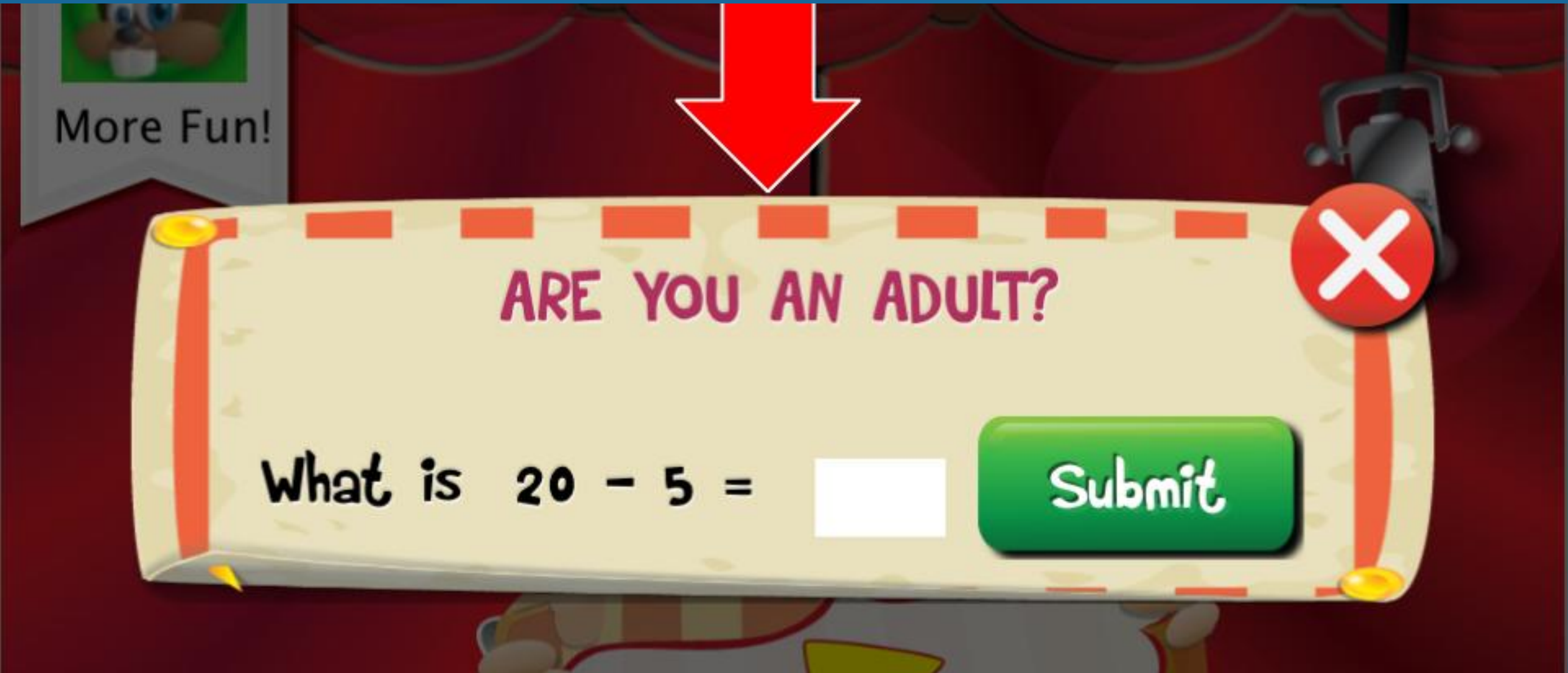
PARENTAL GATE

- Implementation used to be a best practice, now it is required

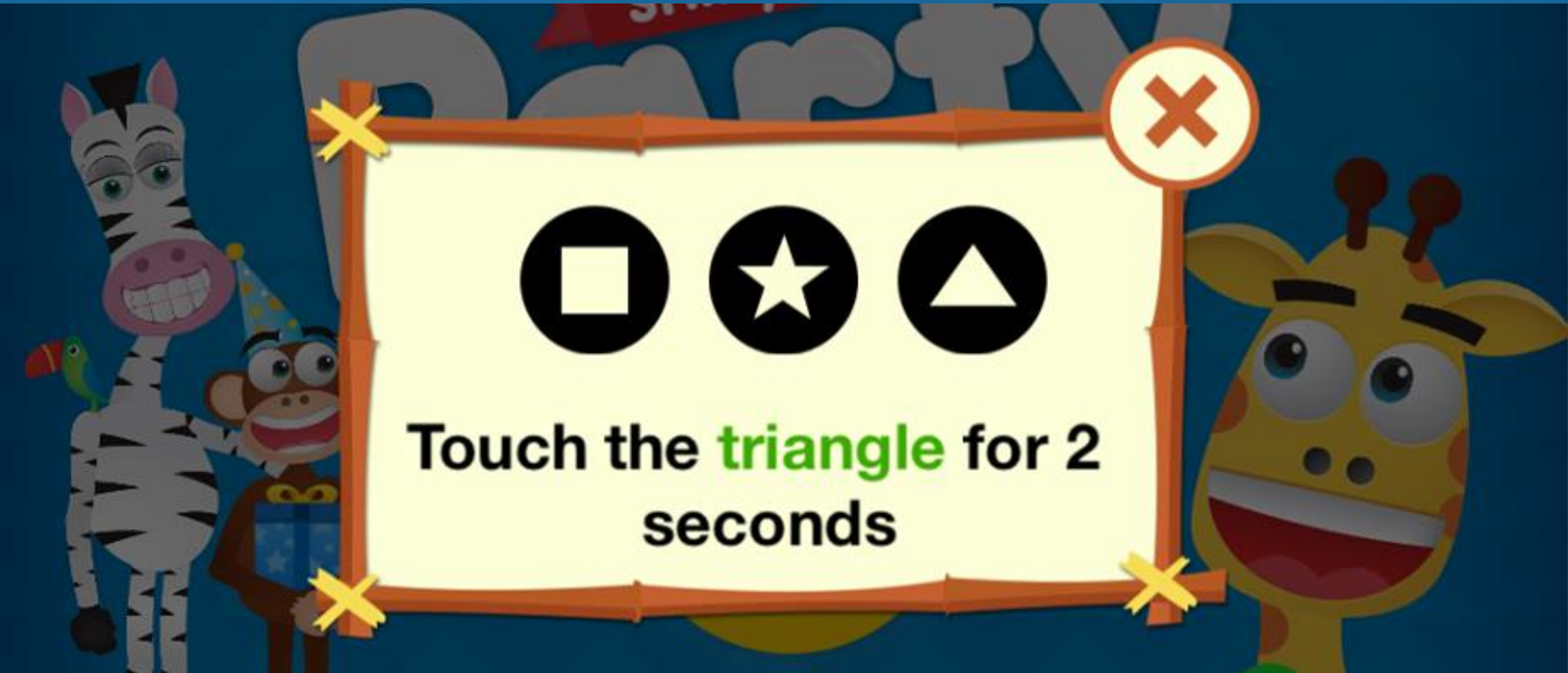
> 24. Kids Apps
>
> - 24.3- Apps primarily intended for use by kids under 13 must get parental permission or use a parental gate before allowing the user to link out of the app or engage in commerce
>
> This requires gating when linking out of the app or engaging in commerce such as selling in-app items.

-

PARENTAL GATE



PARENTAL GATE



PARENTAL GATE

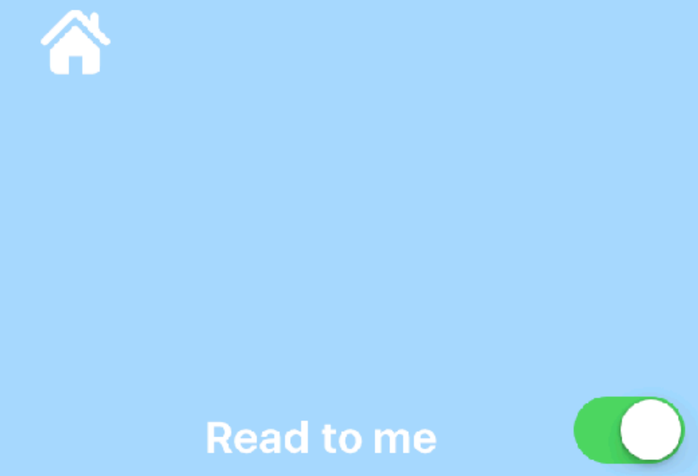
REQUIREMENTS

- 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

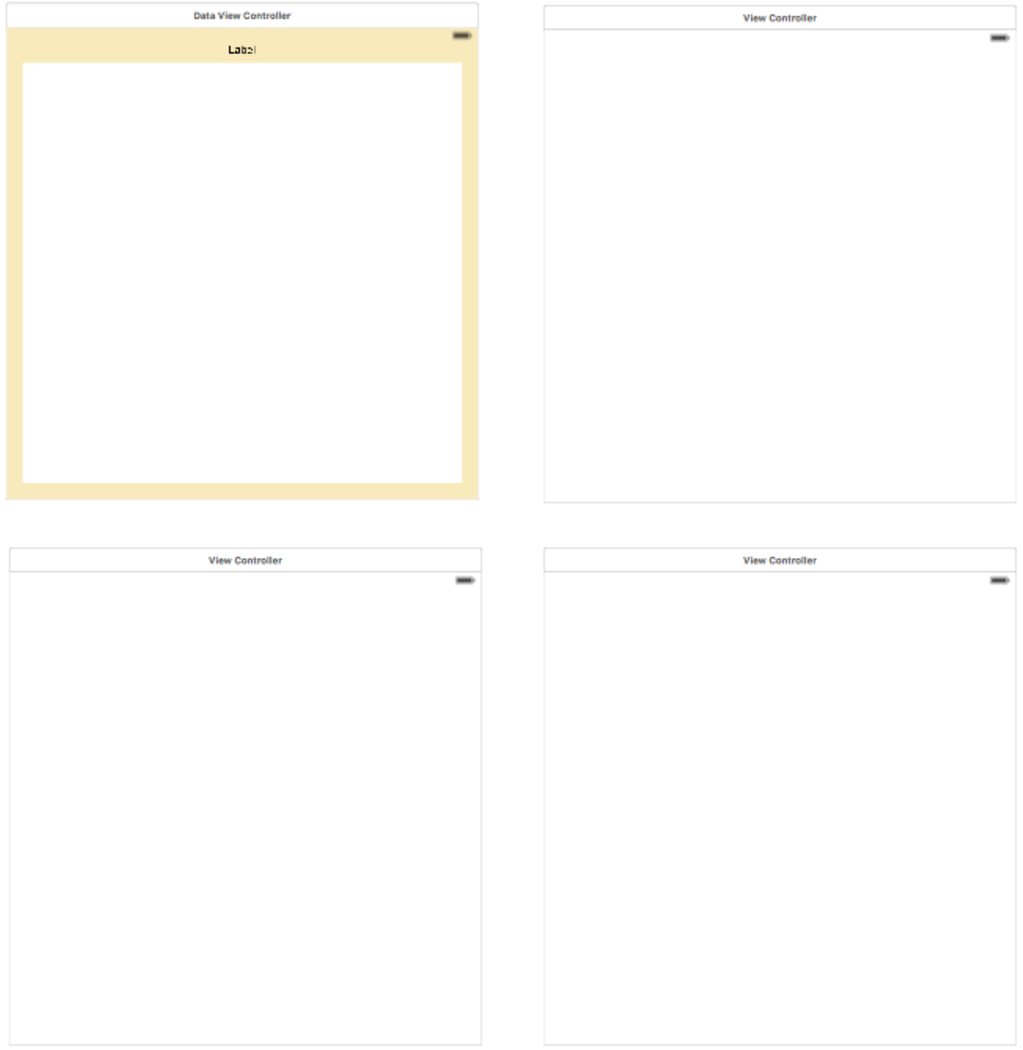
- Settings page in objective-c 😳
- Allow user to select preference to automatically read the page
    - Default should be "on"



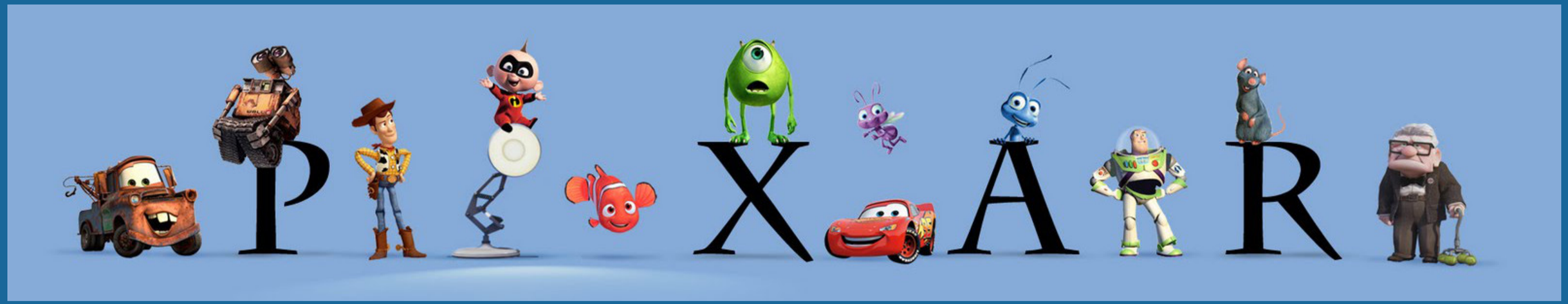
BOOK PAGE OPTIONS

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

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1C

SESSION 1D

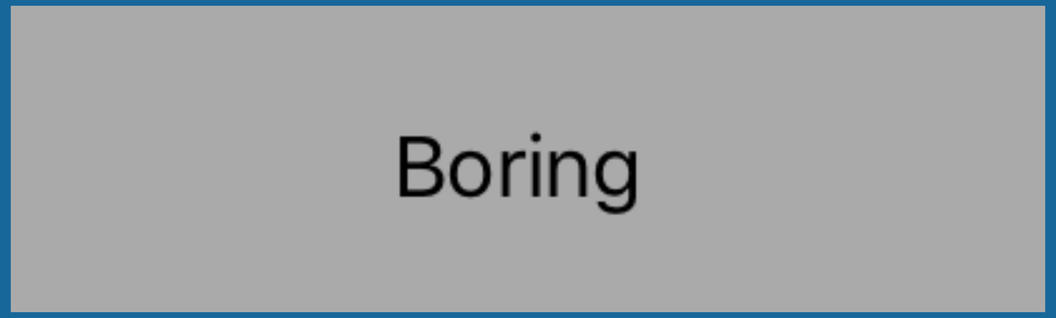
THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1D

ATTRIBUTED STRINGS

NSATTRIBUTEDSTRING



- UILabel text can is 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 sty
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

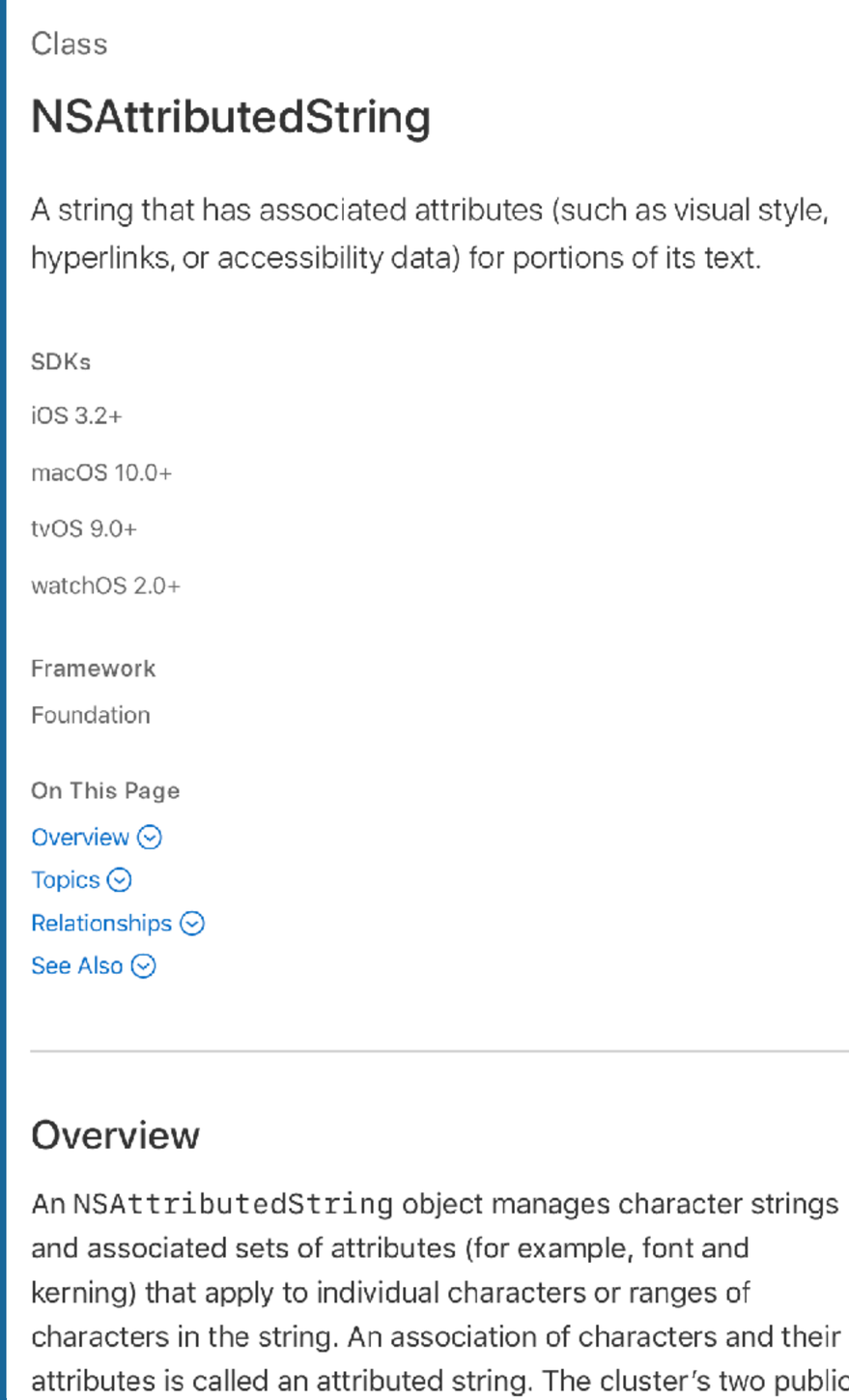
See Also

Overview

An `NSAttributedString` object manages character stri
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 t
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

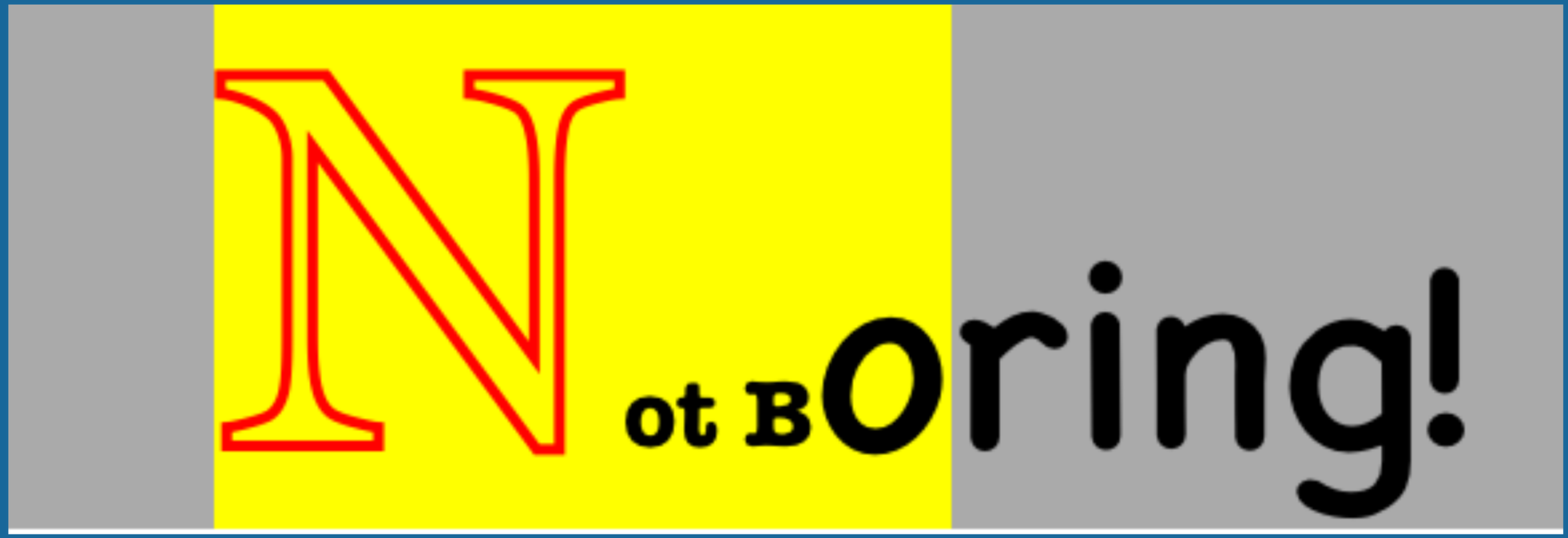
Relationships

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: [
                                                   NSAttributedStringKey.font:UIFont(na
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!"
```

NSATTRIBUTEDSTRING

Let the label use an attributed string instead

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

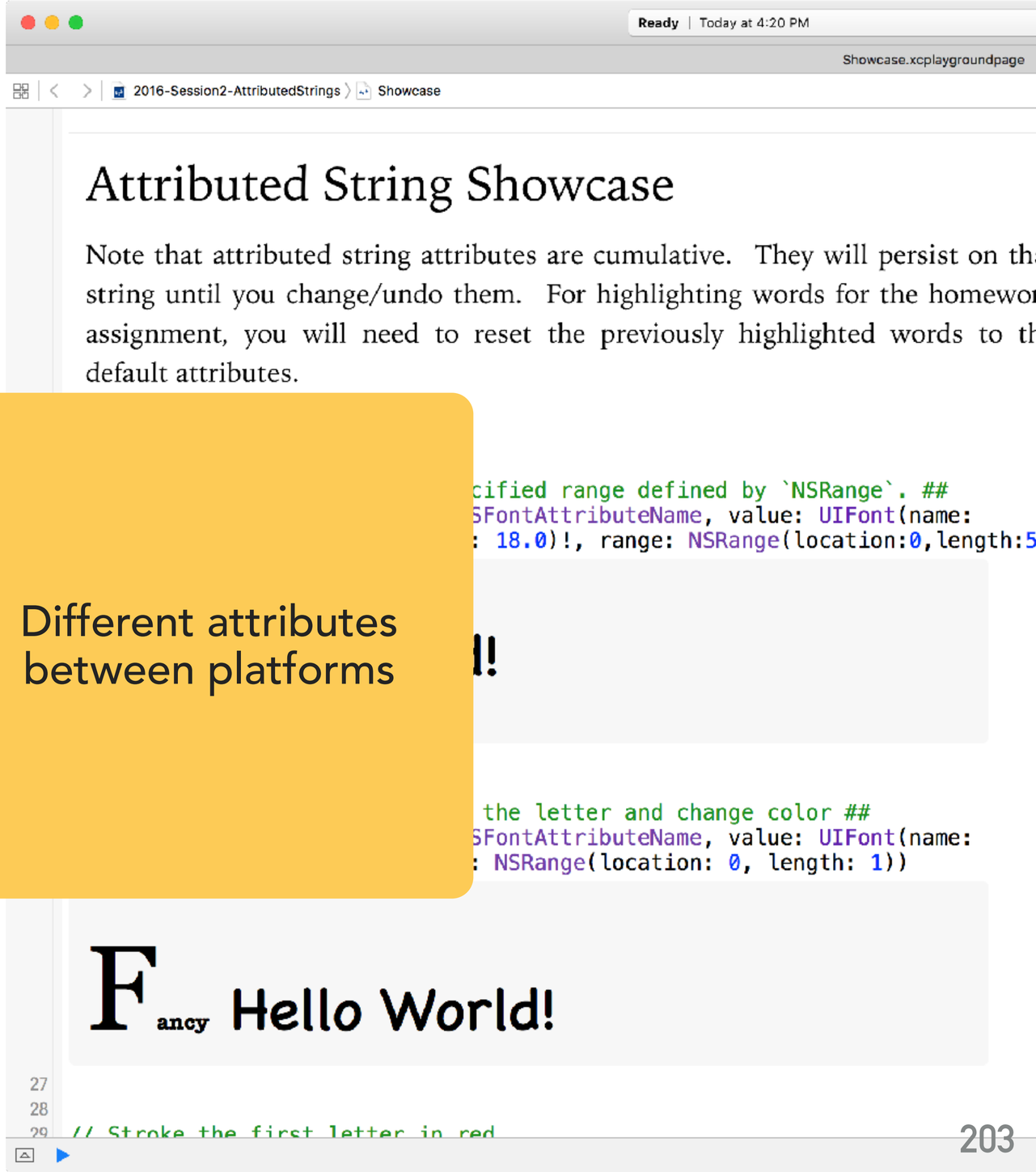
Fancy Hello World!

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.

Different attributes between platforms



NSATTRIBUTEDSTRING

Attributed String Programming Guide

PDF

Table of Contents

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

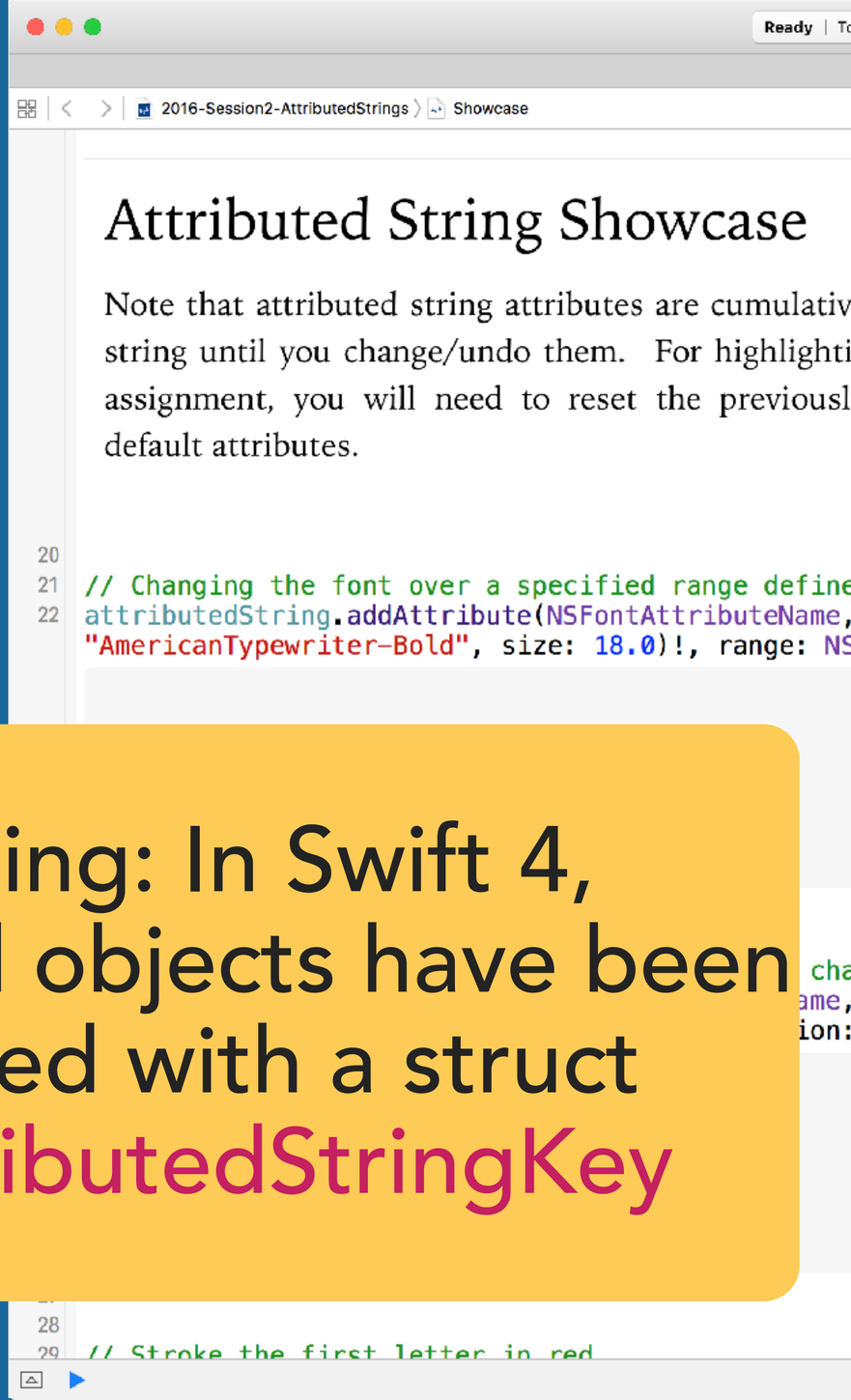
You should read this document if you need to work directly with attributed string objects.

Organization of This Document

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 NSStrokeColorAttributeName;
- NSString *const NSStrokeWidthAttributeName;
- NSString *const NSShadowAttributeName;



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

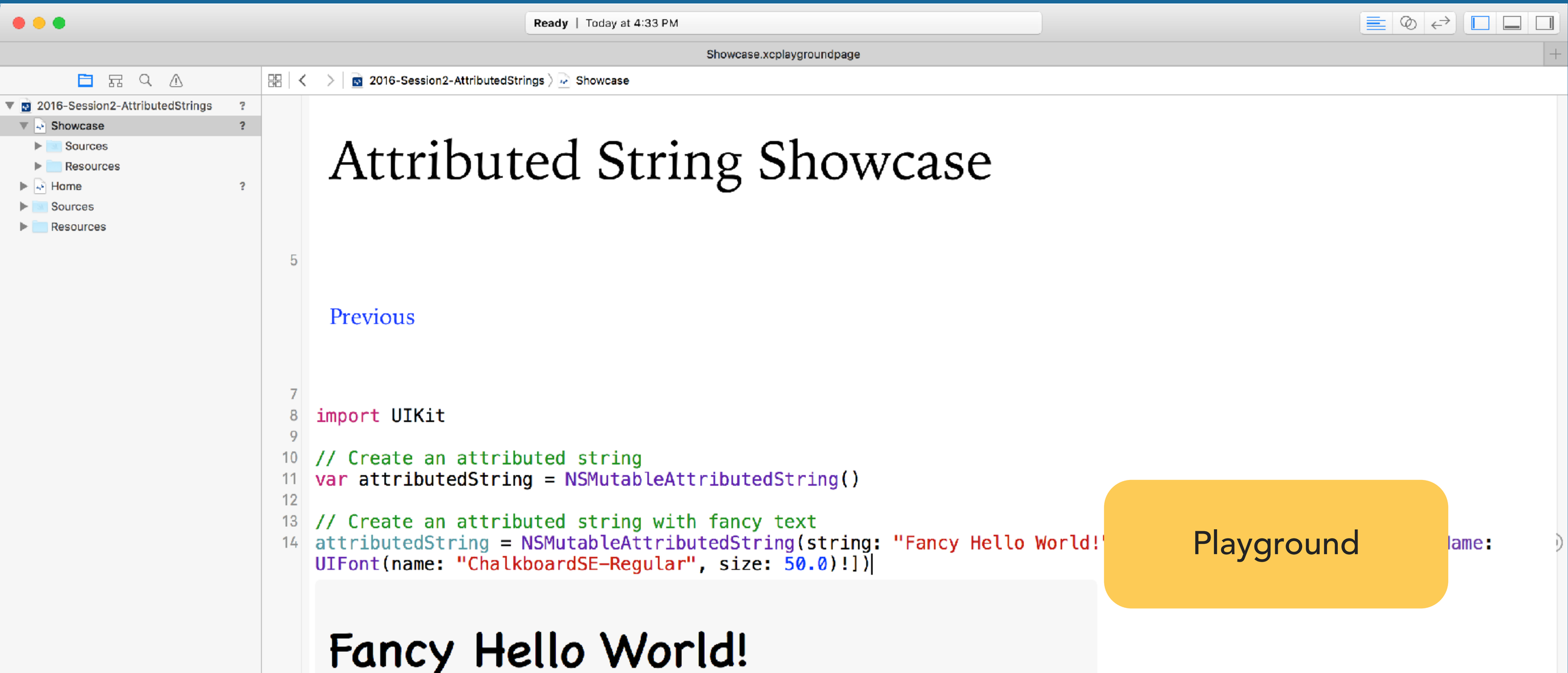
Set label property (not .text)

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
```

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

ATTRIBUTED STRINGS IN SWIFT



Playground

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:NSBackgroundColorAttributeName
                         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

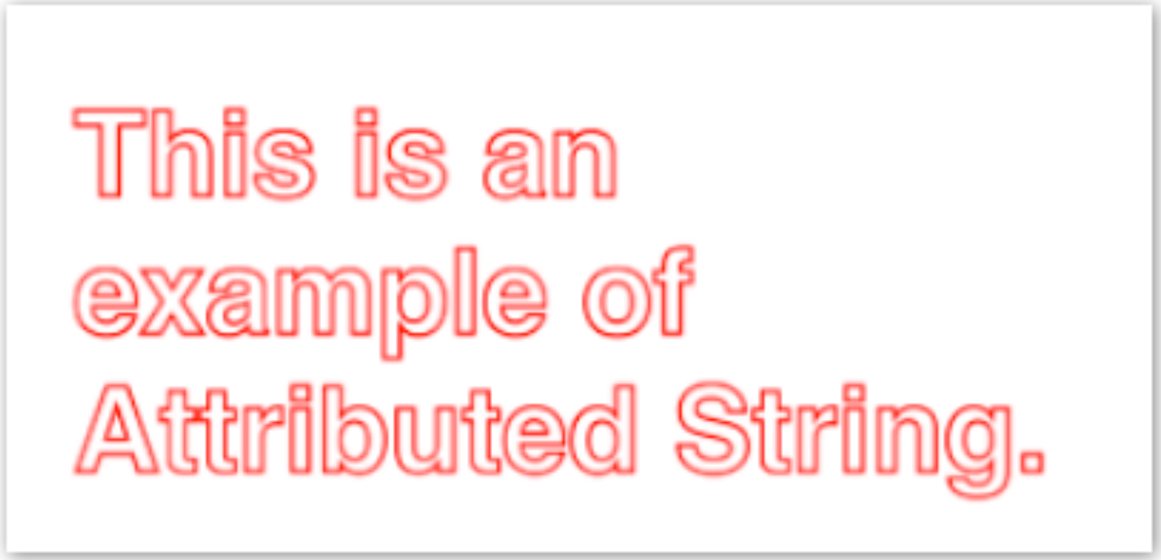
NSATTRIBUTEDSTRINGS

```
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:NSStrokeColorAttributeName value:_red range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSStrokeWidthAttributeName value:[NSNumber numberWithFloat:3.0] range:NSMakeRange(0,
_stringLength)];
```

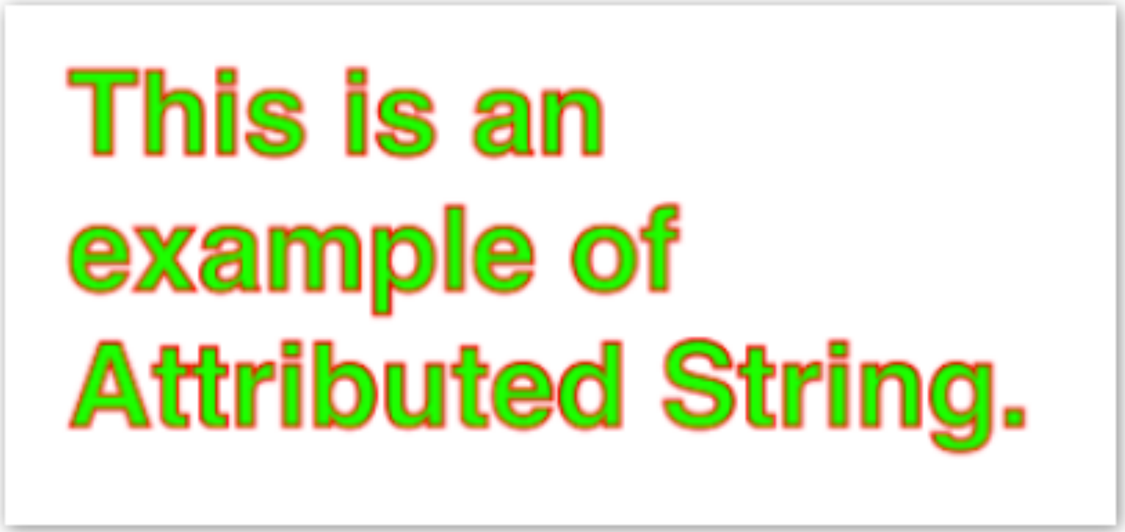


NSATTRIBUTEDSTRINGS

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

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:NSForegroundColorAttributeName value:_green range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSStrokeColorAttributeName value:_red range:NSMakeRange(0, _stringLength)];
[attString addAttribute:NSStrokeWidthAttributeName value:[NSNumber numberWithFloat:-3.0] range:NSMakeRange(0,
_stringLength)];
```



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:NSForegroundColorAttributeName 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:NSForegroundColorAttributeName value:_red range:NSMakeRange(0, _stringLength)];

[attString addAttribute:NSKernAttributeName value:[NSNumber numberWithInt:5] range:NSMak
```

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:NSUnderlineStyleAttributeName value:[NSNumber numberWithInt:2] range:NSMakeRange(0,
_stringLength)];
```

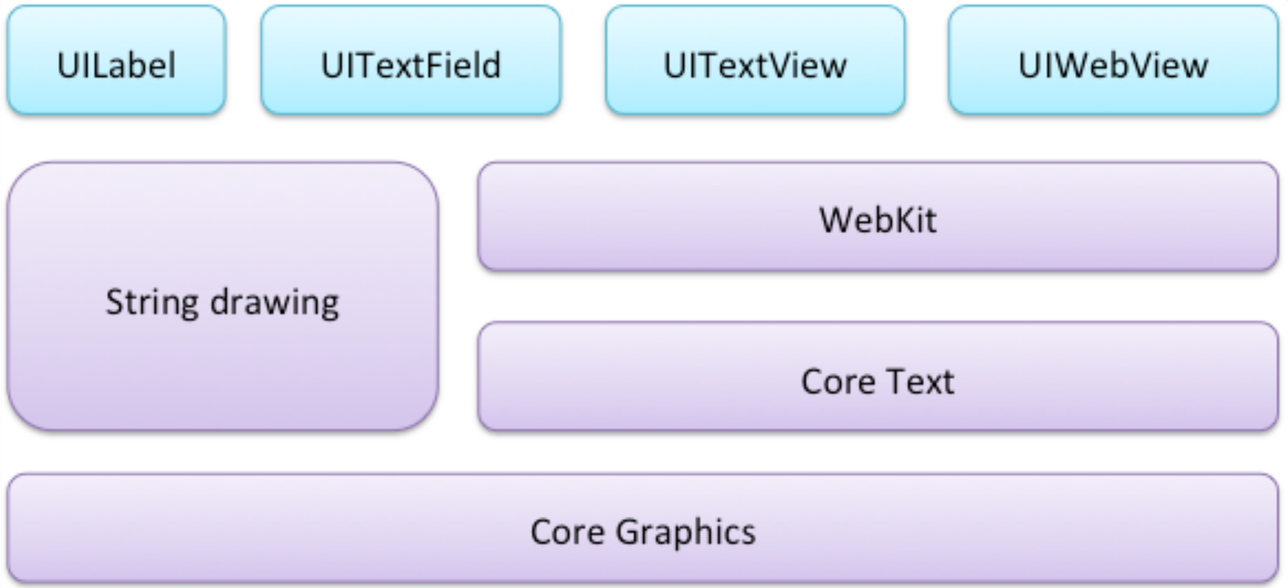

```
UIColor *_blue=[UIColor blueColor];
UIColor *_blueL=[UIColor colorWithRed:0 green:0 blue:0.5 alpha:0.7];
UIFont *font=[UIFont fontWithName:@"Helvetica-Bold" size:30.0f];

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

A BIT OF TEXT KIT

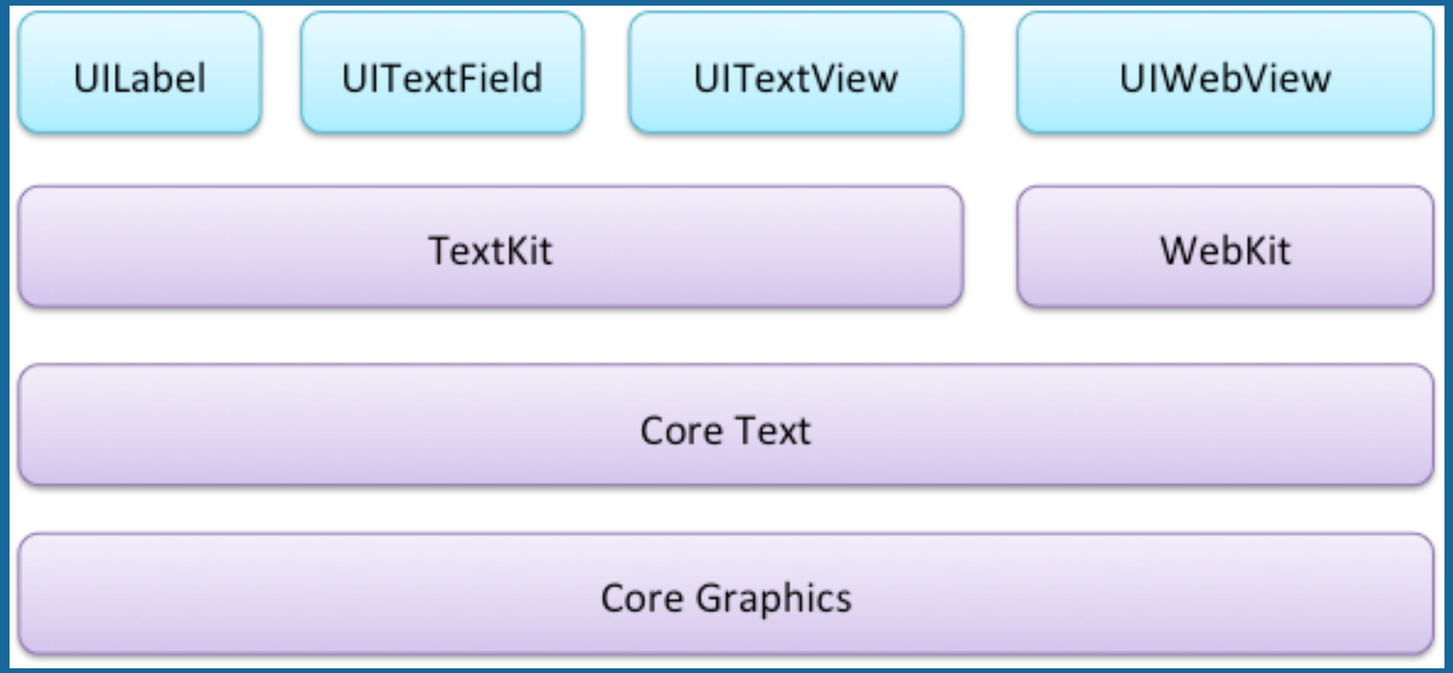
TEXT KIT

SUBTITLE

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



TEXT KIT

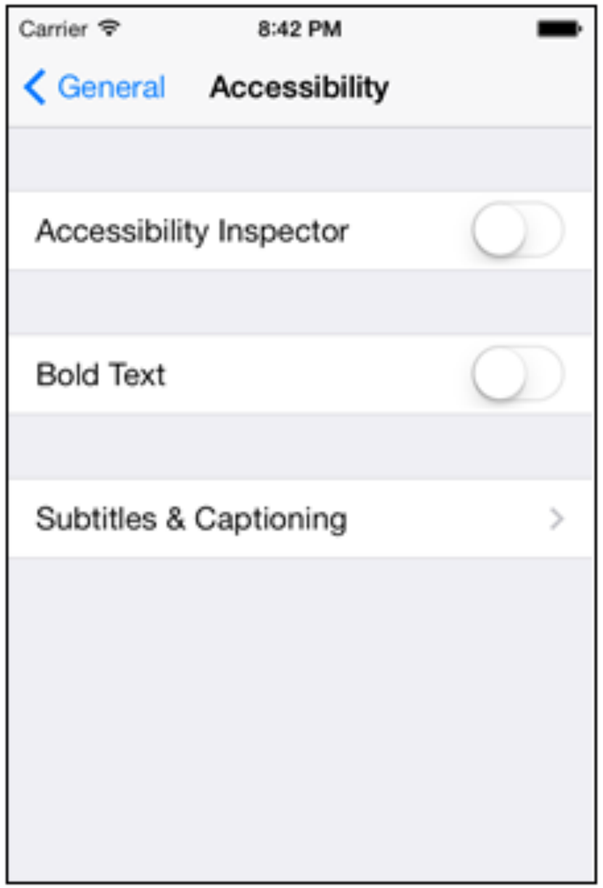


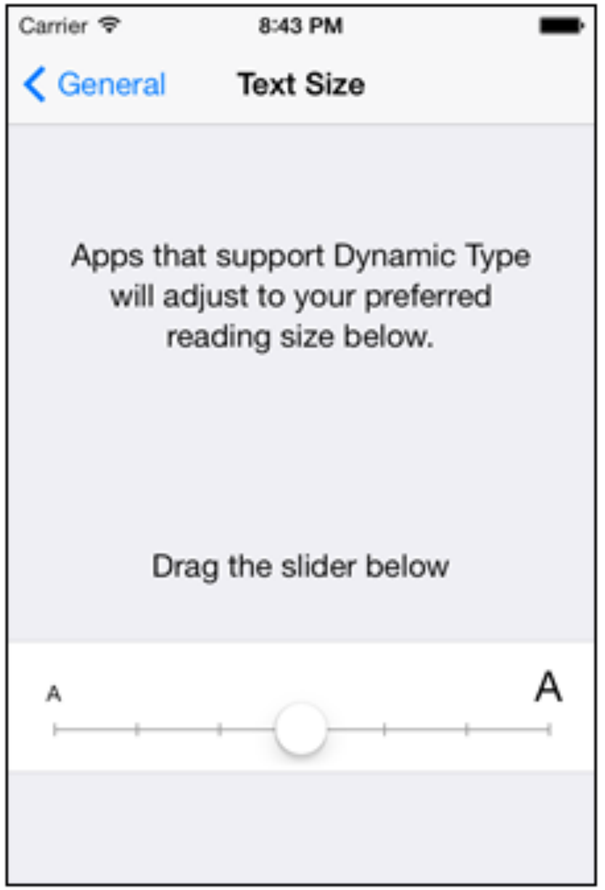
- Built on top of Core Text framework
  - Notoriously difficult API to work with

TEXT KIT

SUBTITLE

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



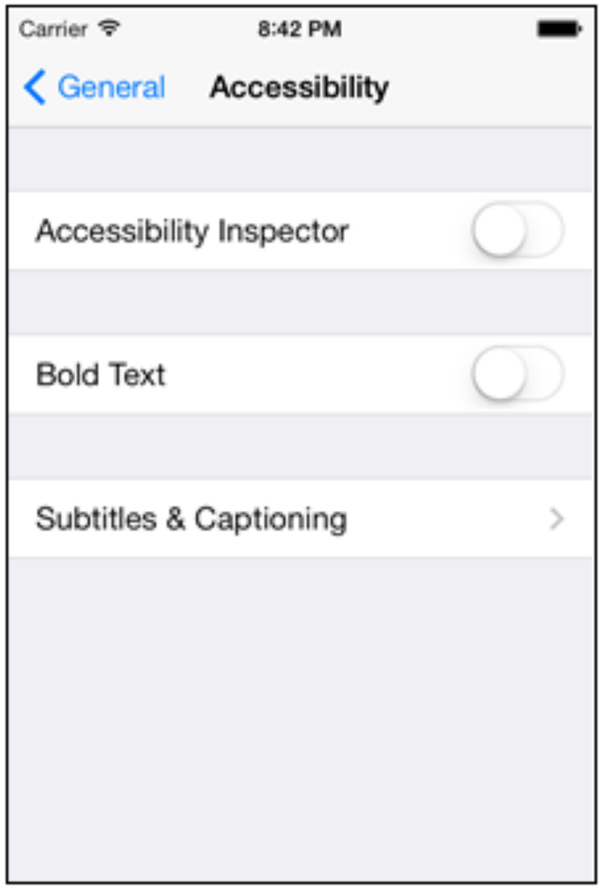


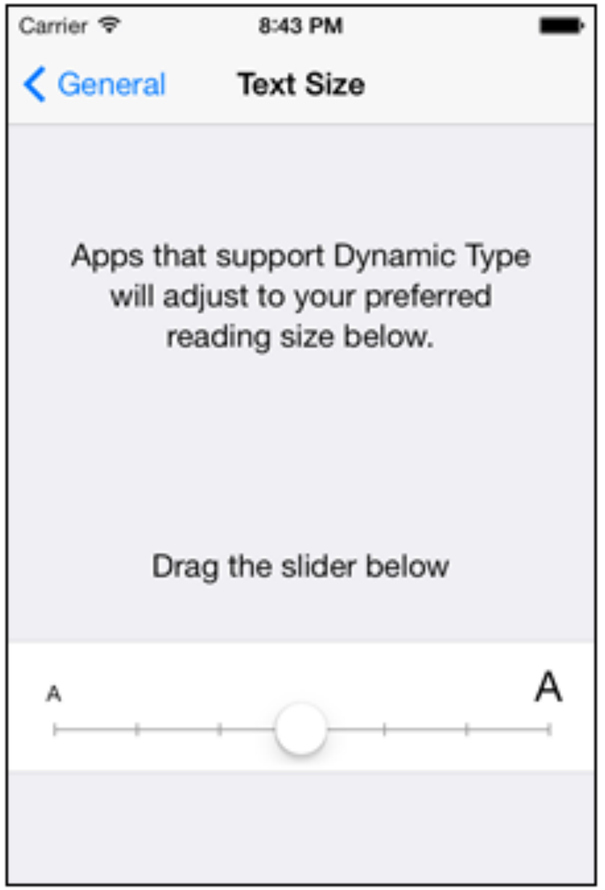
TEXT KIT

SUBTITLE

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

Users can ruin your interface

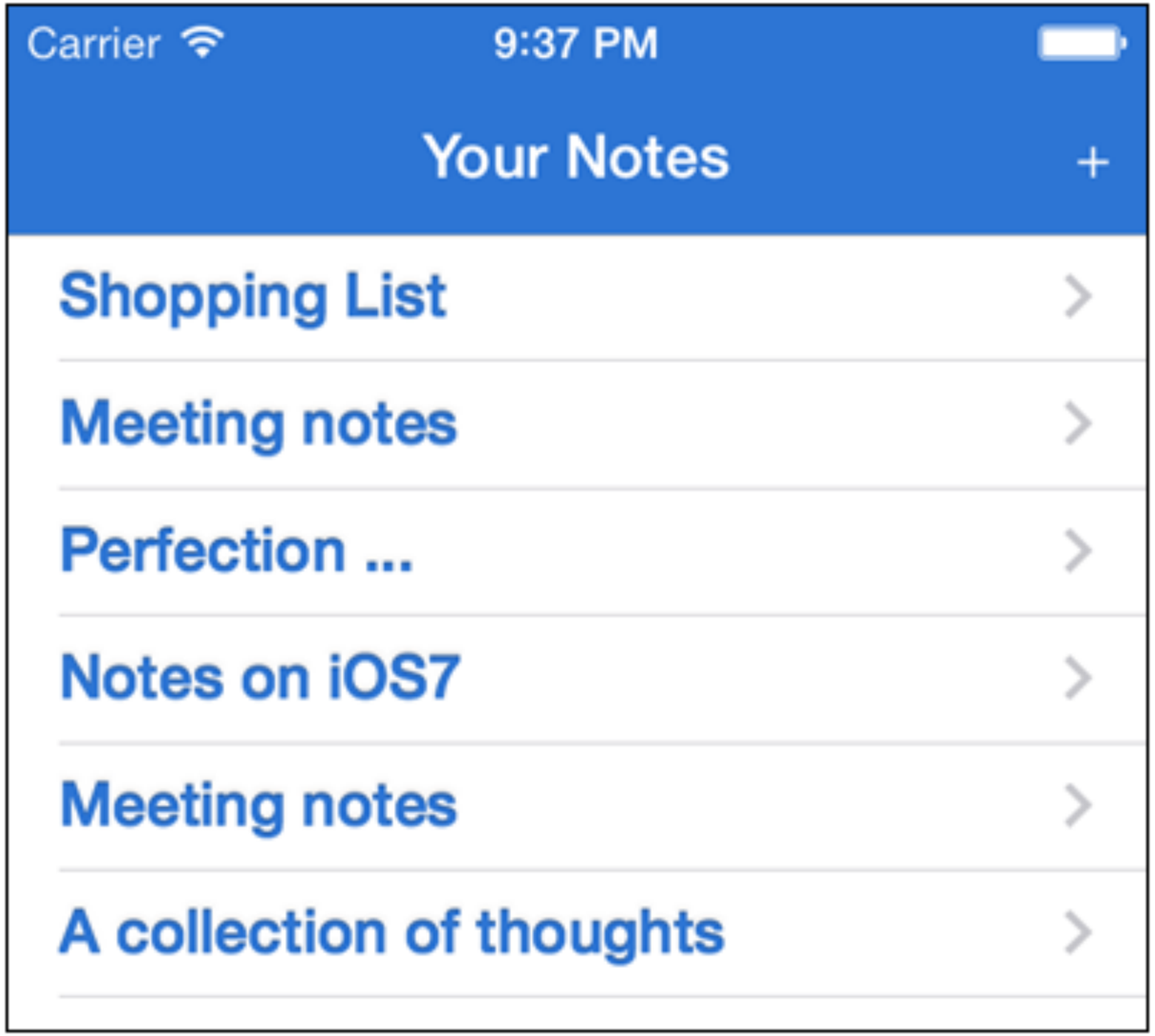




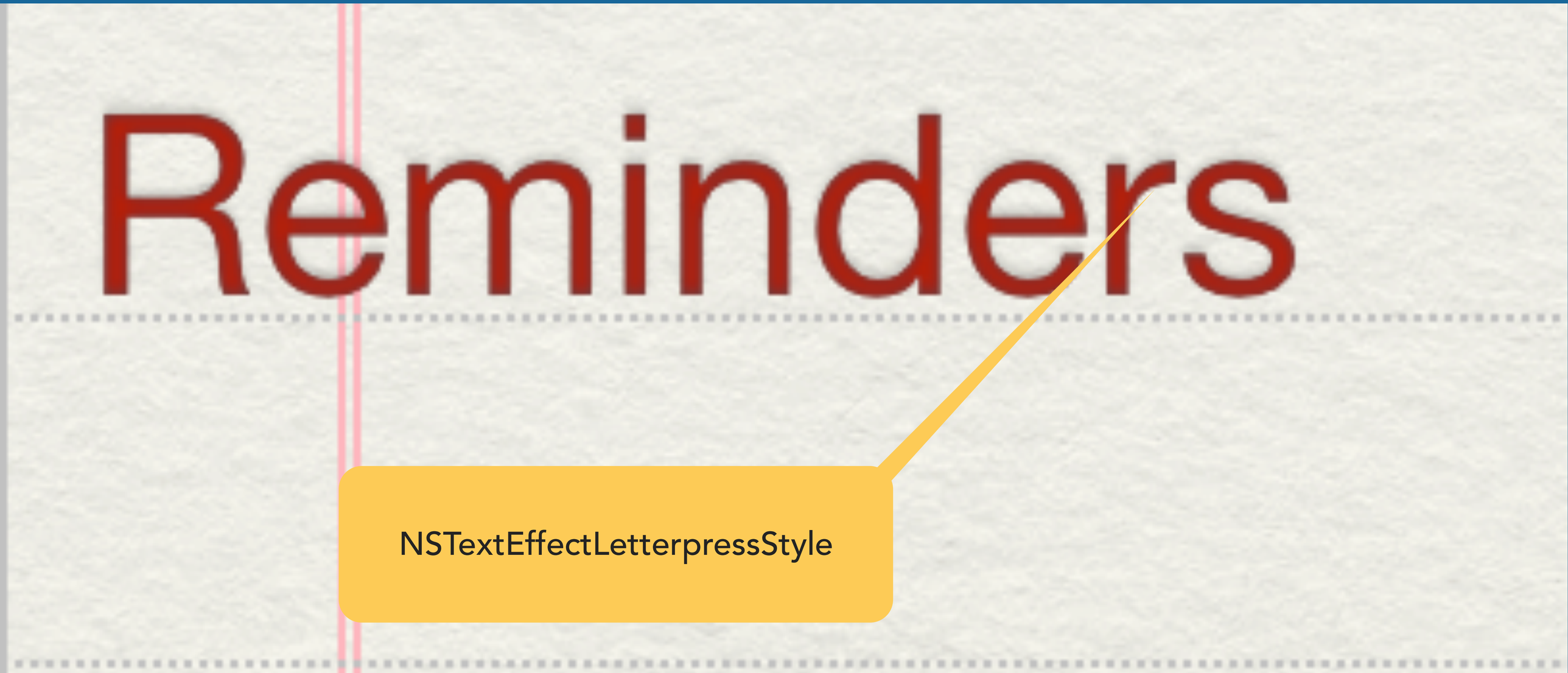
TEXT KIT

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



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

TEXT KIT

- 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

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1D

SESSION 1E

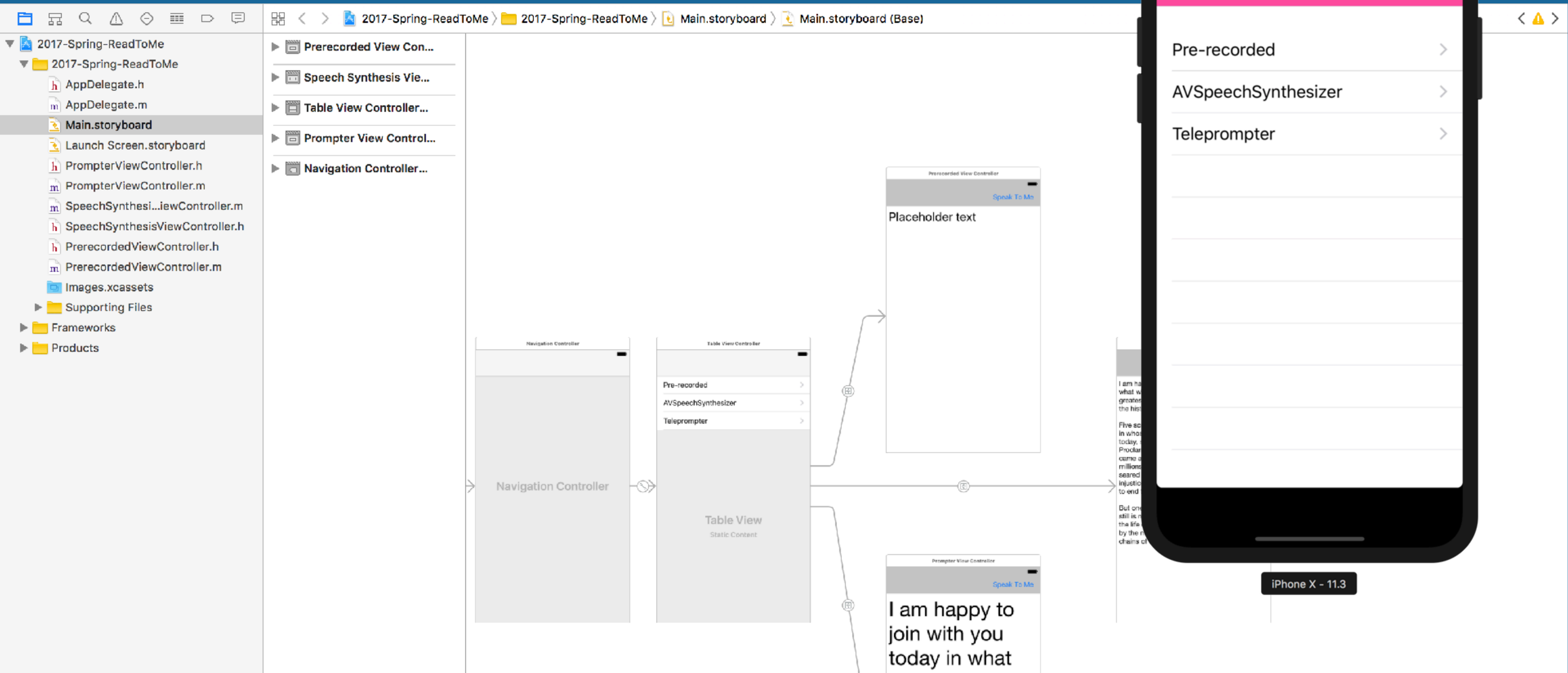
THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

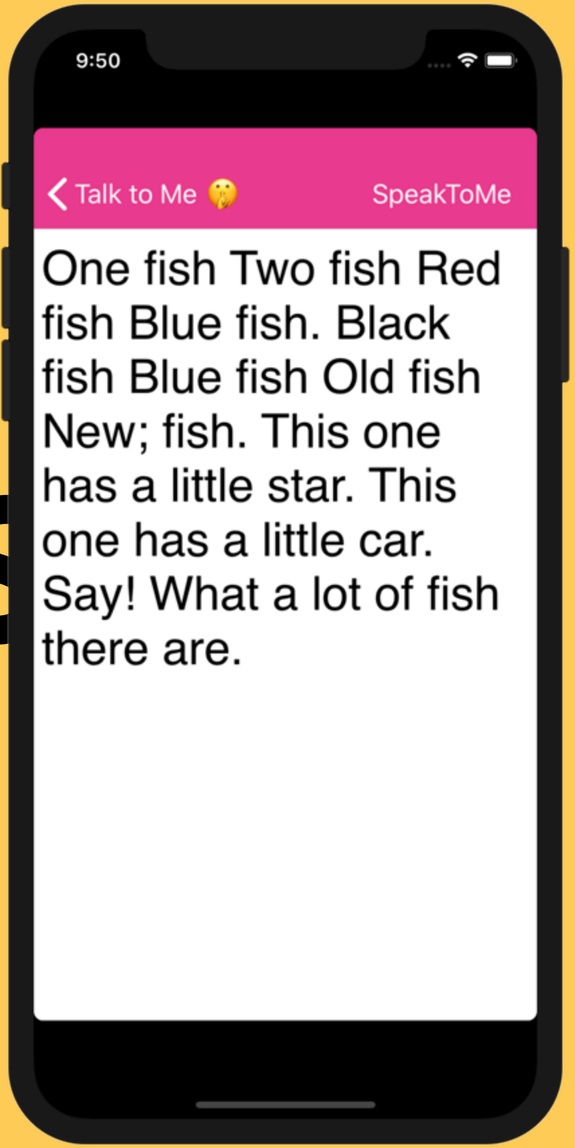
MPCS 51032 • SPRING 2020 • SESSION 1E

READ-TO-ME DEMO

READ-TO-ME

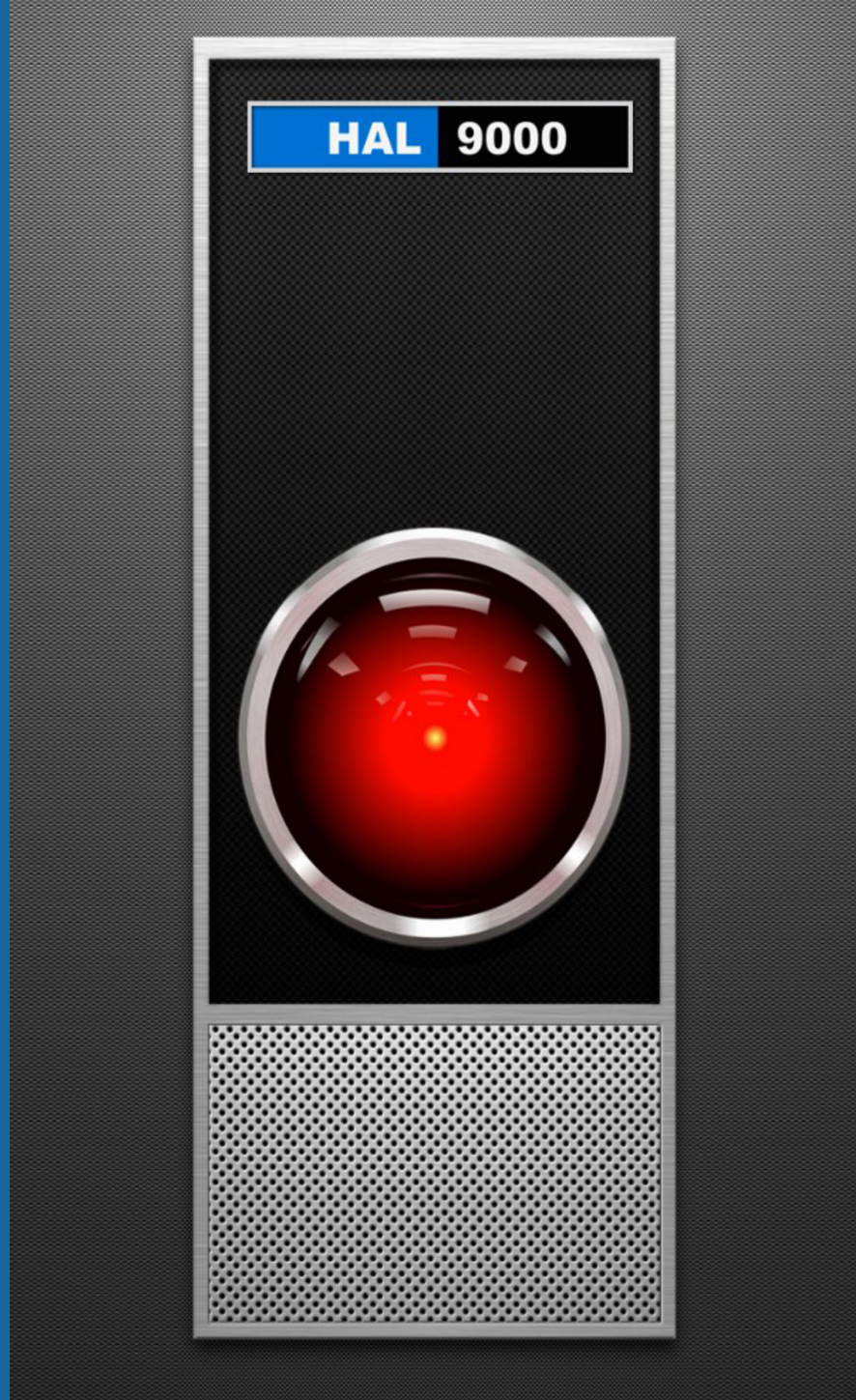

2017-Spring-ReadToMe
2017-Spring-ReadToMe
AppDelegate.h
AppDelegate.m
Main.storyboard
Launch Screen.storyboard
PrompterViewController.h
PrompterViewController.m
SpeechSynthesi...iewController.m
SpeechSynthesisViewController.h
PrerecordedViewController.h
PrerecordedViewController.m
Images.xcassets
Supporting Files
Frameworks
Products
2017-Spring-ReadToMe
2017-Spring-ReadToMe
Main.storyboard
Main.storyboard (Base)
Prerecorded View Con...
Speech Synthesis Vie...
Table View Controller...
Prompter View Control...
Navigation Controller...
Navigation Controller
Table View Controller
Pre-recorded
AVSpeechSynthesizer
Teleprompter
Table View
Static Content
Prerecorded View Controller
Speak To Me
Placeholder text
Prompter View Controller
Speak To Me
I am happy to join with you today in what
9:51
Talk to Me
Pre-recorded
AVSpeechSynthesizer
Teleprompter
iPhone X - 11.3

SPEECH SYNTHES



SPEECH SYNTHESIS

- AVSpeechSynthesizer
  - Introduced in iOS7
- Offline speech synthesis capability



SPEECH SYNTHESIS

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

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

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

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.

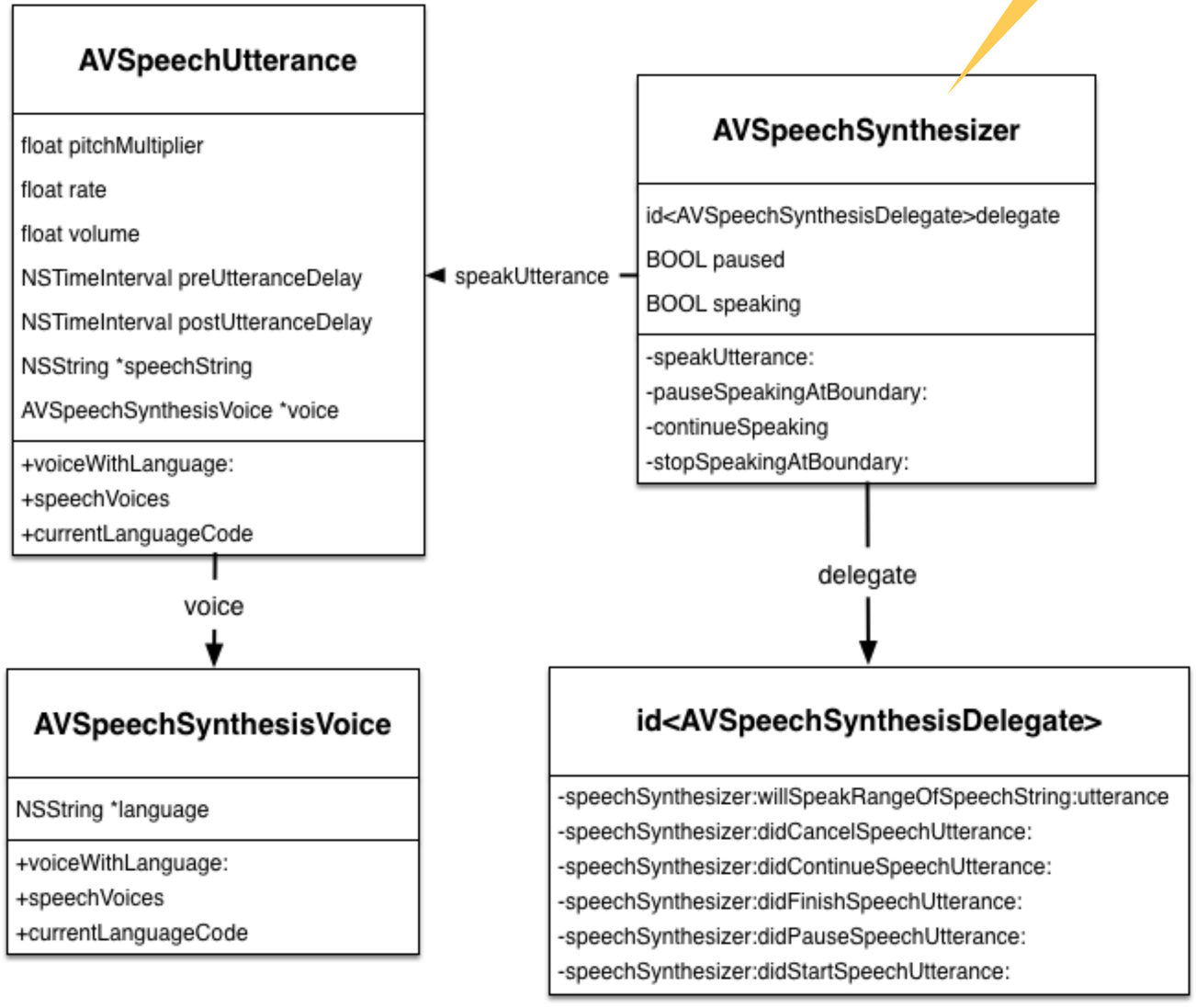
The speech synthesizer maintains a queue of utterances to be spoken. If the synthesizer is not currently speaking, calling `speak(_:)` begins speaking that utterance immediately (or begin waiting through its `preUtteranceDelay` if one is set). If the synthesizer is speaking, utterances are added to a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

SUBTITLE

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

Wait until word is spoken before pausing

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

SPEECH SYNTHESIS

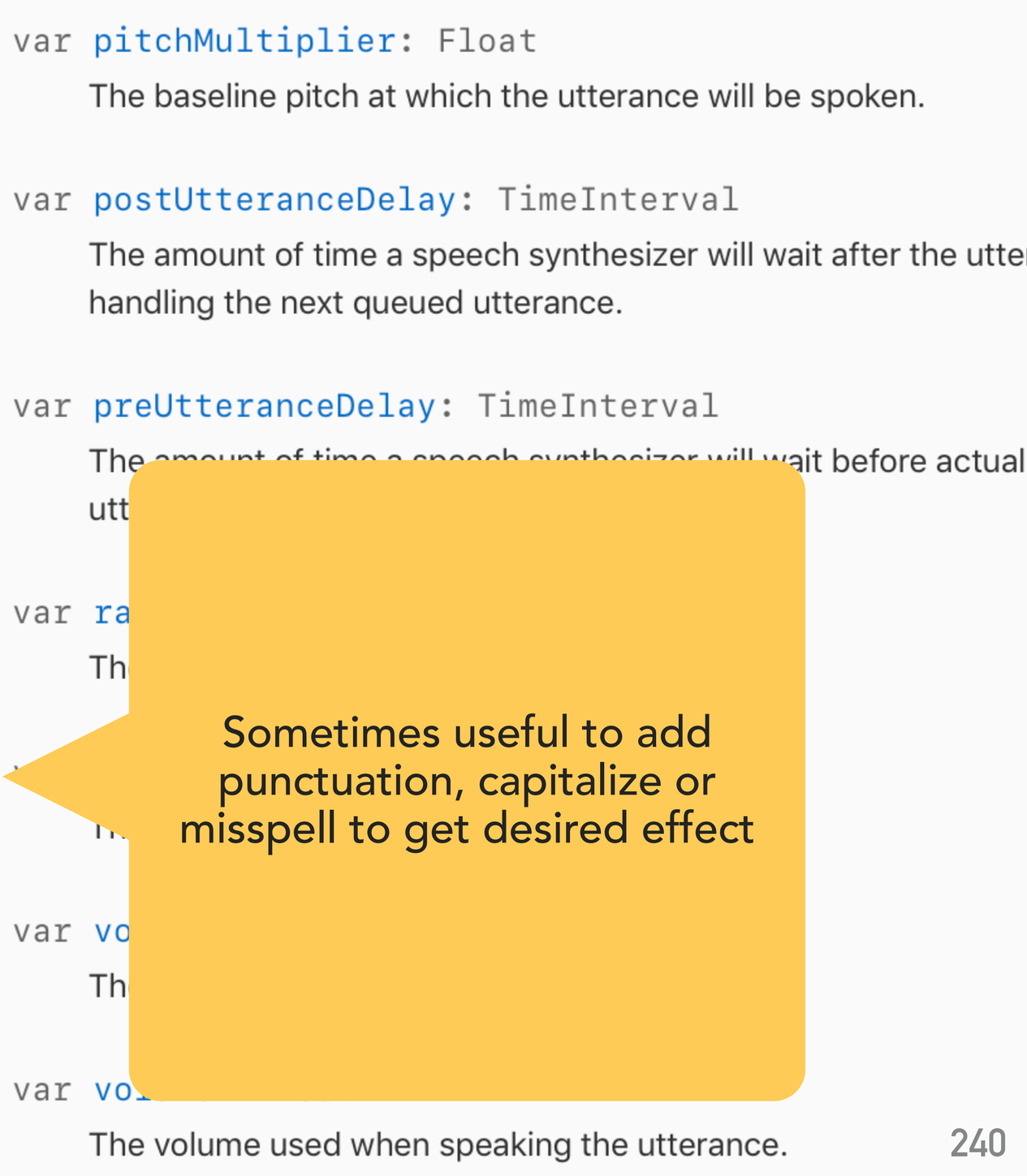
```
let string = "One fish Two fish Red fish Blue fish. Black fish Blue fish Old fish New; fish.
    This one has a little star. This one has a little car. Say! What a lot of fish there are;"

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



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ったたくさんのひとた
    ち、ファゼーロとロザーロ、羊飼のミーロや、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 ra

```
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'
```

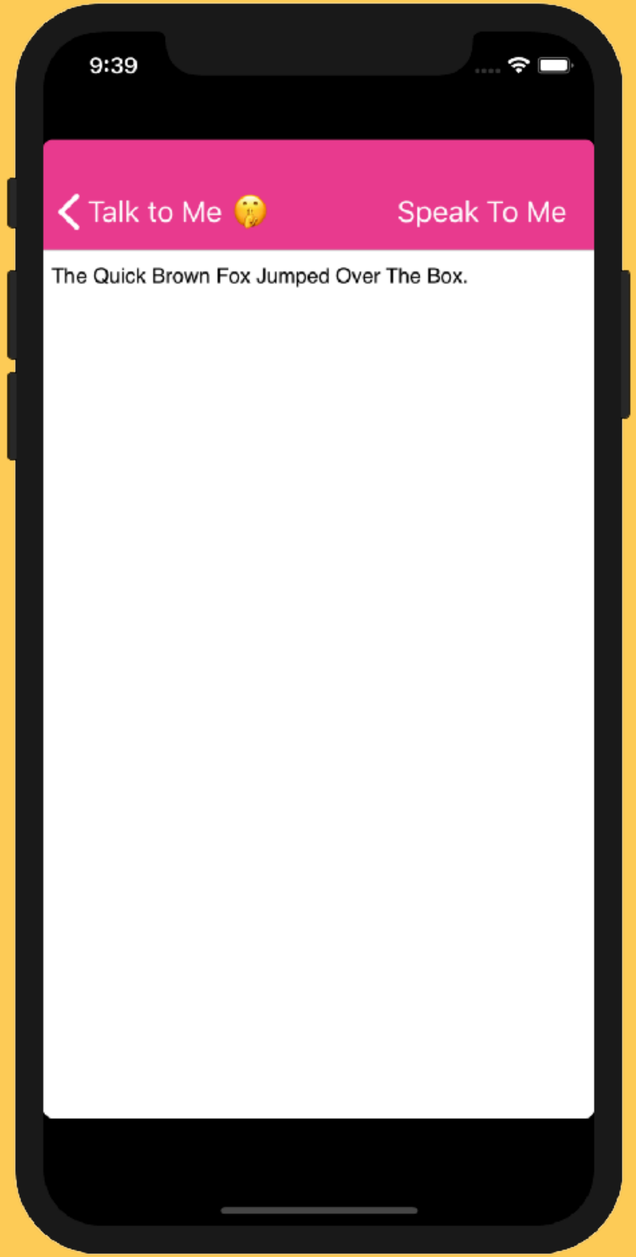
?

SPEECH SYNTHESIS

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

}
```

- Detect text language from text

PRE-RECORDED AUDIO



AUDACITY

Audacity®

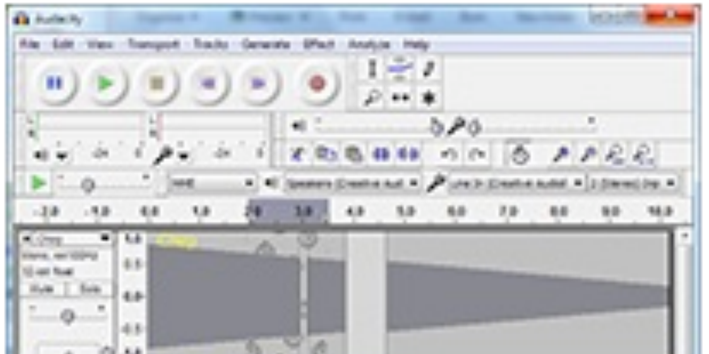
Google™ Custom Search Search

Search Audacity sites and other selected sites.

Home | About | Download | Help | Contact Us | Get Involved | Donate

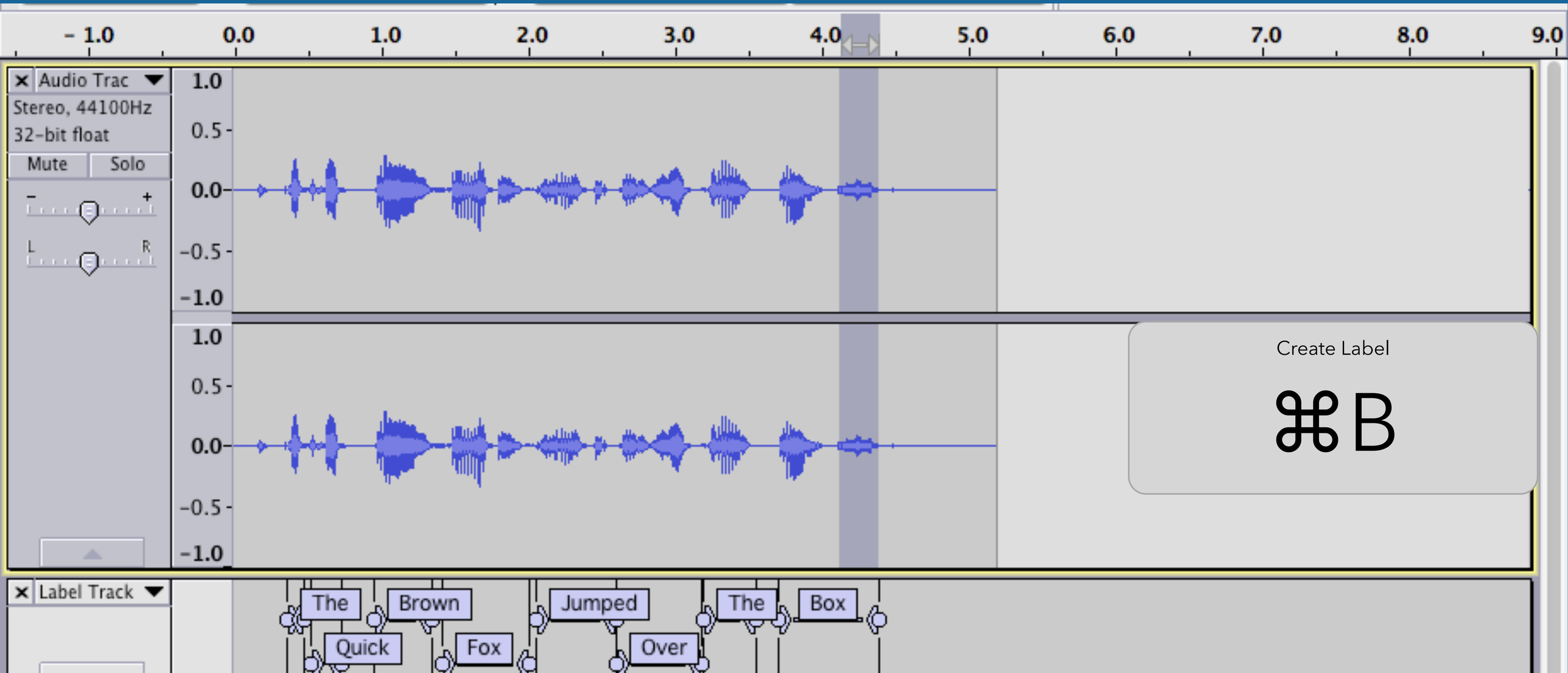
Audacity® is free, open source, cross-platform software for recording and editing sounds.

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

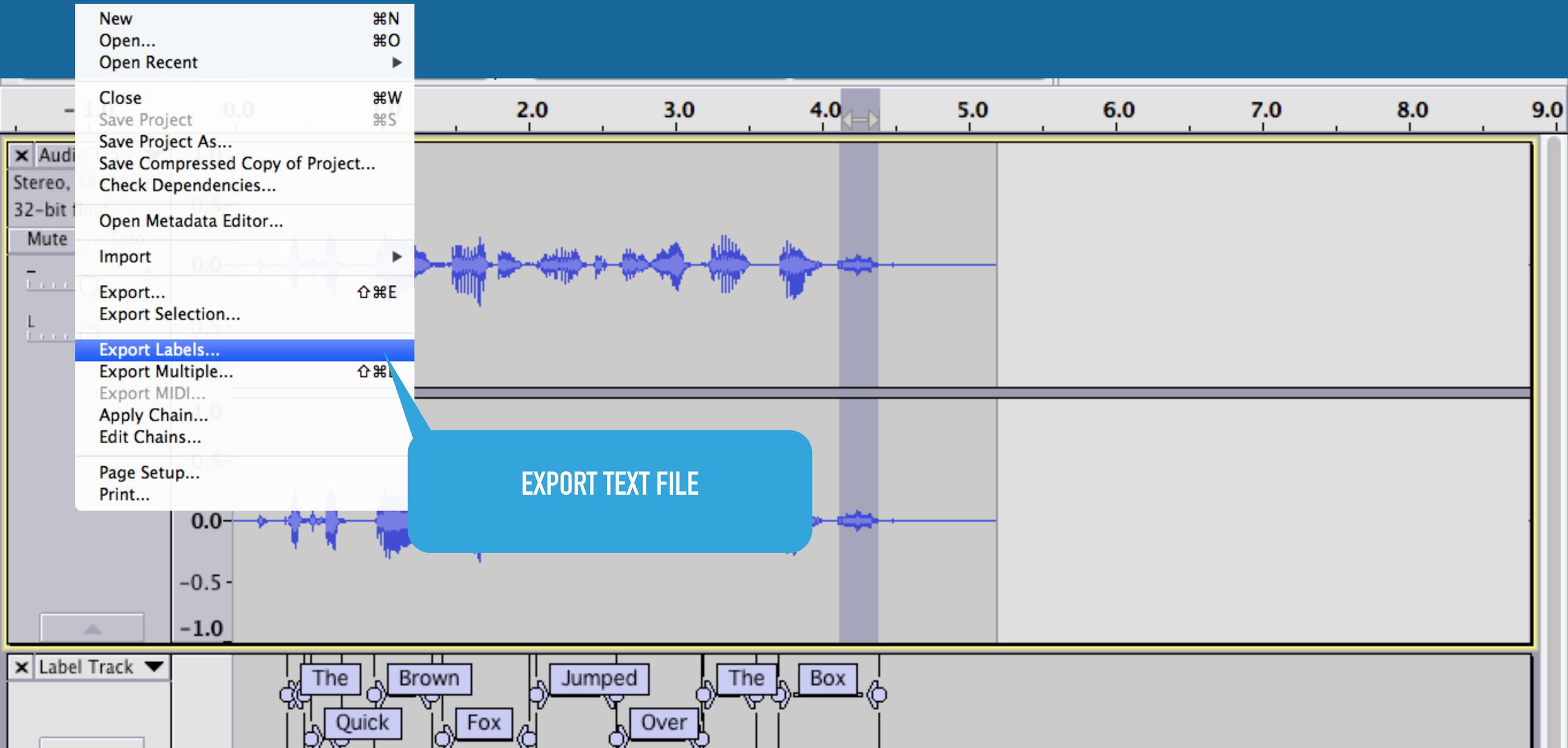


Download Audacity 2.0.3

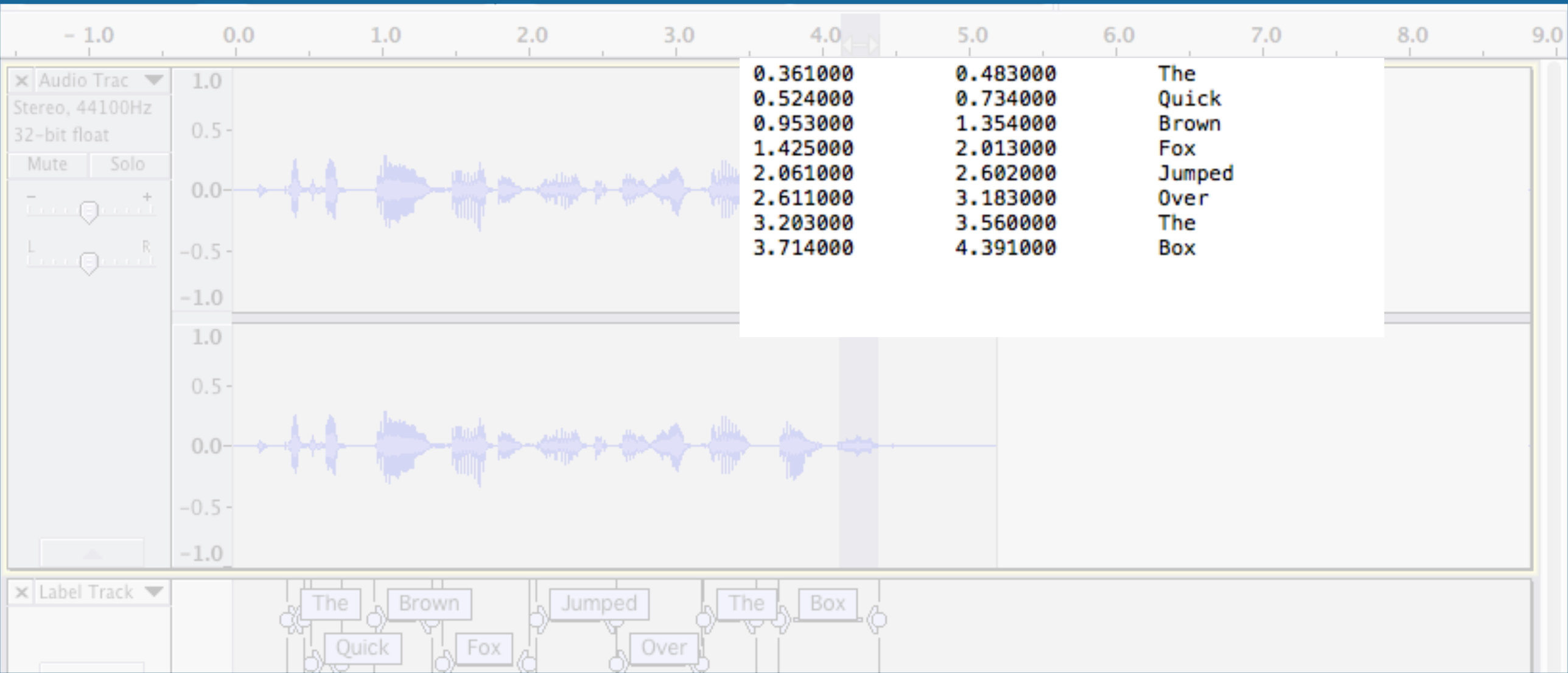
AUDACITY


- 1.0
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
Audio Trac
Stereo, 44100Hz
32-bit float
Mute
Solo
L
R
1.0
0.5
0.0
-0.5
-1.0
Create Label
⌘B
Label Track
The
Brown
Jumped
The
Box
Quick
Fox
Over

AUDACITY


New ⌘N
Open... ⌘O
Open Recent
Close ⌘W
Save Project ⌘S
Save Project As...
Save Compressed Copy of Project...
Check Dependencies...
Open Metadata Editor...
Import
Export... ⇧⌘E
Export Selection...
Export Labels...
Export Multiple... ⇧⌘
Export MIDI...
Apply Chain...
Edit Chains...
Page Setup...
Print...
EXPORT TEXT FILE
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
Stereo,
32-bit
Mute
0.0
-0.5
-1.0
Label Track
The
Brown
Jumped
The
Box
Quick
Fox
Over

AUDACITY



0.361000	0.483000	The
0.524000	0.734000	Quick
0.953000	1.354000	Brown
1.425000	2.013000	Fox
2.061000	2.602000	Jumped
2.611000	3.183000	Over
3.203000	3.560000	The
3.714000	4.391000	Box

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

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1E

SESSION 1F

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

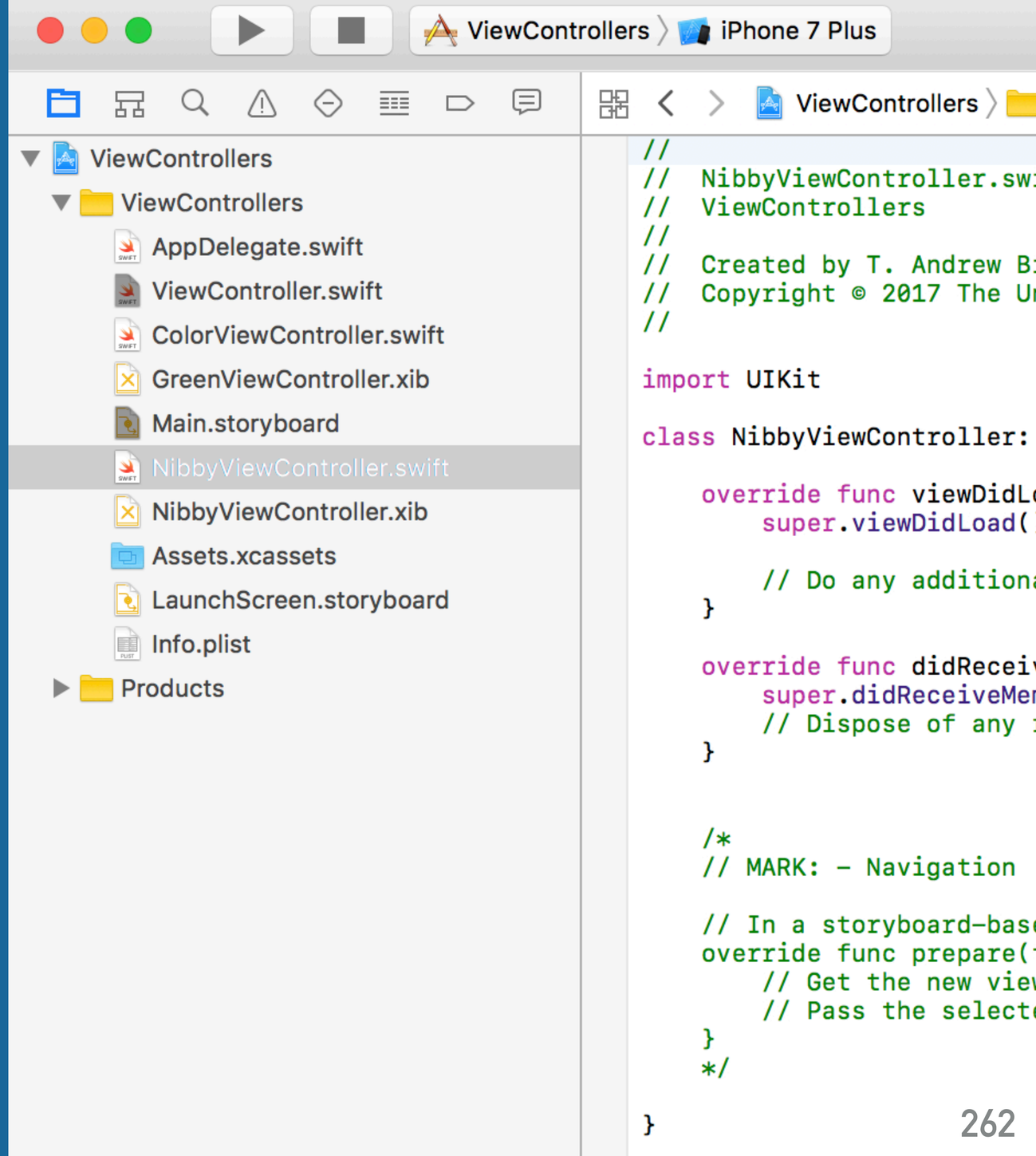
MPCS 51032 • SPRING 2020 • SESSION 1F

NIB BASED VIEWCONTROLLERS

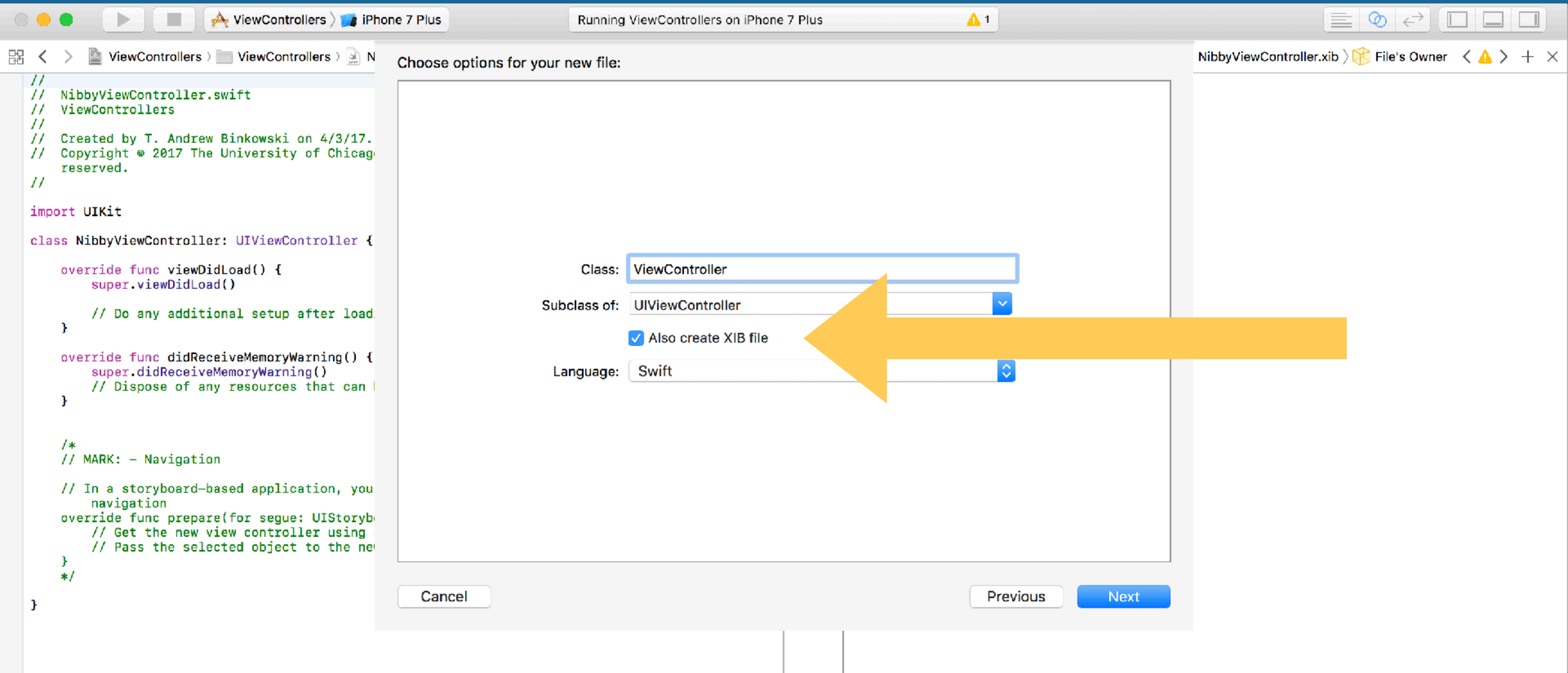
NIB BASED VIEWCONTROLLERS

SUBTITLE

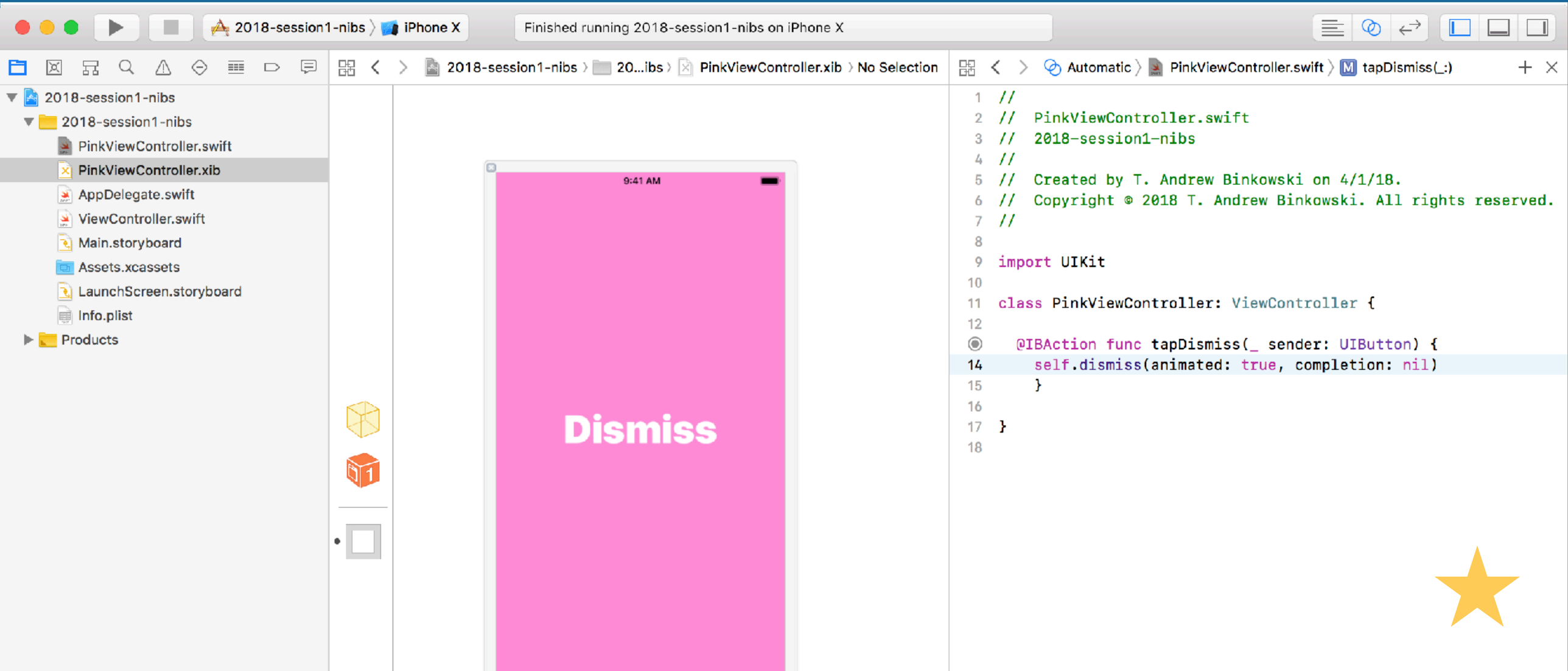
- A storyboard is just a collection of NIB files
- You can still use individual NIB files in a project
  - Easier to work with VCS
  - Separation of tasks
  - ...



NIB BASED VIEWCONTROLLERS



NIB BASED VIEWCONTROLLERS



NIB BASED VIEWCONTROLLERS

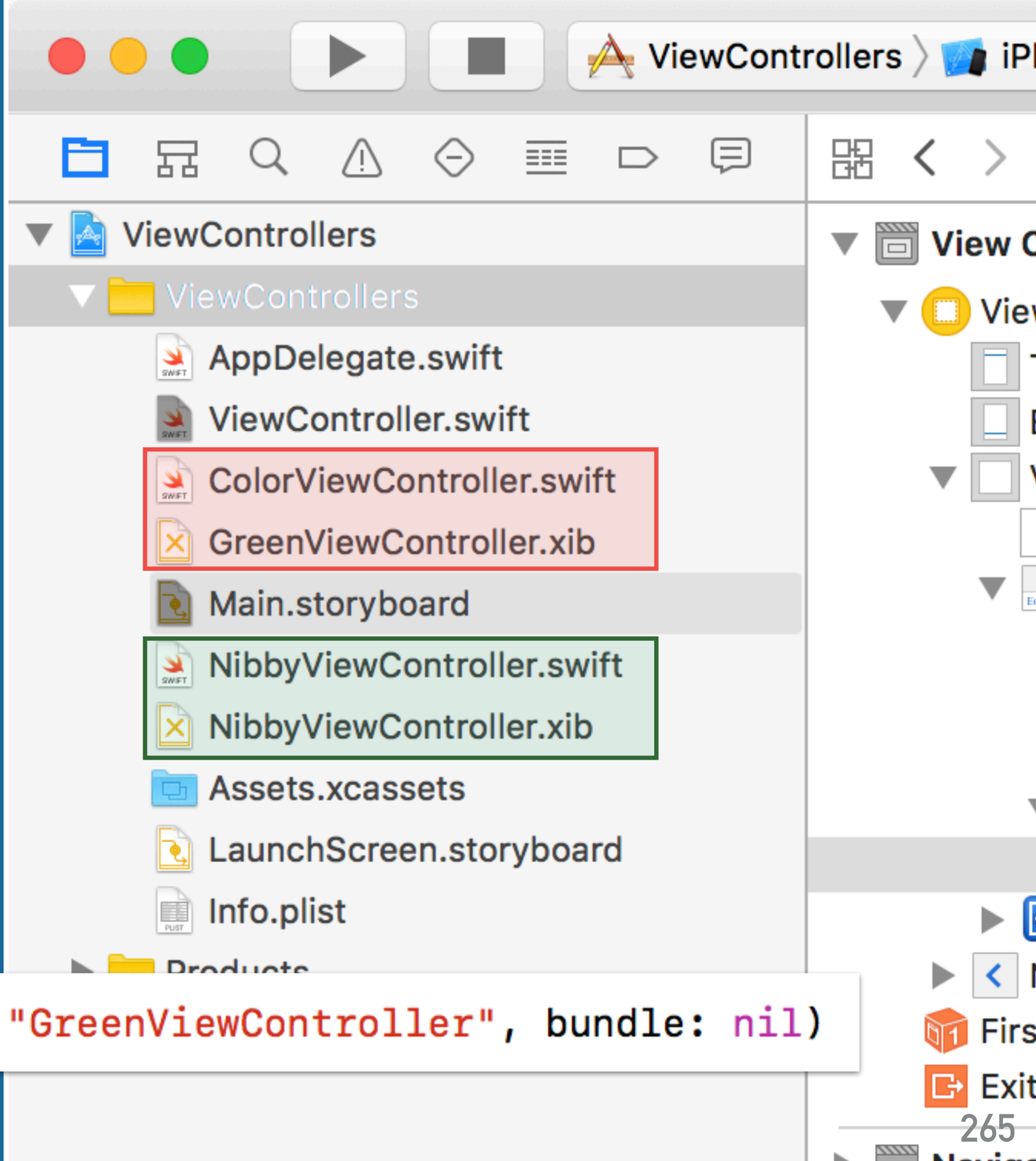
SUBTITLE

- 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()
```

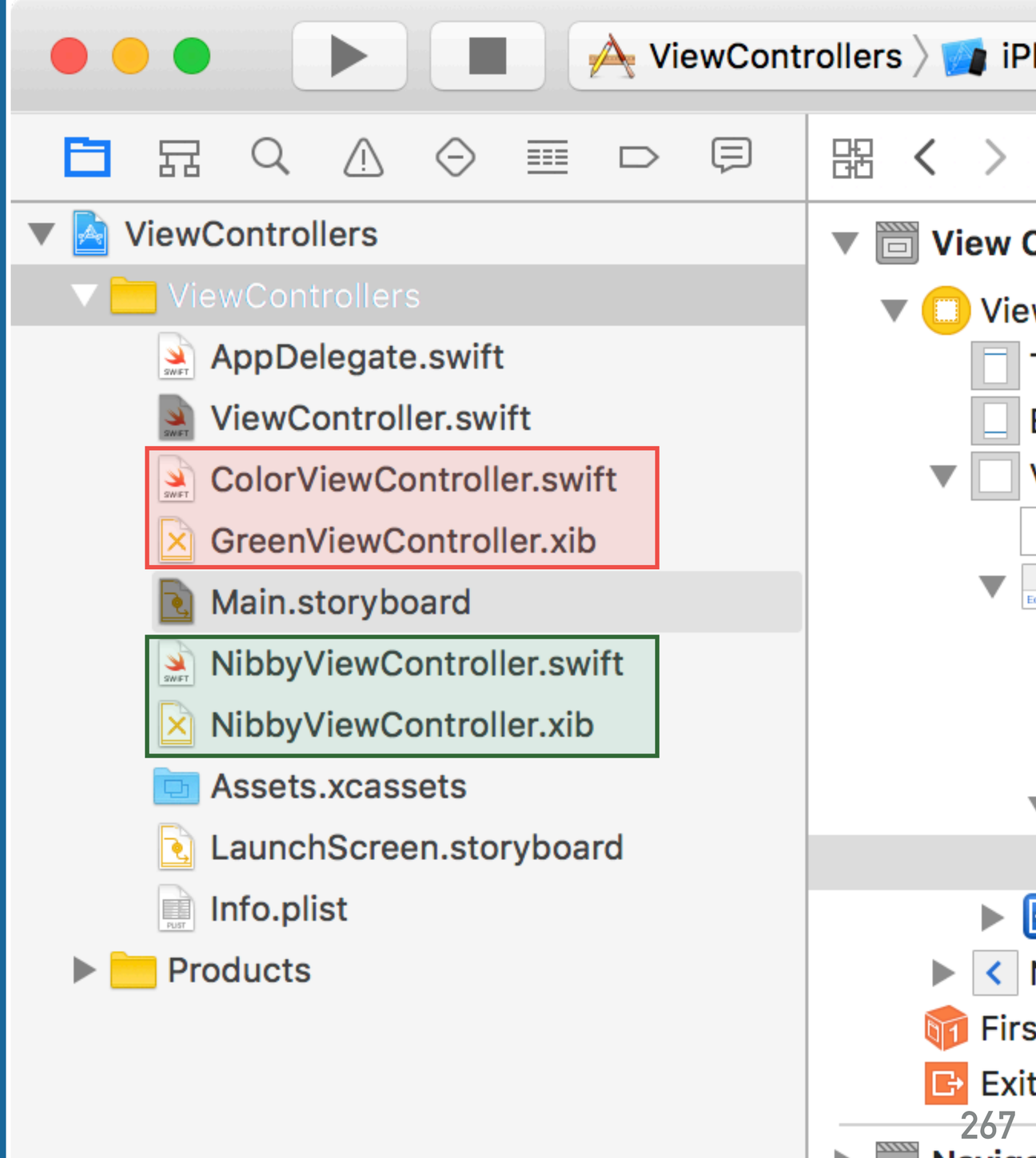
```
let gvc = ColorViewController(nibName: "GreenViewController", bundle: nil)
```

- Use GreenViewController.nib for ColorViewController class

NIB BASED VIEWCONTROLLERS

SUBTITLE

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



THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1F