ITM103 iOS Application Development

Topic 6: Navigation Controller

Objectives

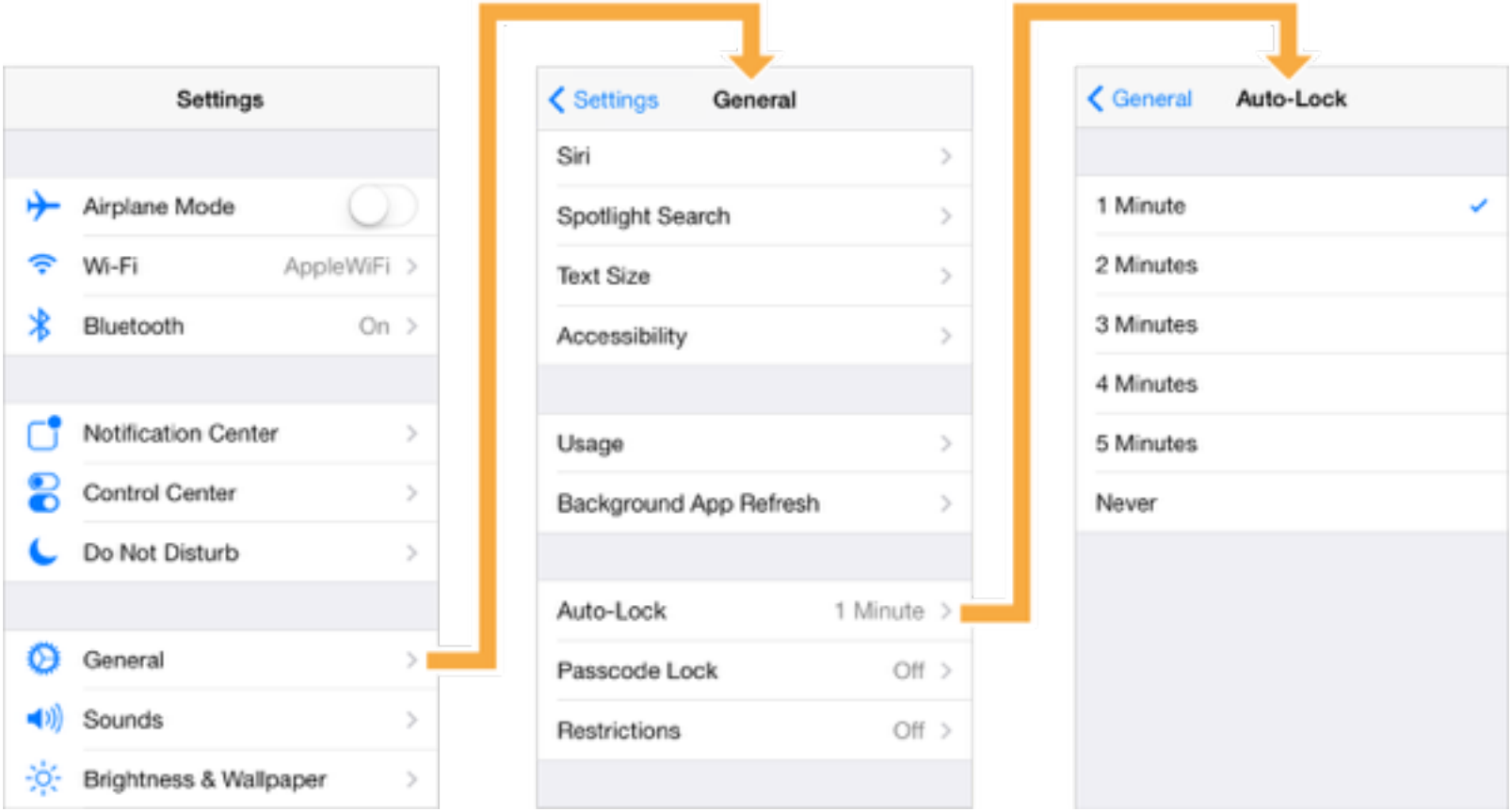
- By the end of the lesson, you will be able to:
  - Understand the hierarchy of UINavigationController
  - Understand the Navigation Bar Structure
  - Navigate with navigation bar
  - Pass data between view controllers

Navigation Controller

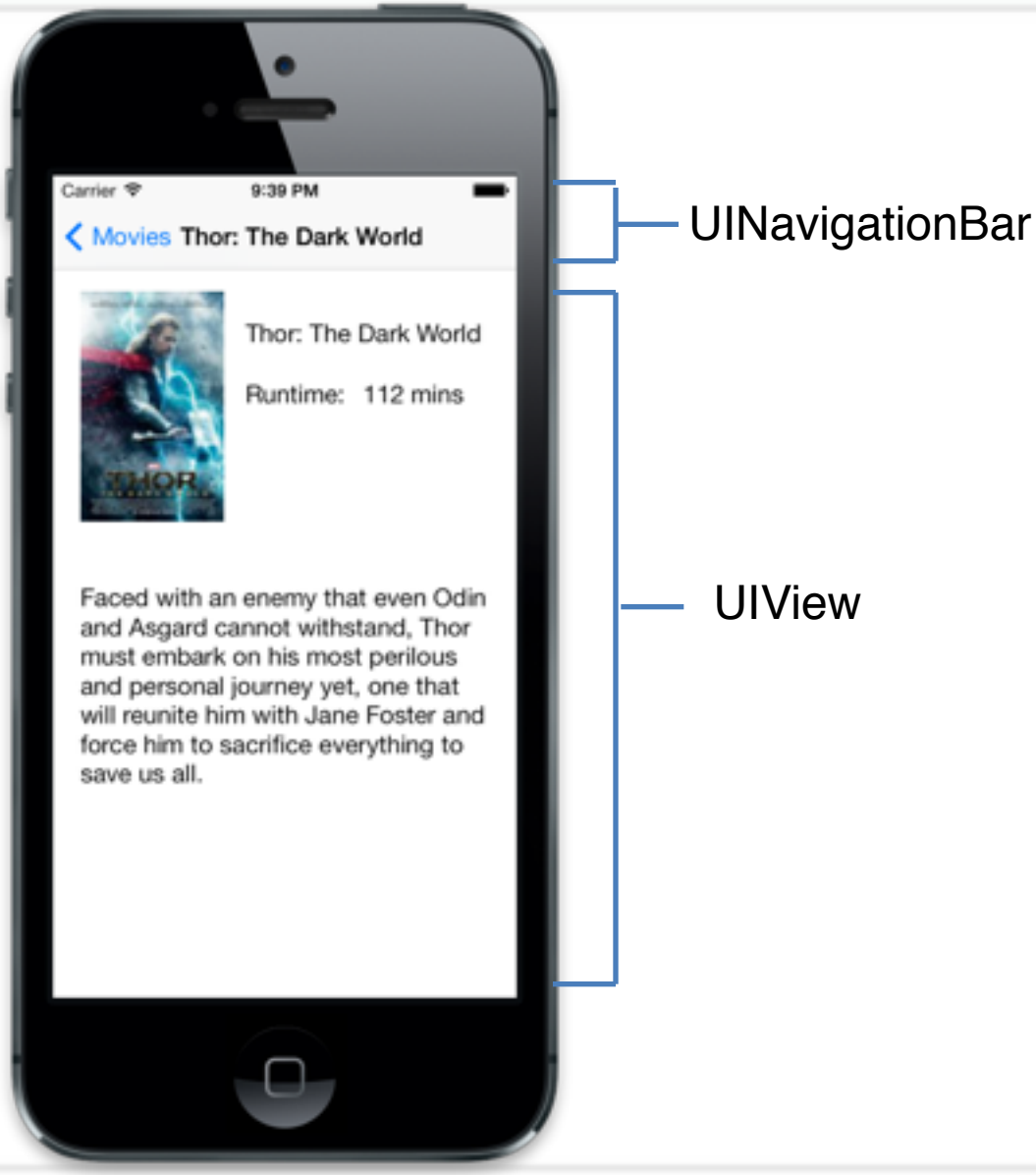
What is it about?

UINavigationController

- Manage multiple views of data in a hierarchy
- This hierarchy of views is represented on a stack.



UINavigationController

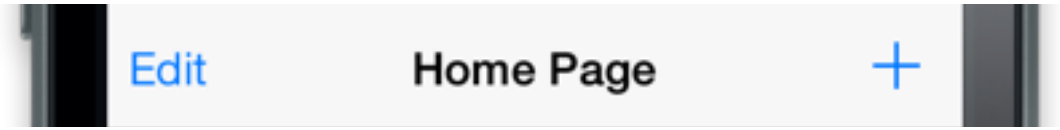


Navigation Controller

Components of the Navigation Controller

UIBarButtonItem

- You can design the nav bar in Storyboard.
- You can also do it in code:




```
self.navigationItem.title = "Home Page"

self.navigationItem.leftBarButtonItem =
    UIBarButtonItem(
        barButtonSystemItem: UIBarButtonSystemItem.Edit,
        target: self,
        action: #selector(editButtonClicked))
```

Indicates when object+method to call when clicked

```
self.navigationItem.rightBarButtonItem =
    UIBarButtonItem(
        barButtonSystemItem: UIBarButtonSystemItem.Add,
        target: self,
        action: #selector(addButtonClicked))
```

Item Positions on a navigation bar

Position	Property	Description
Left	`backBarButtonItem` `leftBarButtonItem`	In a navigation interface, the navigation controller assigns a Back button to the left position by default. To get the default Back button provided by the navigation controller, get the value of the `backBarButtonItem` property. To assign a custom button or view to the left position, and thereby replace the default Back button, assign a `UIBarButtonItem` object to the `leftBarButtonItem` property.
Center	`titleView`	In a navigation interface, the navigation controller displays a custom view with the title of your content view controller by default. You can replace this view as desired with your own custom view. If you do not provide a custom title view, the navigation bar displays a custom view with the navigation item's title string. Or, if the navigation item doesn't provide a title, the navigation bar uses the view controller's title.
Right	`rightBarButtonItem`	This position is empty by default. It is typically used to place buttons for editing or modifying the current screen. You can also place custom views here by wrapping the view in a `UIBarButtonItem` object.

Reference: https://developer.apple.com/library/IOs/documentation/WindowsViews/Conceptual/ViewControllerCatalog/Chapters/NavigationControllers.html#//apple_ref/doc/uid/TP40011313-CH2-SW5

UIBarButtonItem

UIBarButtonSystemItem.done	Done
UIBarButtonSystemItem.cancel	Cancel
UIBarButtonSystemItem.edit	Edit
UIBarButtonSystemItem.save	Save
UIBarButtonSystemItem.add	+
UIBarButtonSystemItem .flexibleSpace	Blank space to add between other items. The space is distributed equally between other items
UIBarButtonSystemItem .fixedSpace	Blank space to add between other items. Only the width property is used when this value is set.
UIBarButtonSystemItem.compose	
UIBarButtonSystemItem.reply	
UIBarButtonSystemItem.action	

UIBarButtonItem

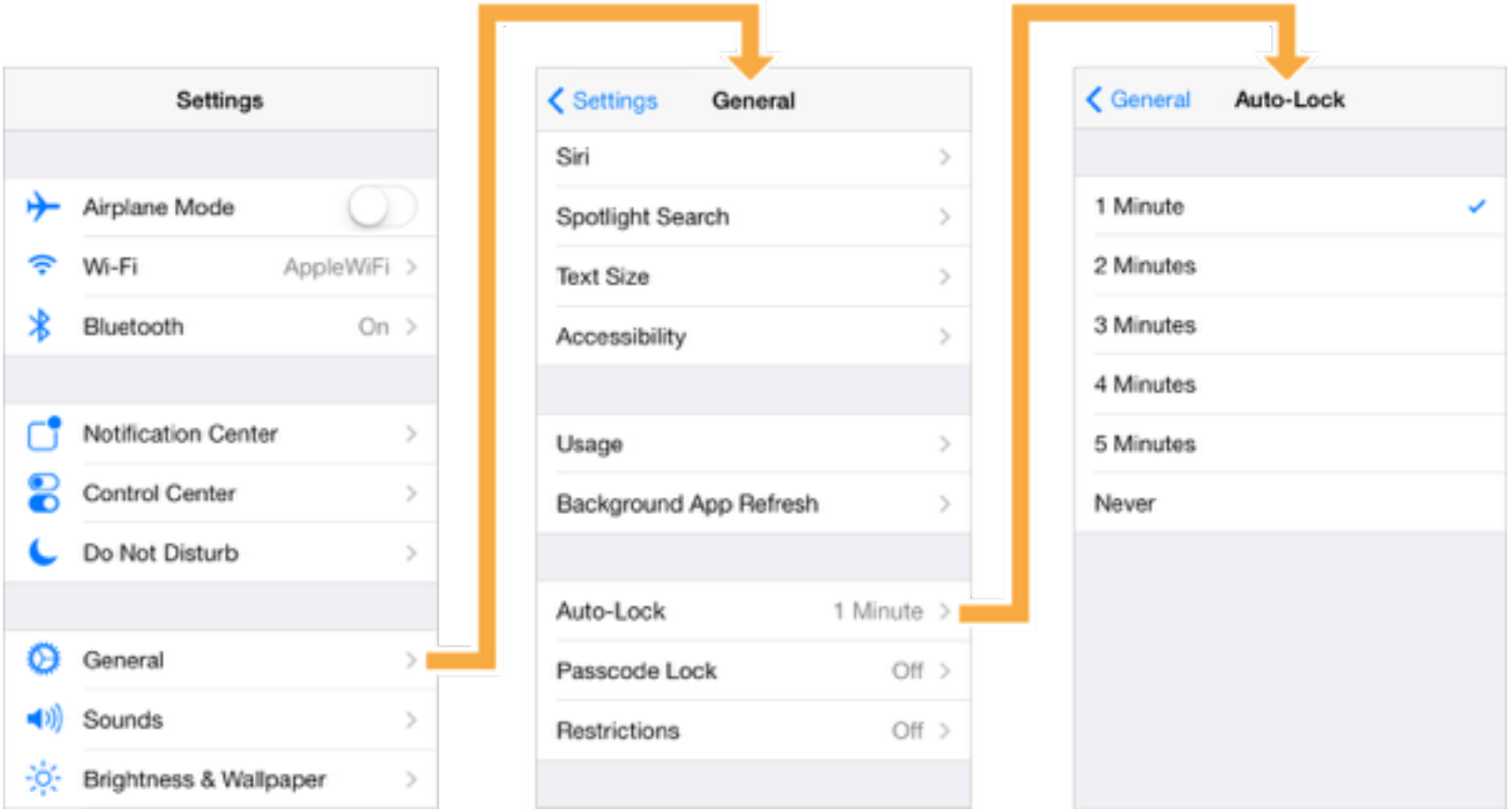
UIBarButtonSystemItem.organize	
UIBarButtonSystemItem.bookmarks	
UIBarButtonSystemItem.search	
UIBarButtonSystemItem.refresh	
UIBarButtonSystemItem.stop	
UIBarButtonSystemItem.camera	
UIBarButtonSystemItem.trash	
UIBarButtonSystemItem.play	
UIBarButtonSystemItem.pause	
UIBarButtonSystemItem.rewind	
UIBarButtonSystemItem.fastForward	
UIBarButtonSystemItem.undo	Undo
UIBarButtonSystemItem.redo	Redo
UIBarButtonSystemItem.pageCurl	

Navigation Controller

Navigation to the Next Screen

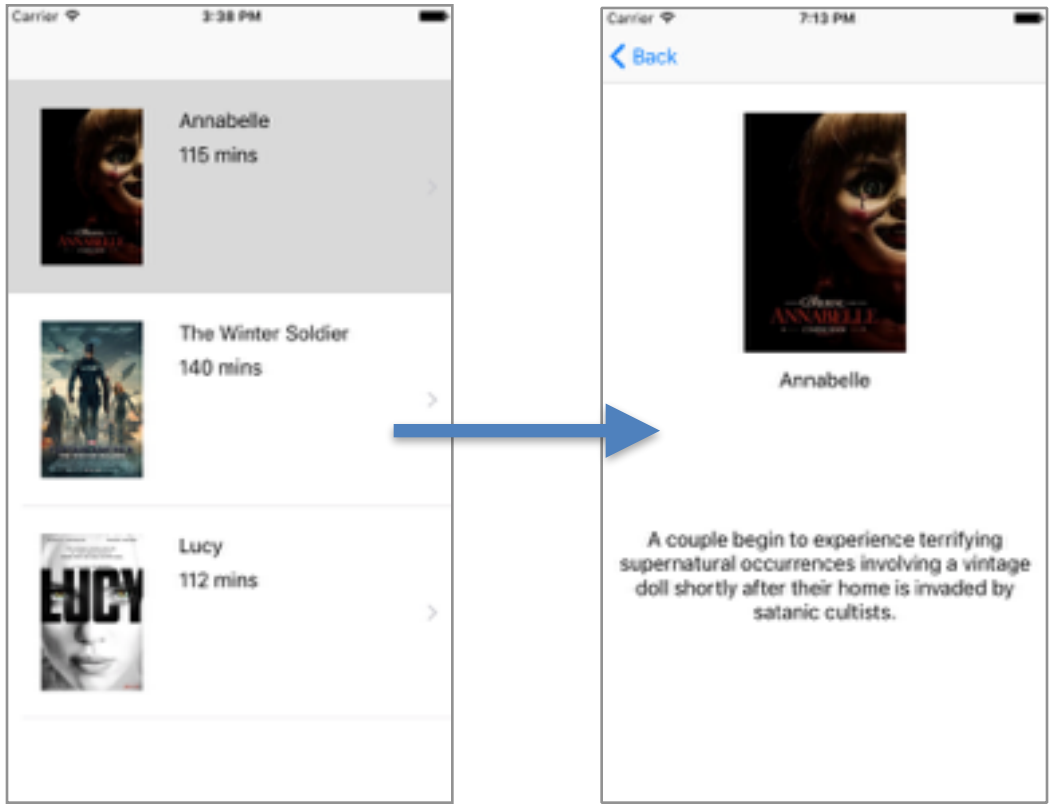
Why?

- Allow user to go into a detail page



Why?

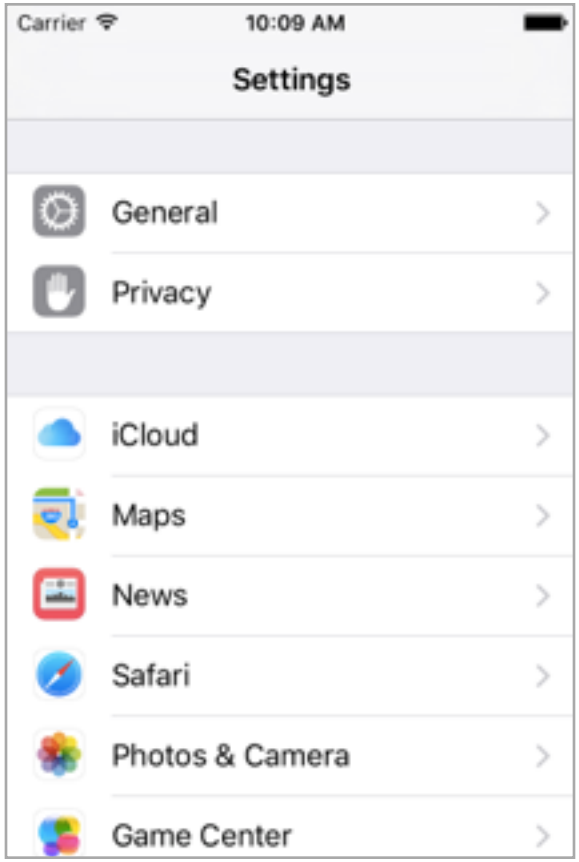
- Allow user to go into a detail page



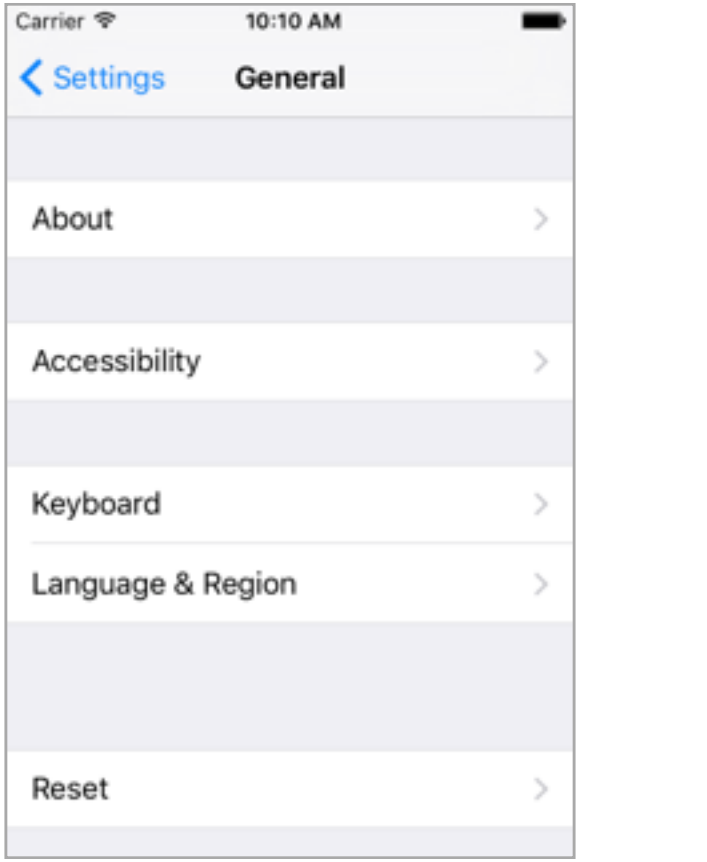
Imagine…

- Each screen is a card placed in a stack.
- New screens are **push**ed on top of the other.
- Screens are **pop**ped to go back to the previous screen.

Imagine…


Carrier
10:09 AM
Settings
General
Privacy
iCloud
Maps
News
Safari
Photos & Camera
Game Center

Imagine…

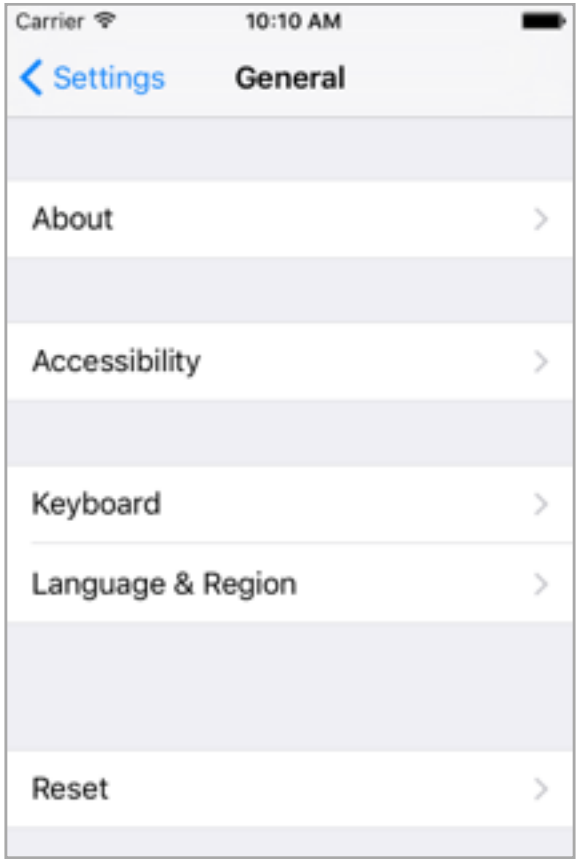


Push

Imagine…

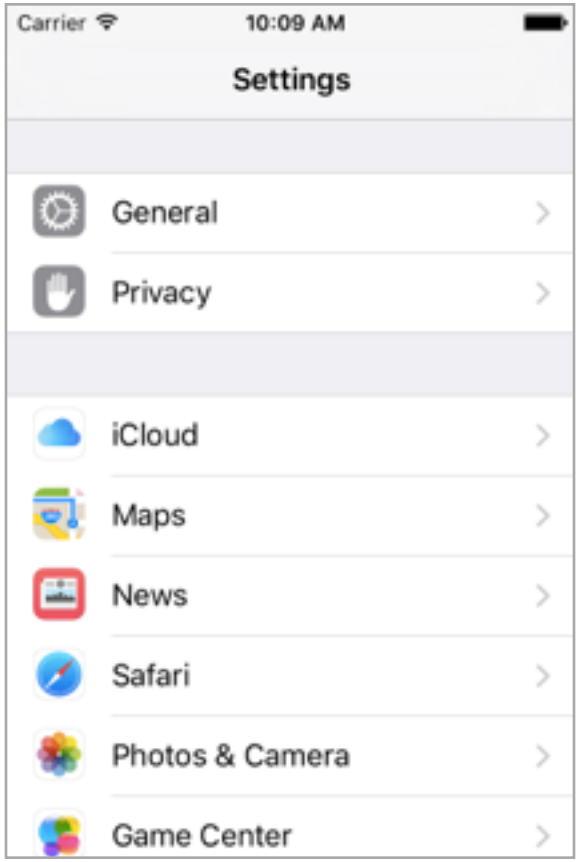
Push

Imagine…


Carrier
10:10 AM
Settings
General
About
Accessibility
Keyboard
Language & Region
Reset

Pop

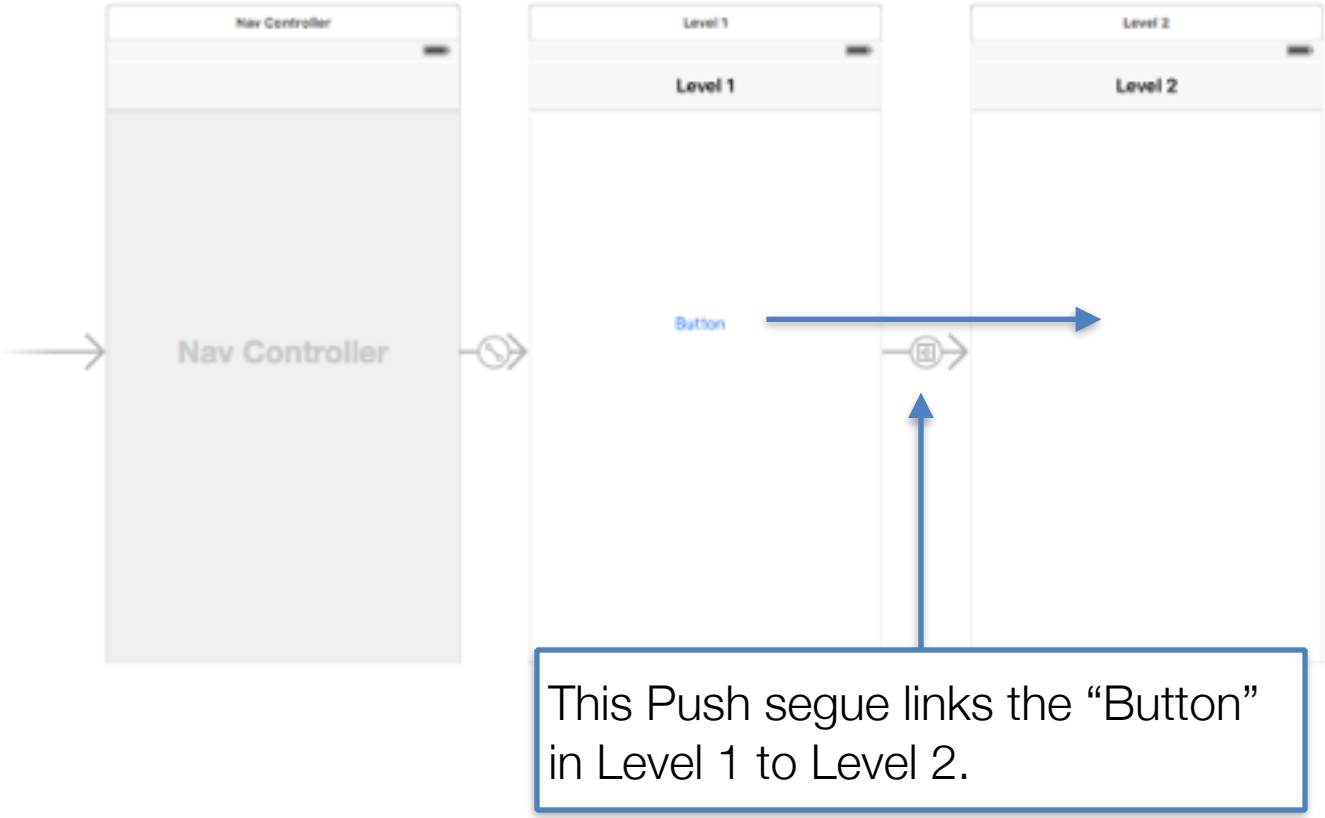
Imagine…


Carrier
10:09 AM
Settings
General
Privacy
iCloud
Maps
News
Safari
Photos & Camera
Game Center

Pop

Pushing to Stack

- You can design the pushing of a new screen in Storyboard using the **Push** segue



Pushing to Stack

- You can also do it manually in Swift:

```
// Loads the Main.storyboard file
let s = UIStoryboard(name: "Main", bundle: nil)


// Loads next level View Controller from
// Main.storyboard
let v = s.instantiateViewController(withIdentifier:
    "Level2ViewController")


// Pushes the newly loaded view controller on
// to the stack.
self.navigationController?.pushViewController(v, animated: true)
```

Navigation Controller

Passing Data to the Next Screen

Pass data to another View Controller

MovieDetailViewController.swift

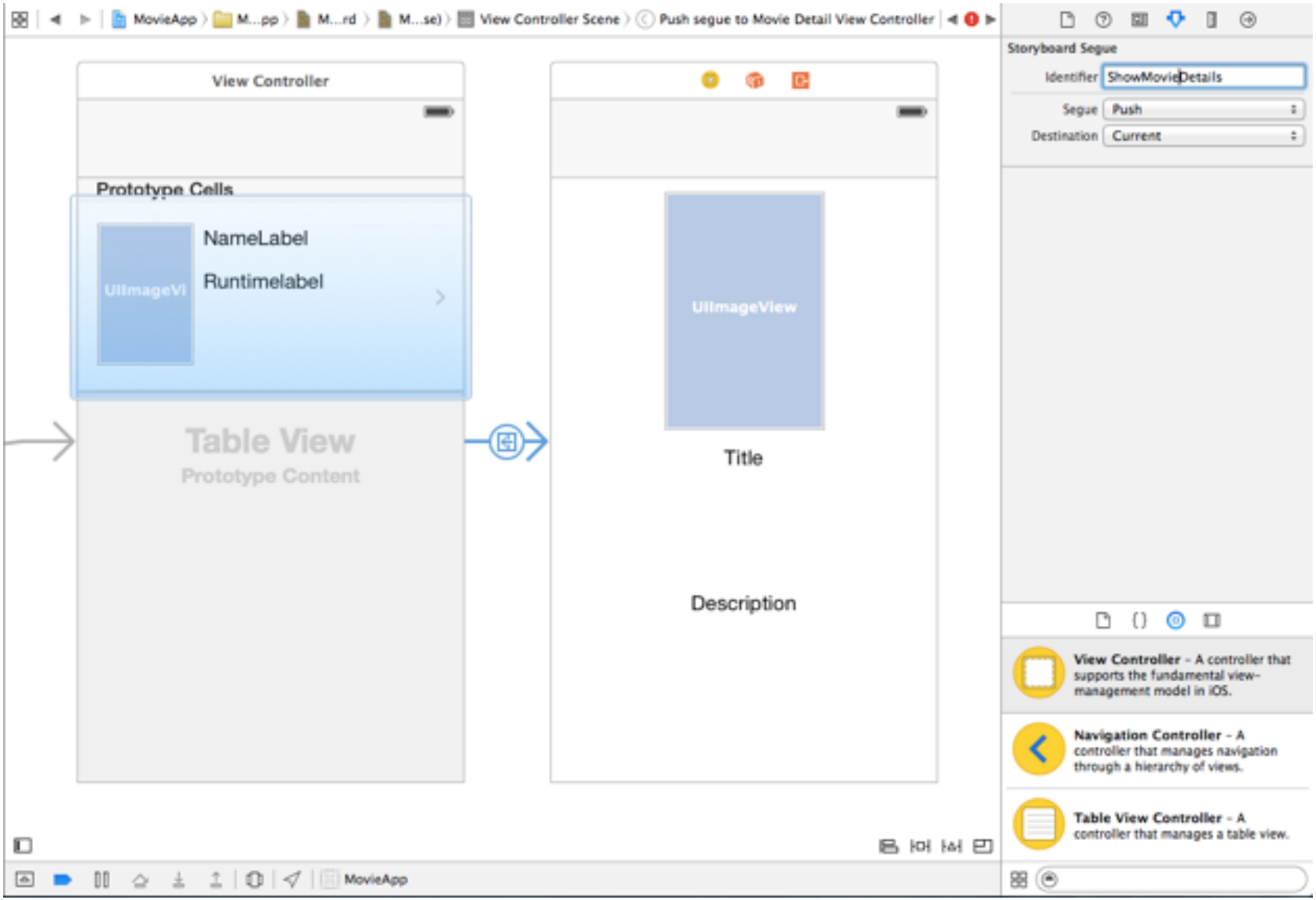
```
class MovieDetailViewController: UIViewController {
    @IBOutlet weak var movieImage: UIImageView!
    @IBOutlet weak var titleLabel: UILabel!
    @IBOutlet weak var descriptionLabel: UILabel!


    var movieItem : Movie! {
        didSet {
            self.navigationItem.title = movieItem.movieName
        }
    }


    override func viewWillAppear(animated: Bool) {
        super.viewWillAppear(animated)

        if movieItem != nil {
            titleLabel.text = movieItem.movieName
            descriptionLabel.text = movieItem.movieDesc
            movieImage.image = UIImage(named: movieItem.imageName)
        }


    }
}
```

This property is created so that the view controller before this can set which **Movie** object to show.

<u>NOTICE</u> how you do **NOT** need to declare a class field to hold the value of the property?

Set the views to show details about the movie just before this view controller appears.

Pass data to another View Controller



Pass data to another View Controller

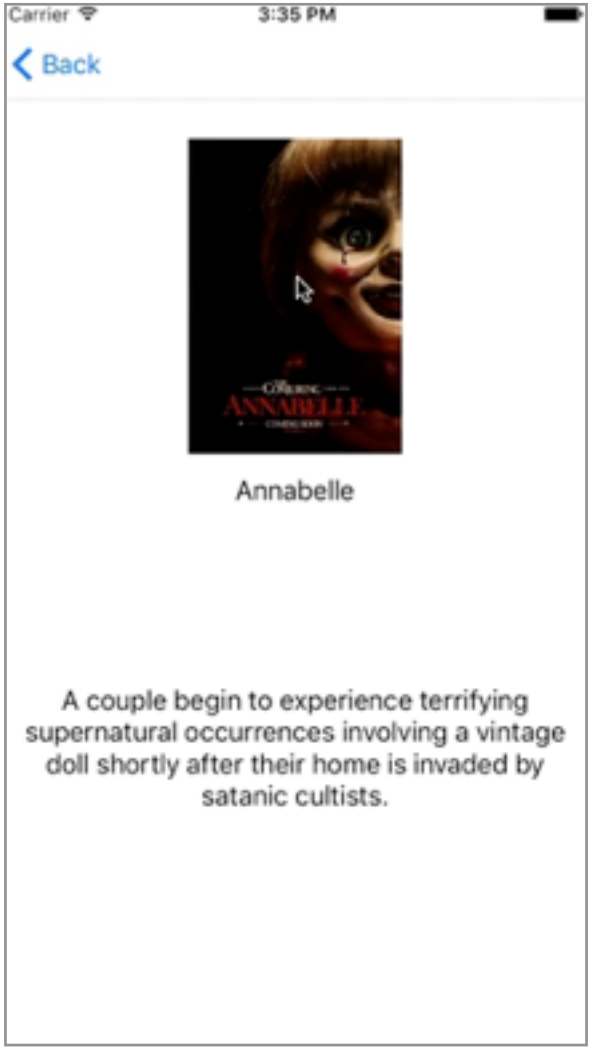
```
override func prepare(for segue: UIStoryboardSegue, sender: Any?)
{

    if(segue.identifier == "ShowMovieDetails")
    {
        let detailViewController = segue.destination
            as! MovieDetailViewController
        let myIndexPath = self.tableView.indexPathForSelectedRow

        if(myIndexPath != nil)
        {
            let movie = appDelegate.movieList[myIndexPath!.row]
            detailViewController.movieItem = movie
        }

    }
}
```

If the segue is the one we set up in the storyboard…

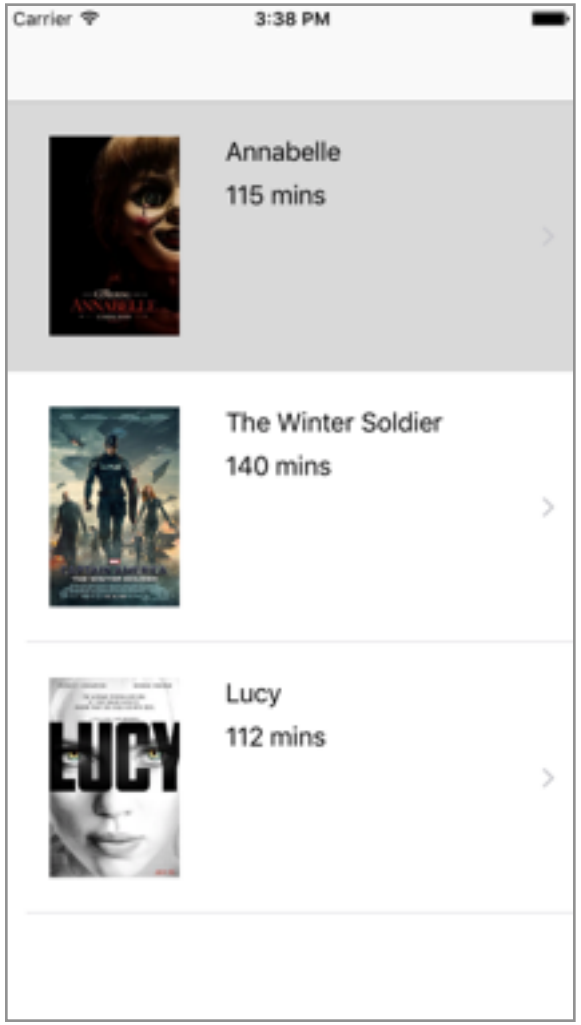
Set up the next view controller to show the movie the user selected.

Pass data to another View Controller

How it looks when everything is hooked up:



Pass data to another View Controller



User taps on movie

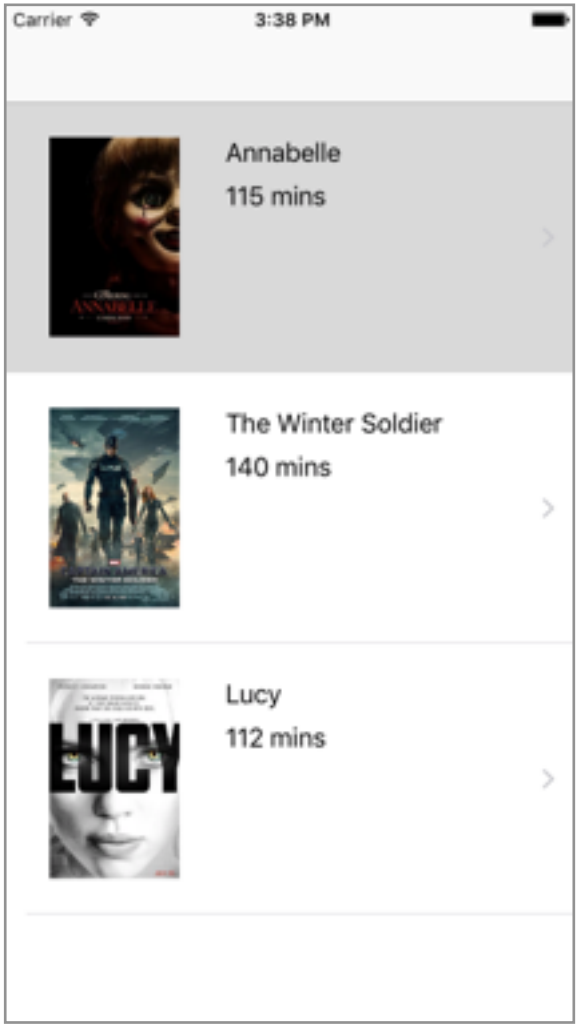
current ViewController
prepare(for segue)
pass data to next-level controller

next-level ViewController
viewDidLoad

next-level ViewController
viewWillAppear

next-level ViewController
viewDidAppear

Pass data to another View Controller



User taps on movie

current ViewController
prepare(for segue)
pass data to next-level controller

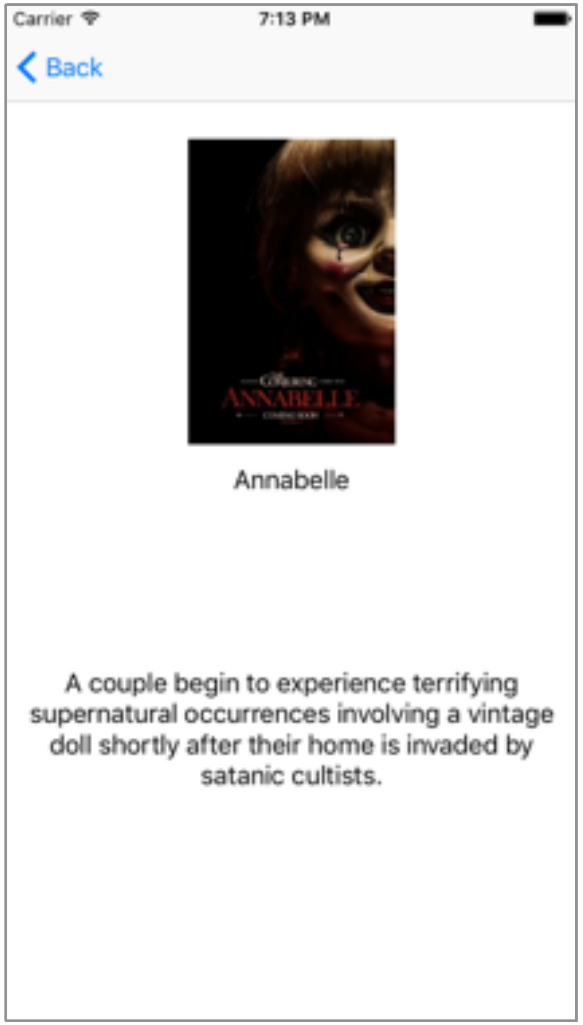
next-level ViewController
viewDidLoad

next-level ViewController
viewWillAppear

next-level ViewController
viewDidAppear

next-level View Controller Appears

Pass data to another View Controller



User taps on movie

current ViewController
prepare(for segue)
pass data to next-level controller

next-level ViewController
viewDidLoad

next-level ViewController
viewWillAppear

next-level ViewController
viewDidAppear

next-level View Controller Appears

Navigation Controller

Navigation to the Previous Screen

Popping to the Previous Screen

- You have little control over the behaviour of the default Back button.

- If you want to do some processing when the user taps the Back button, you can:
  - hide the default Back button,
  - show your own button,
  - write code to handle what happens

Popping to the Previous Screen

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Hide the back button, but add our own cancel button
    //
    self.navigationItem.hidesBackButton = true
    self.navigationItem.leftBarButtonItem =
        UIBarButtonItem(
            barButtonSystemItem: UIBarButtonSystemItem.Cancel,
            target: self,
            action: #selector(cancelButtonPressed))
}
```

Hides the Back button, and adds our own Cancel button.

```
func cancelButtonPressed(sender: AnyObject)
{
    // Do any validations before popping
    // to the previous screen.


    self.navigationController?.popViewController(animated: true)
}
```

Here you can do any validations you need, and decide when to pop back to the previous screen.

Summary

- UINavigationController
- Navigation Controller Hierarchy
- Navigation Bar Structure
- UIBarButtonItem
- Pass values to other view controllers