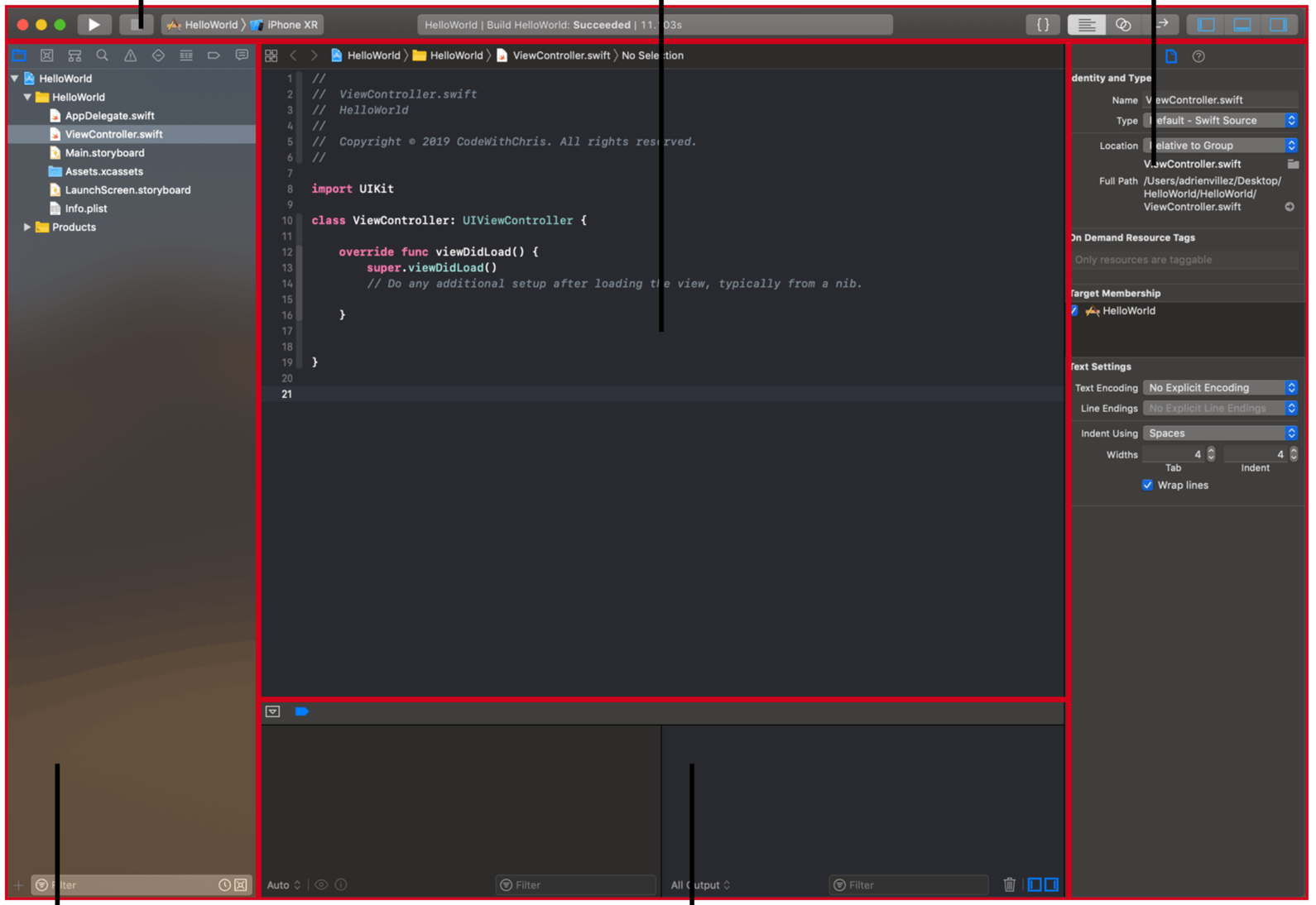
Workshop 2 - Part 1: Exploring Xcode

Let’s open Xcode!

Toolbar

Editor Area

Utility Area



Navigator Area

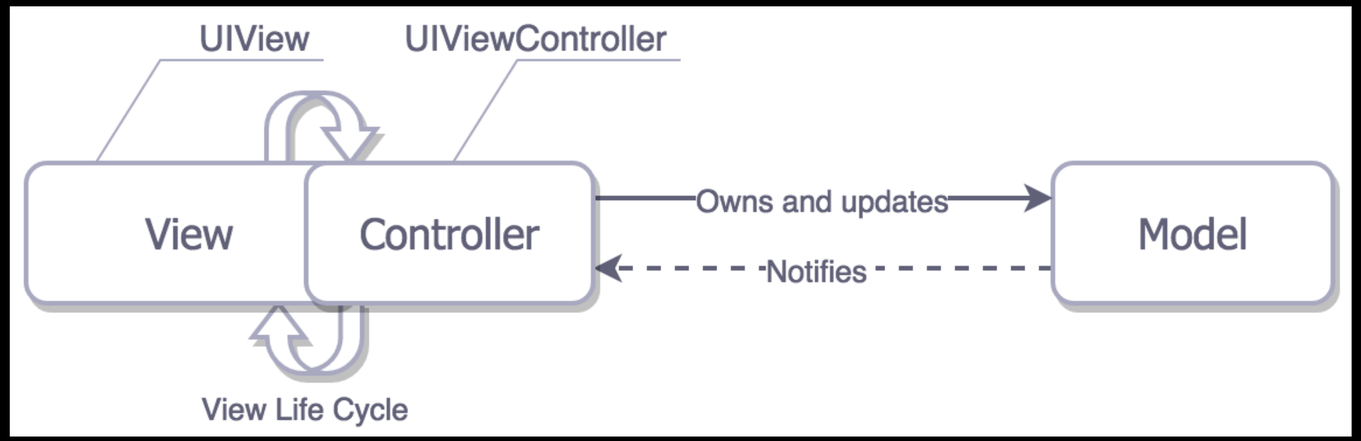
Debug Area

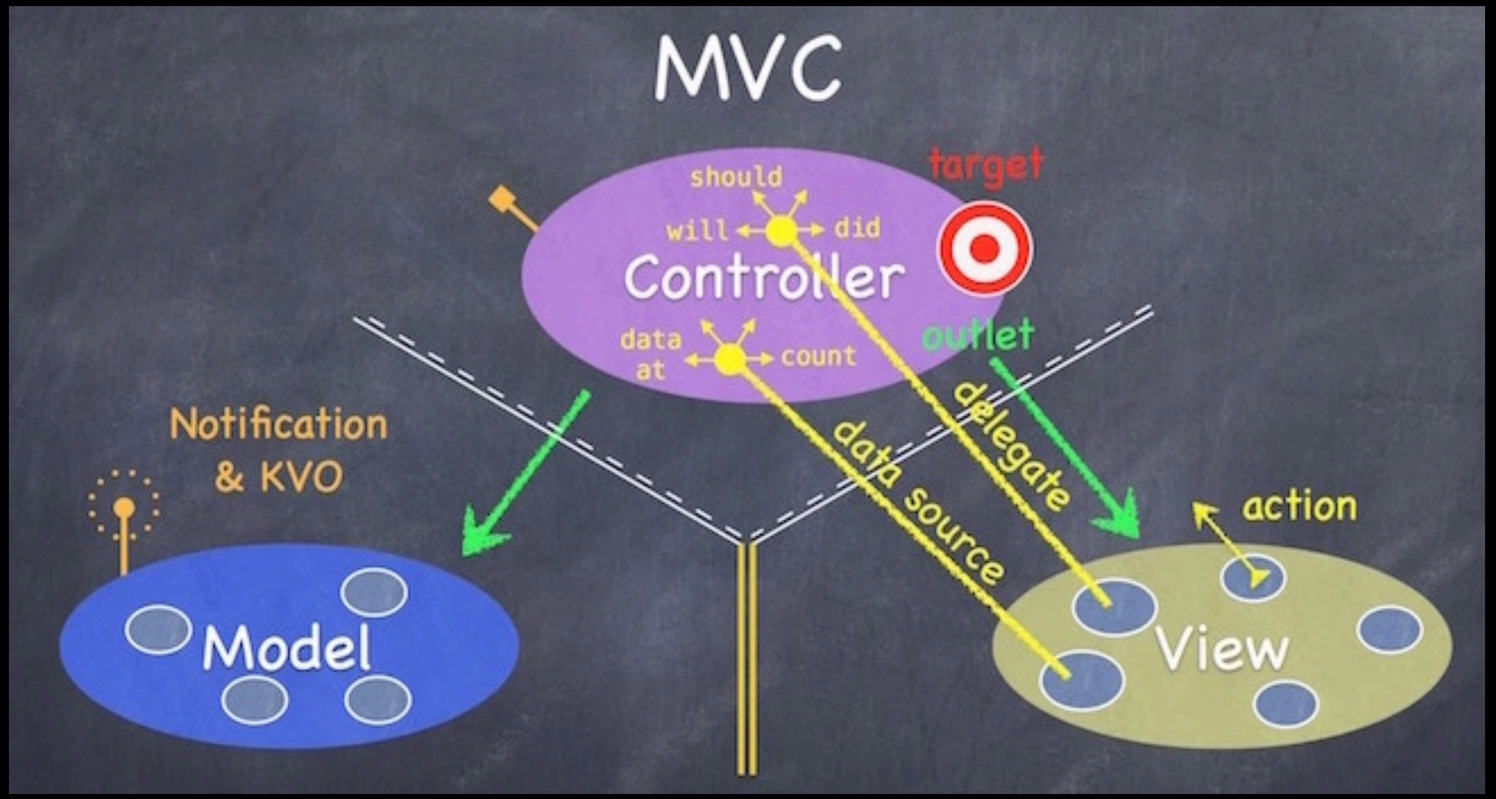
Project structure

- Launchers - AppDelegate, LaunchScreen
- Model - Objects, API calls, CoreData
- Resources - Assets, files, Localizations
- Supporting files - info.plist files, Helpers
- UI - Controllers, Views, Interface Builders

Main design pattern

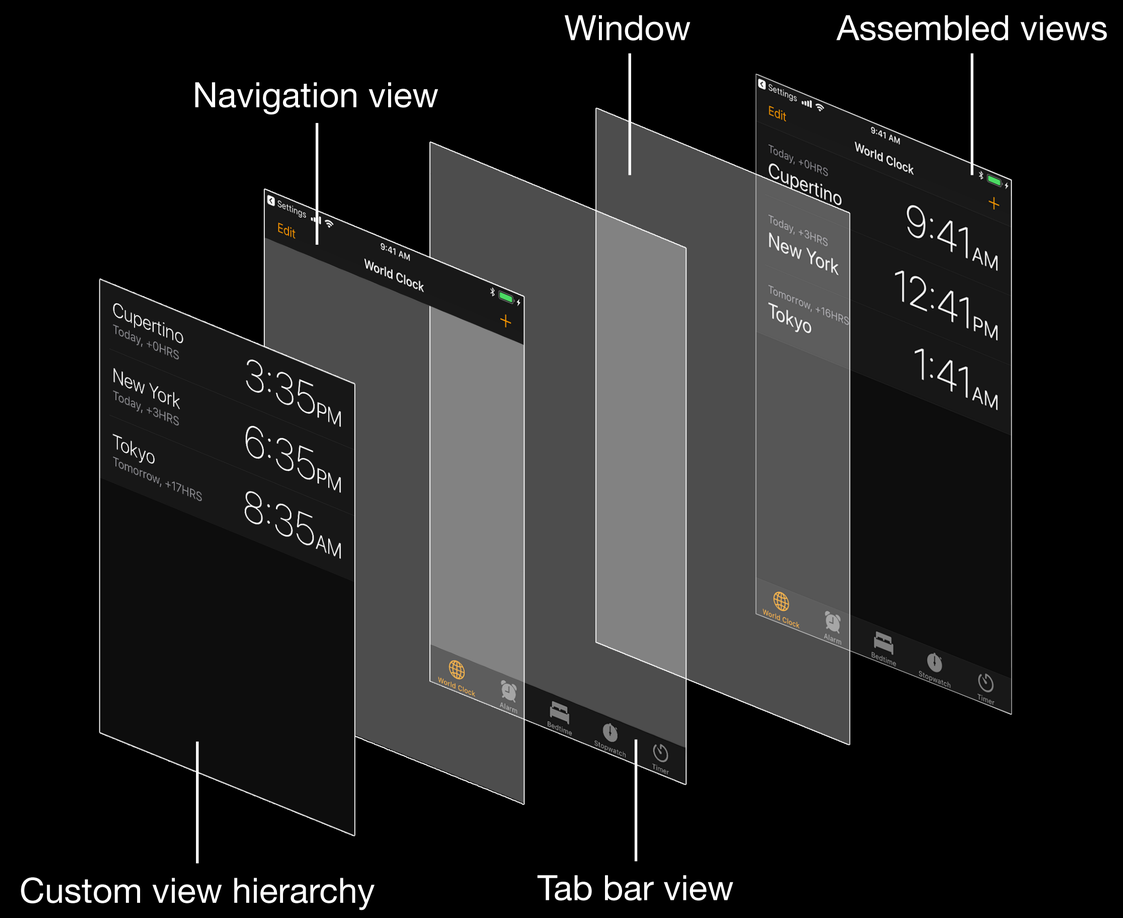
MVC - Model-View-Controller





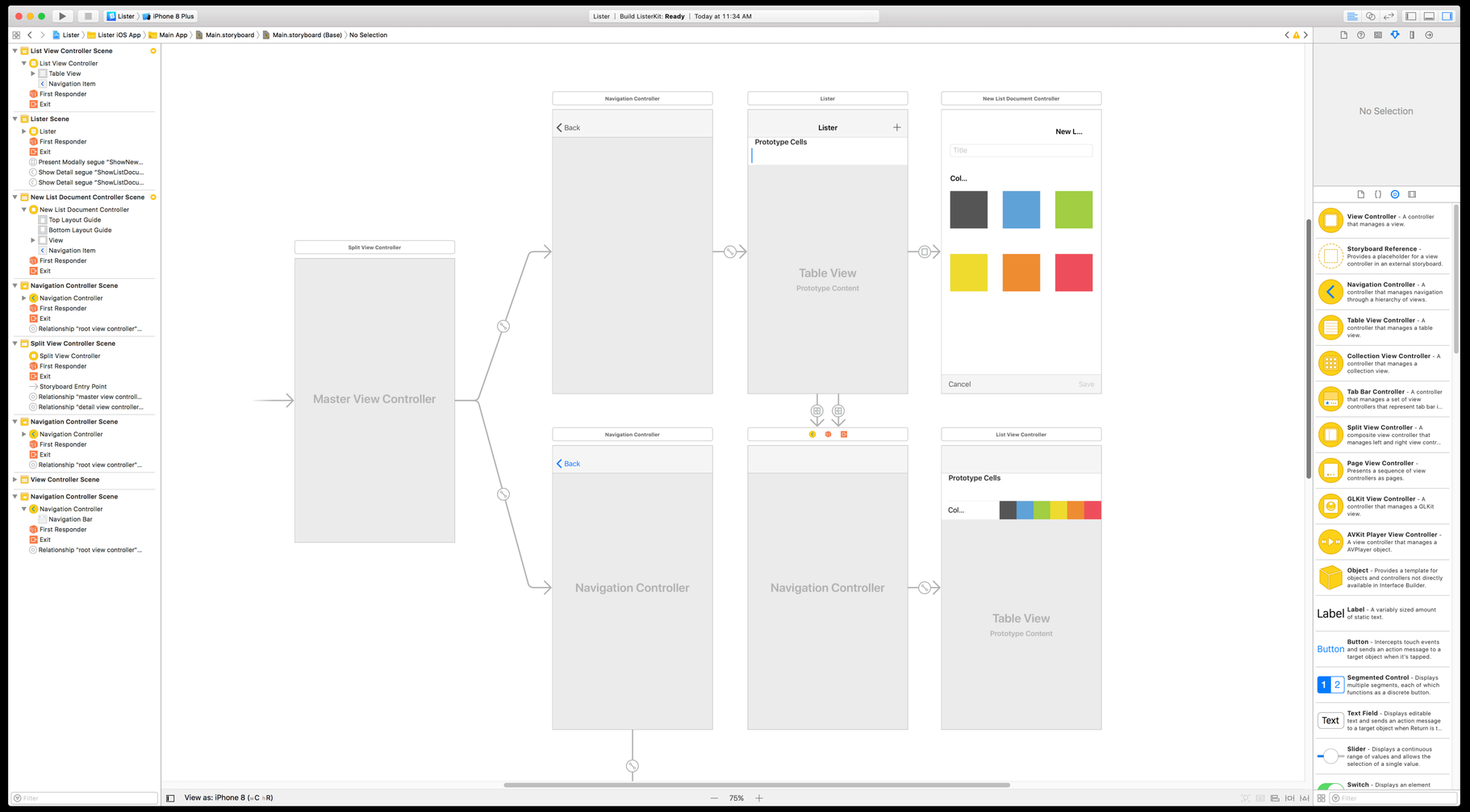
Workshop 2 - Part 2: Interface Builder Basics

Common system views



Interface Builder

Storyboards



Workshop 2 - Part 3:
UI Elements

UI Elements

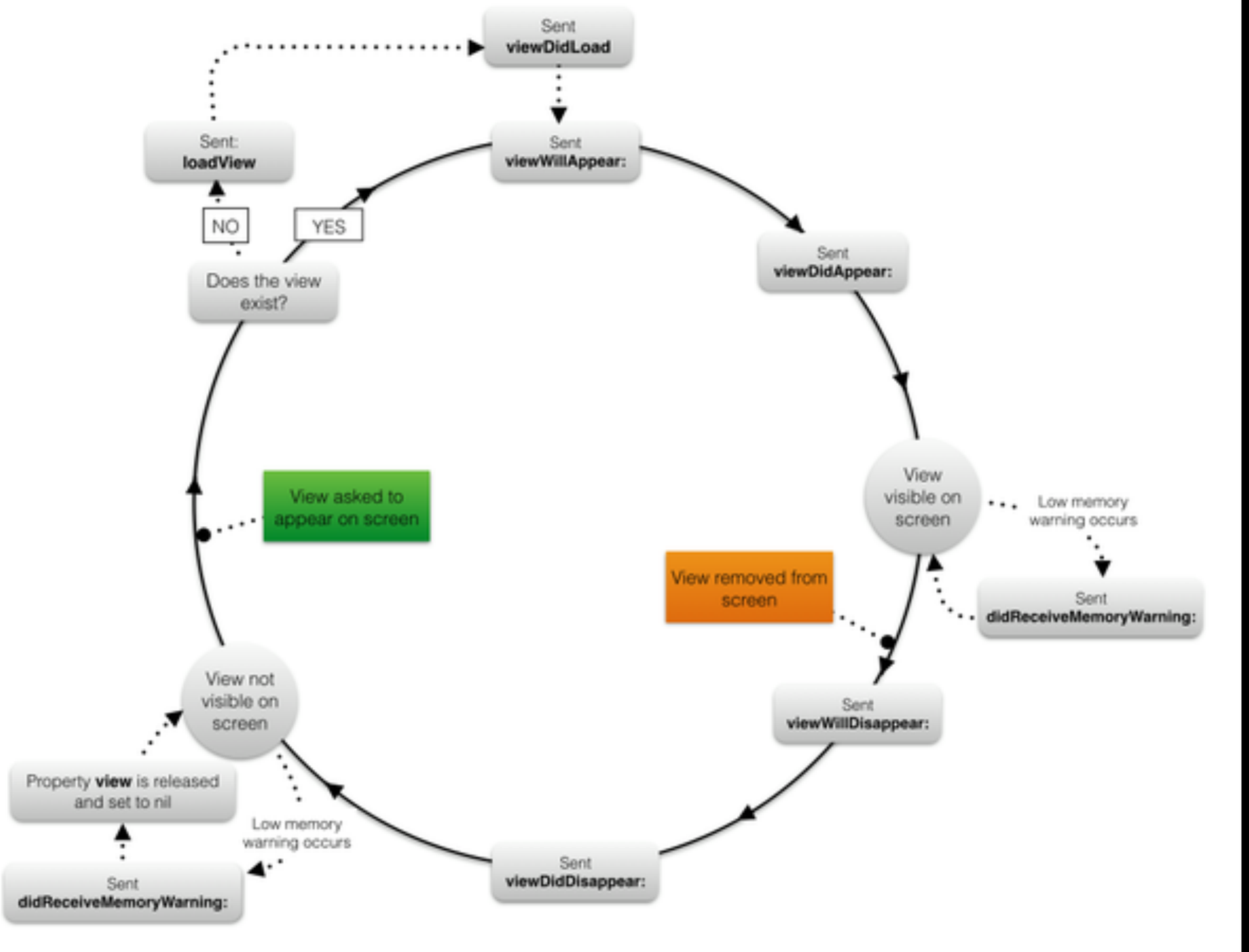
Categories

- Container views
- Bars
- Views
- Text
- Controls
- Content views

Container views

UIViewController

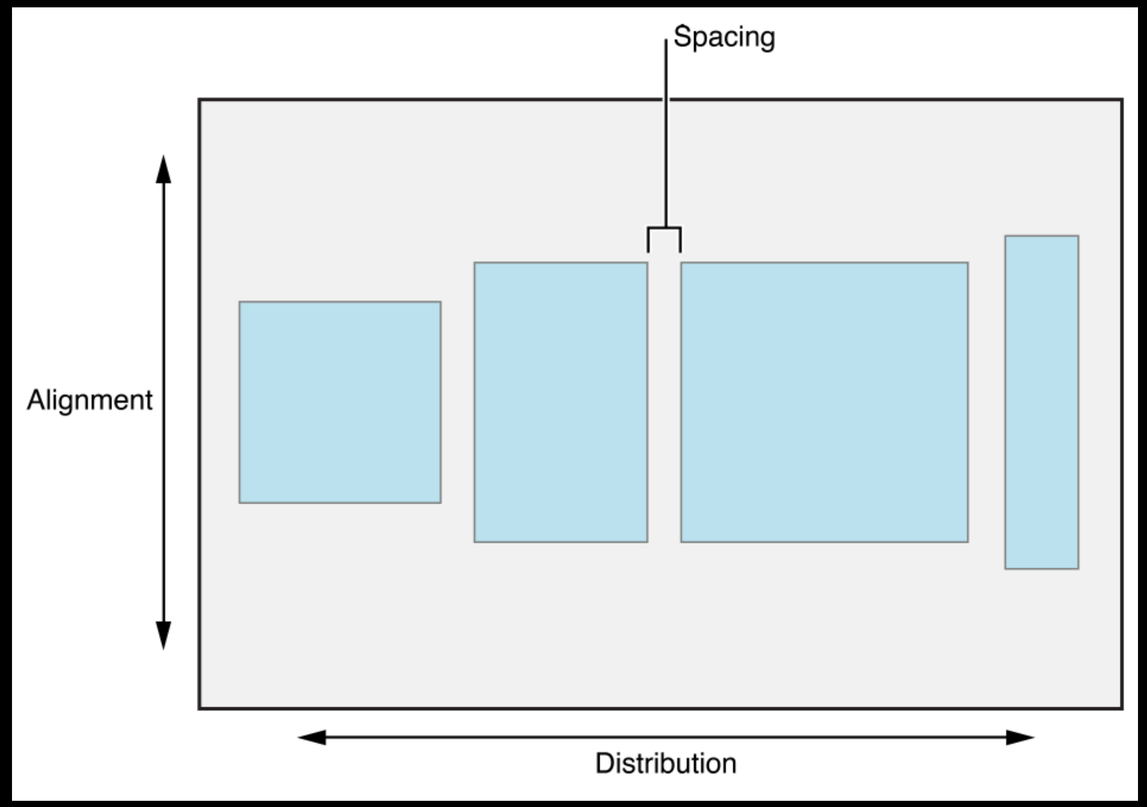
- An object that manages a view hierarchy for your UIKit app
- The most basic screen
- Contains just a view
- You can add other elements into it as subviews of its main view

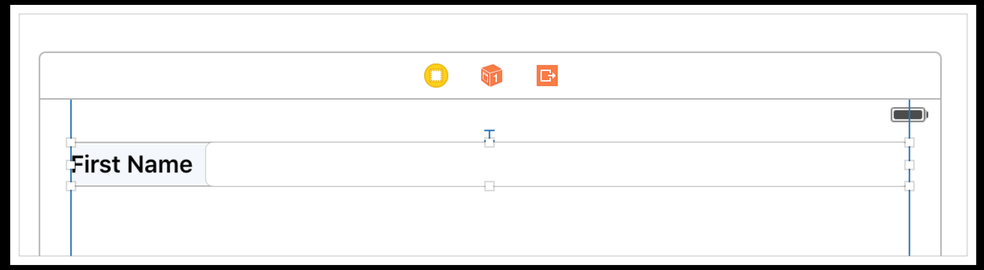

Sent
viewDidLoad
Sent:
loadView
Sent
viewWillAppear:
NO
YES
Does the view
exist?
Sent
viewDidAppear:
View asked to
appear on screen
View
visible on
screen
Low memory
warning occurs
View removed from
screen
Sent
didReceiveMemoryWarning:
Sent
viewWillDisappear:
View not
visible on
screen
Property view is released
and set to nil
Low memory
warning occurs
Sent
viewDidDisappear:
Sent
didReceiveMemoryWarning:

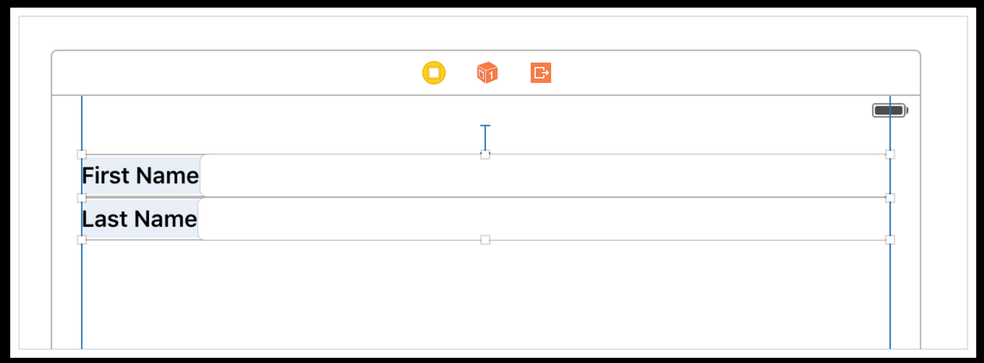
Container views

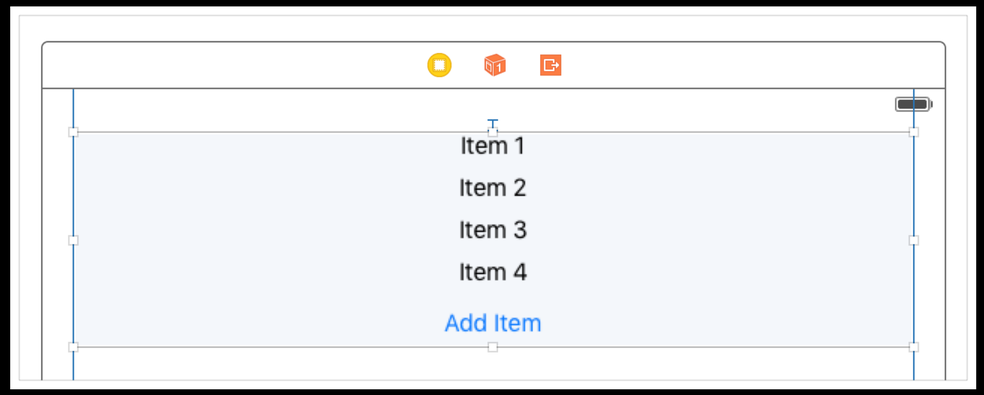
UIStackView

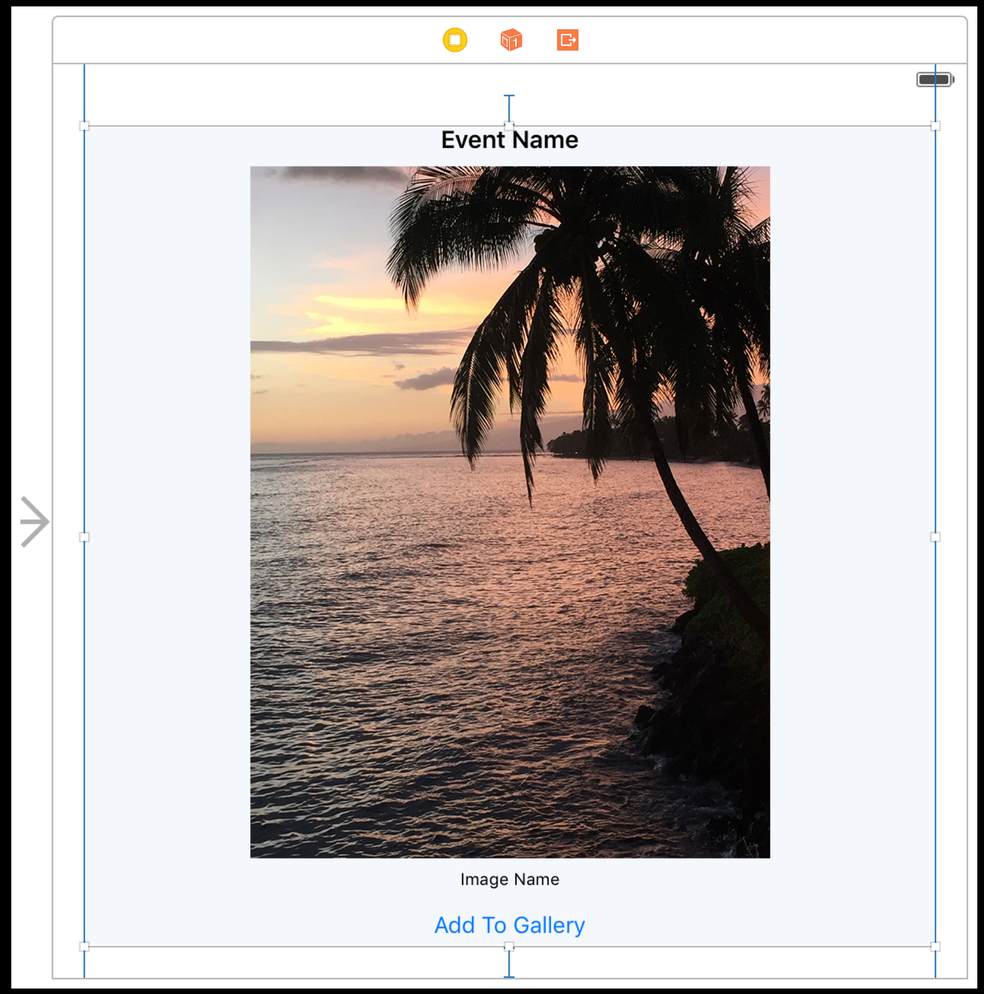
- A streamlined interface for laying out a collection of views in either a column or a row

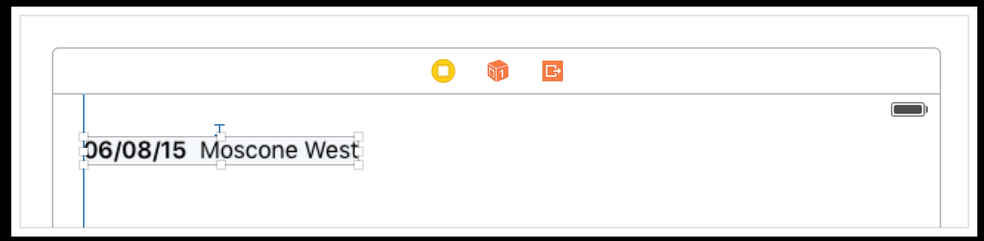



First Name


First Name
Last Name


Item 1
Item 2
Item 3
Item 4
Add Item


Event Name
Image Name
Add To Gallery


06/08/15 Moscone West

Container views

UIScrollView

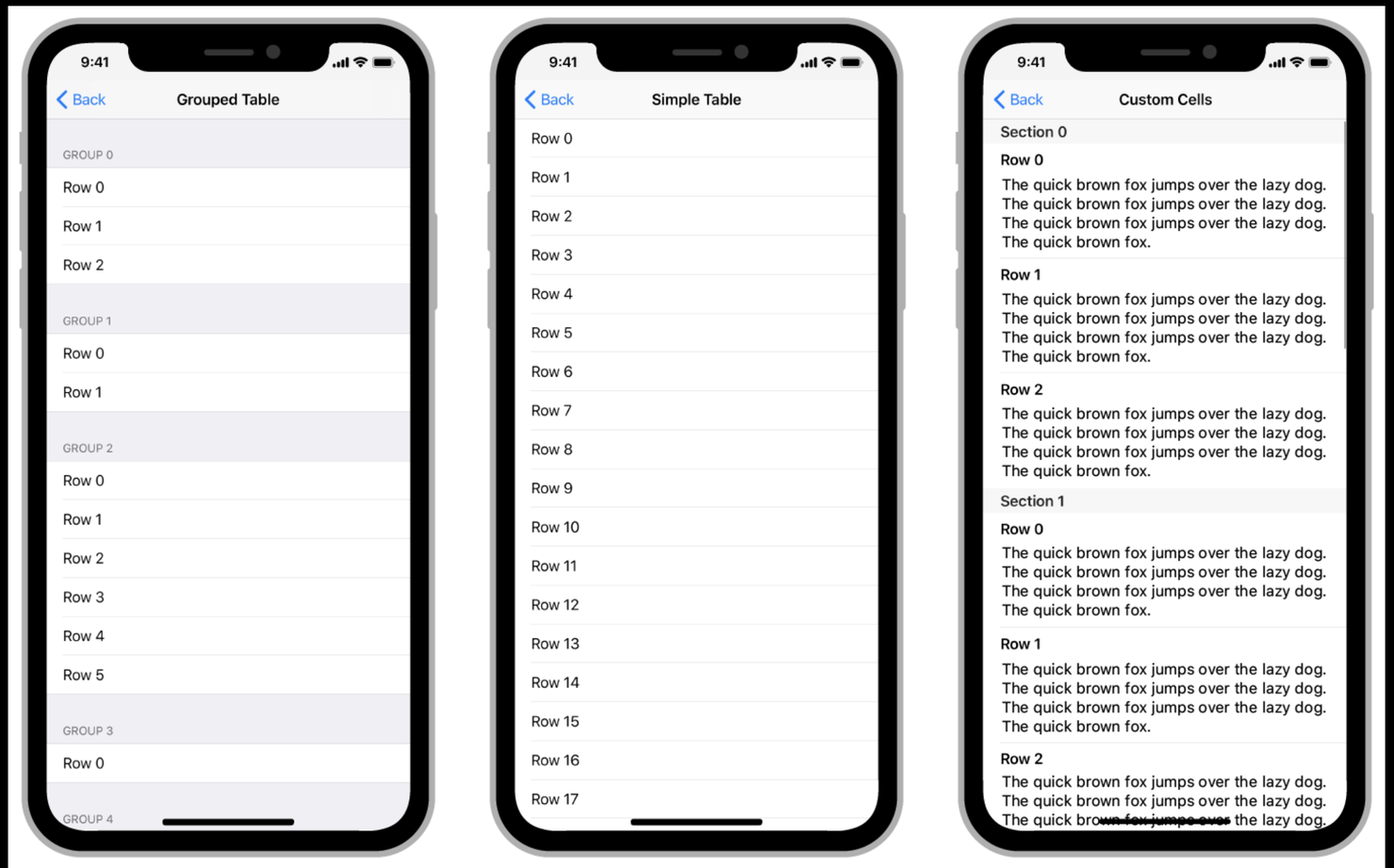
- A view that allows the scrolling and zooming of its contained views
- Used very rarely (better alternatives)
- Great for specific purposes

Container views

Table Views

- Display data in a single column of customizable rows

- UITableView
- UITableViewController

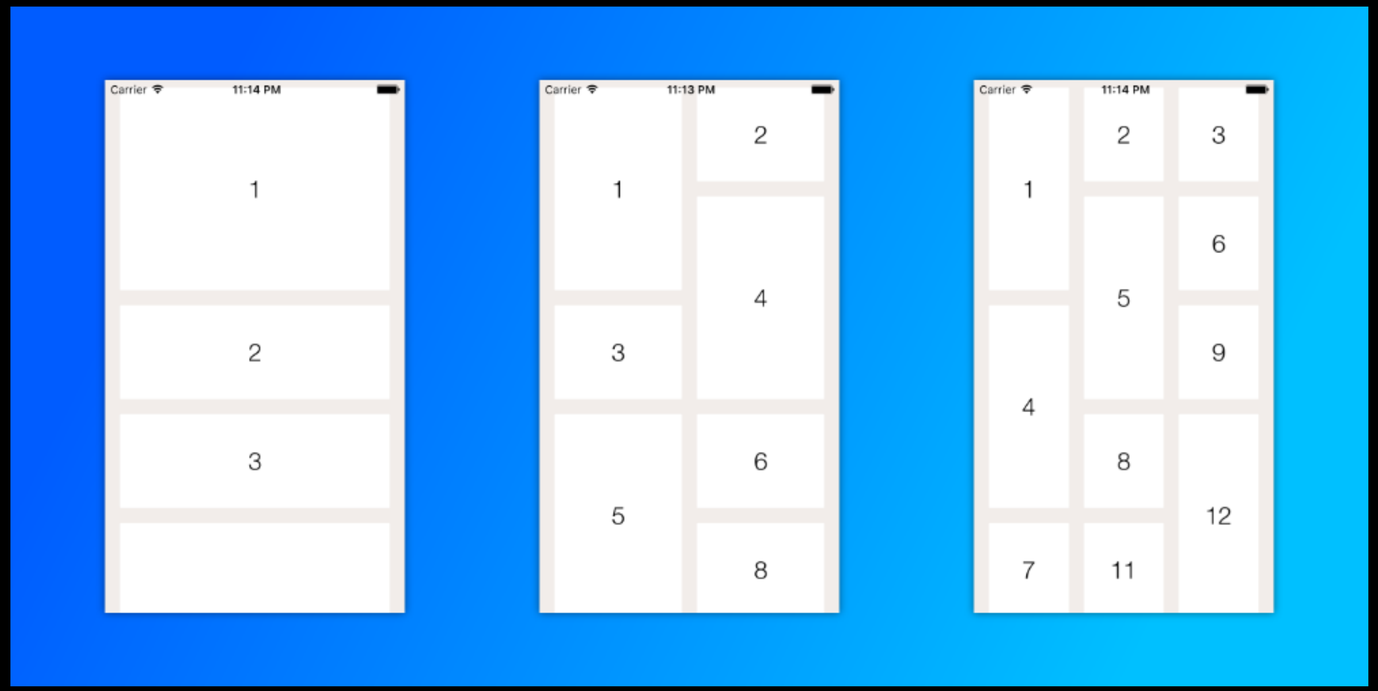


Container views

Collection Views

- Display nested views using a configurable and highly customizable layout

- UICollectionView
- UICollectionViewController



UITableView

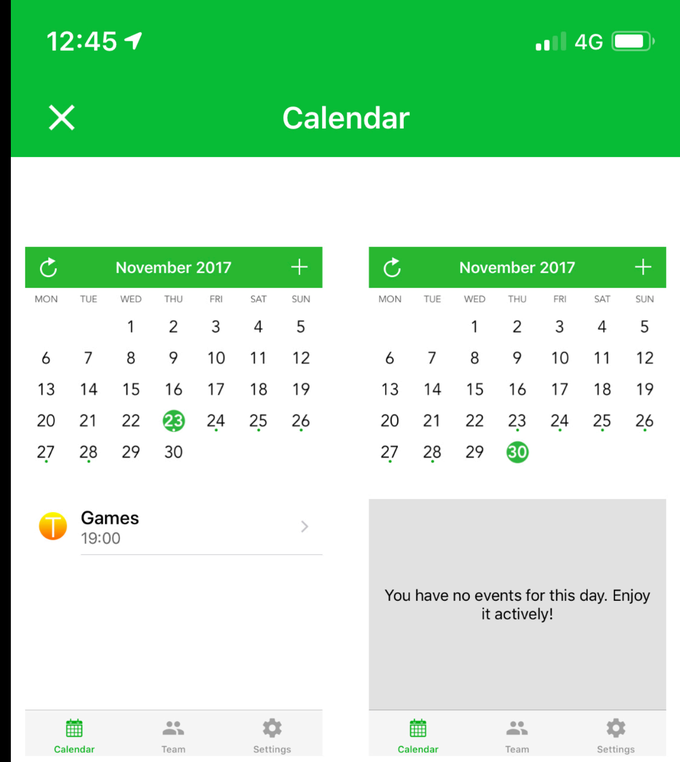
vs

UICollectionView

Container views

UIPageViewController

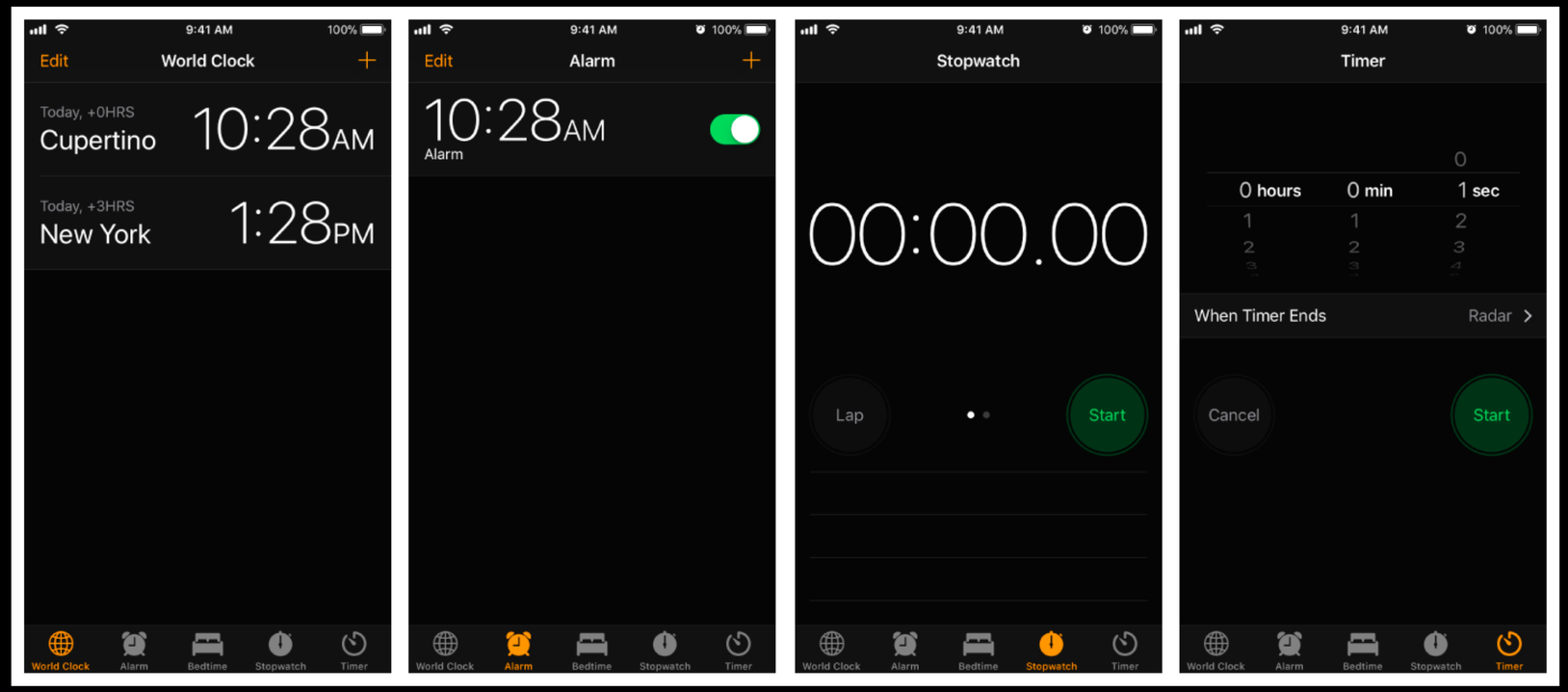
- A container view controller that manages navigation between pages of content, where each page is managed by a child view controller
- Dots at the bottom are optional
- Needs to have content controllers
- Animation can be like a book or just slide



If there is a dot under a day on the calendar, you have one or more events on that day. Clicking on that day will show you them in the list below the calendar. For more information about the event, click on it in the list.

Container views

UITabBarController



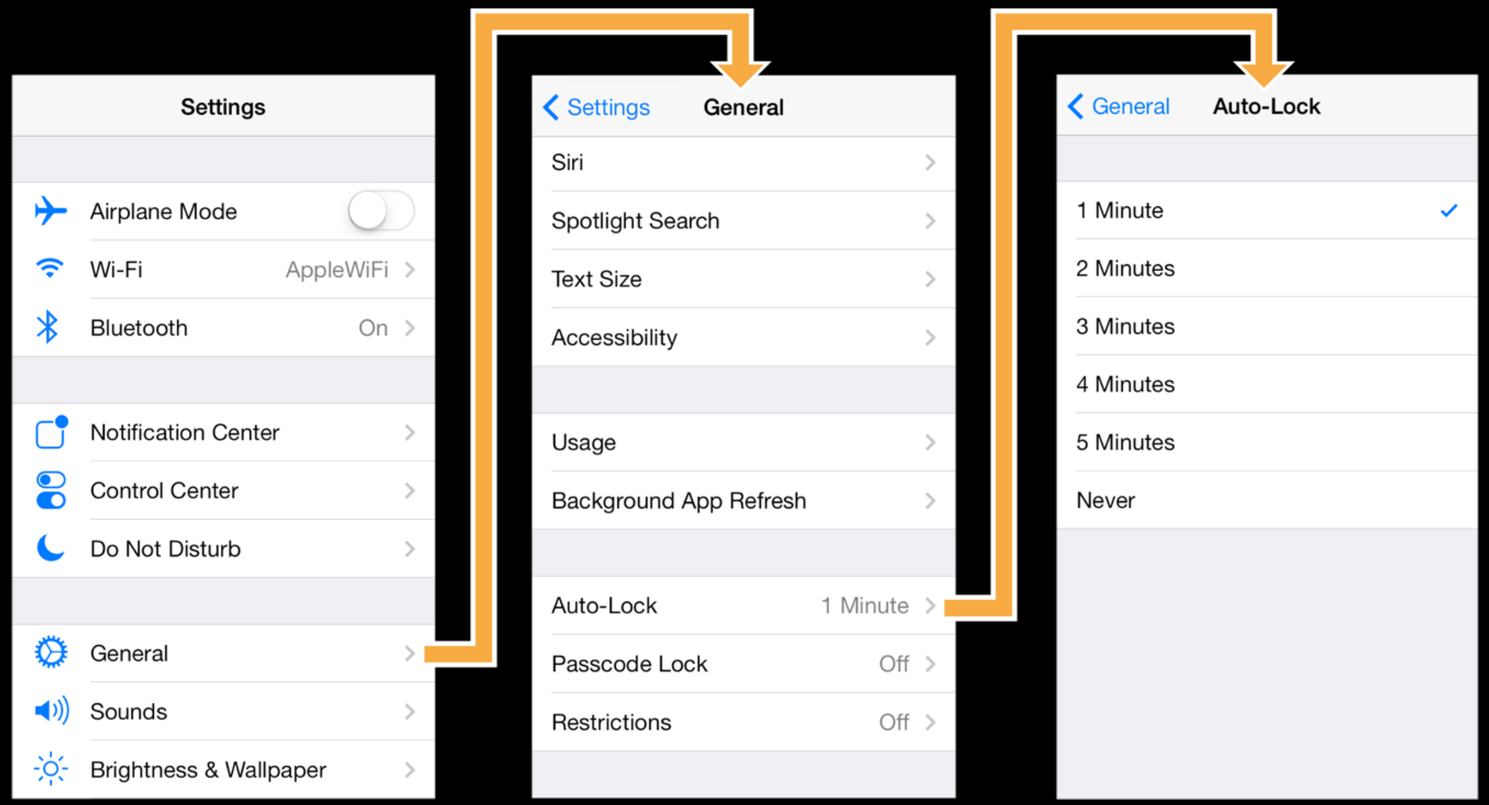
- A container view controller that manages a radio-style selection interface, where the selection determines which child view controller to display

- Maximum 5 tabs, otherwise it will automatically give 4 and a more tab with 3 dots, that will display the other tabs as a list

Container views

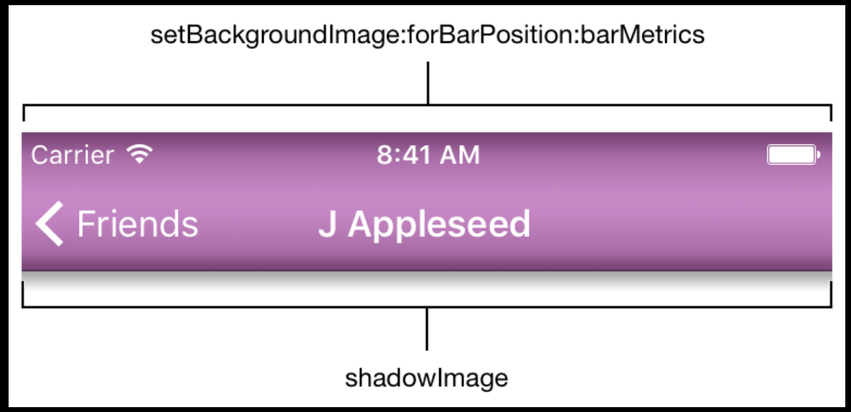
UINavigationController

- A container view controller that defines a stack-based scheme for navigating hierarchical content

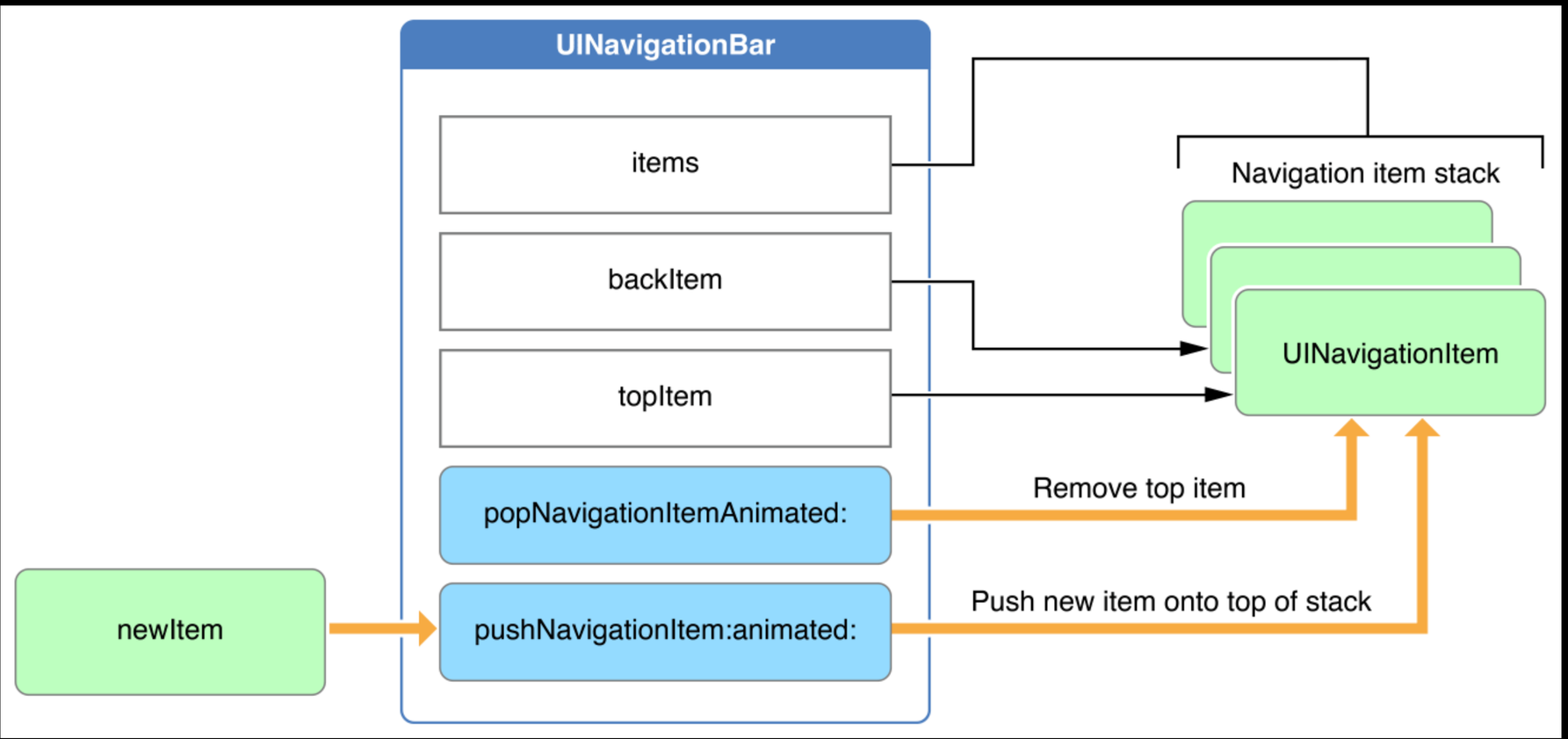


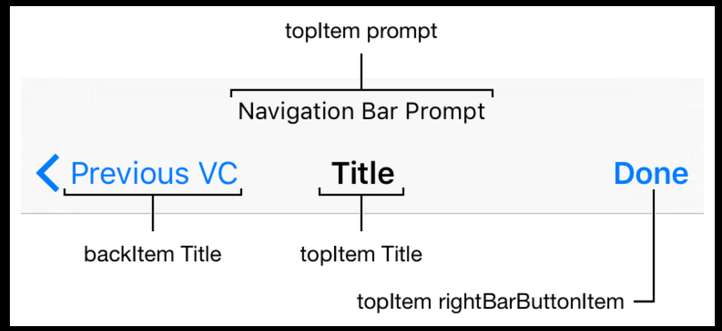
Bars

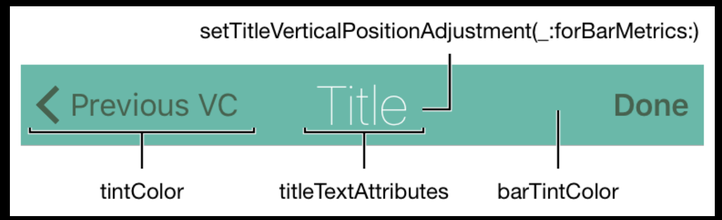
UINavigationBar



- Navigational controls displayed in a bar along the top of the screen, usually in conjunction with a navigation controller





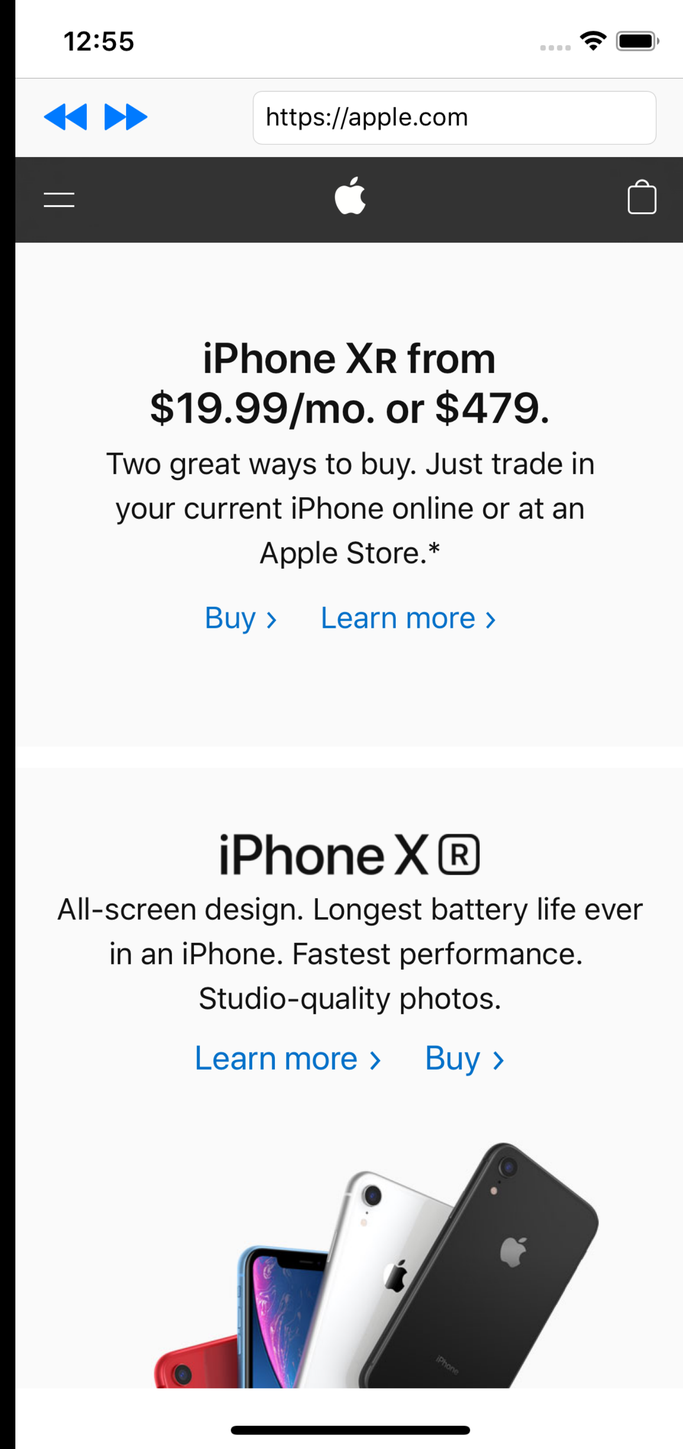


Bars

UIToolbar

- A control that displays one or more buttons along the bottom edge of your interface (usually), but can be placed anywhere

- Can be above the keyboard (with some action buttons like done, next, cancel, …)



Views

UIView

- An object that manages the content for a rectangular area on the screen.
- Most of UI elements are subclasses of UIView
- Main responsibilities:
  - Drawing and animation
  - Layout and subview management
  - Event handling

Views

UIView positioning - Frame vs Bounds

- You can get the position of a view with frame or bounds

- The **bounds** of an UIView is the rectangle, expressed as a location (x, y) and size (width, height) relative to its own coordinate system

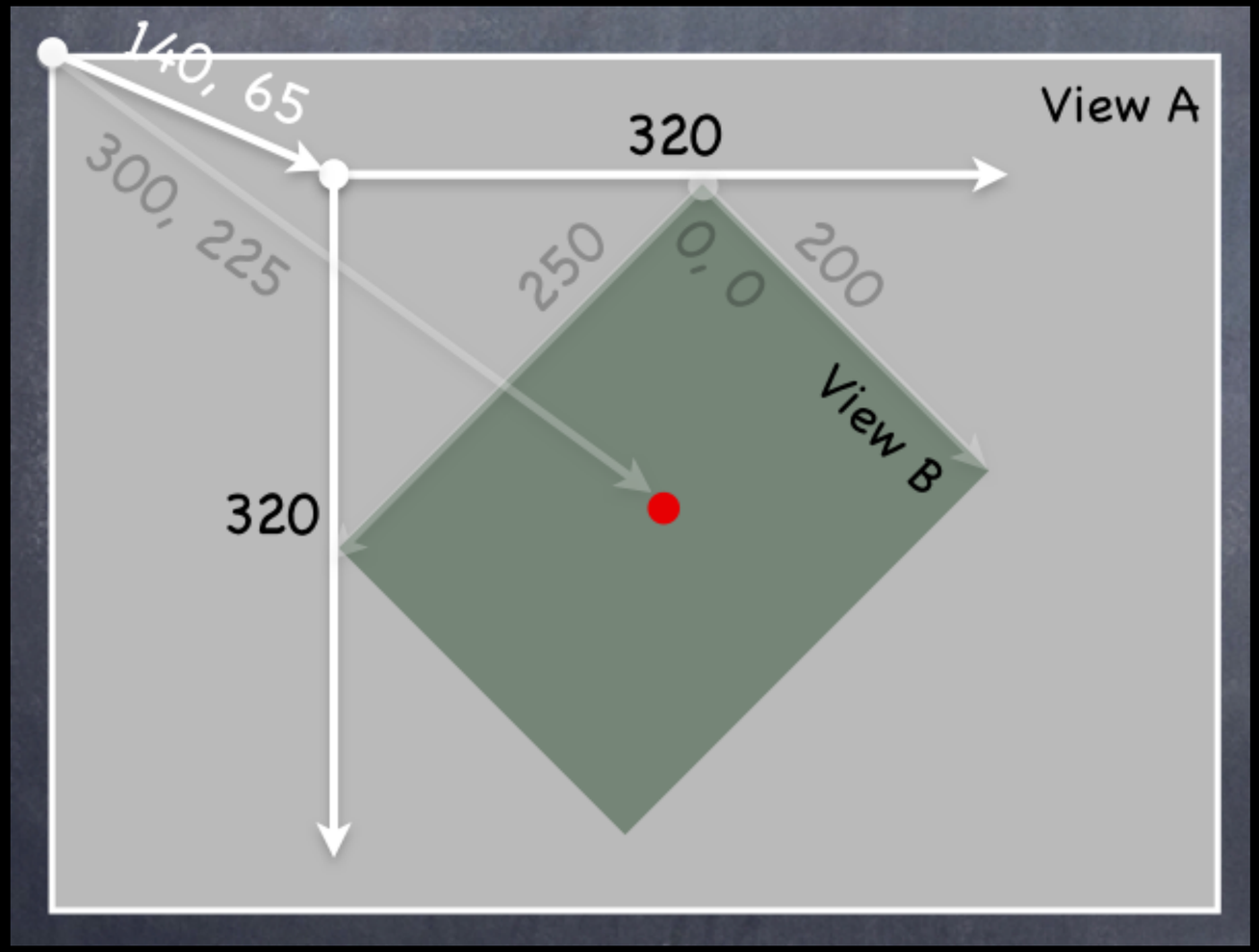
- The **frame** of an UIView is the rectangle, expressed as a location (x, y) and size (width, height) relative to the superview it is contained within

What are the bounds, frame and center of View B?

viewB.bounds = ((0, 0), (200, 250))

viewB.frame = ((140, 65), (320, 320))

viewB.center = (300, 225)



Text

UILabel

- A view that displays one or more lines of read-only text, often used in conjunction with controls to describe their intended purpose

Text

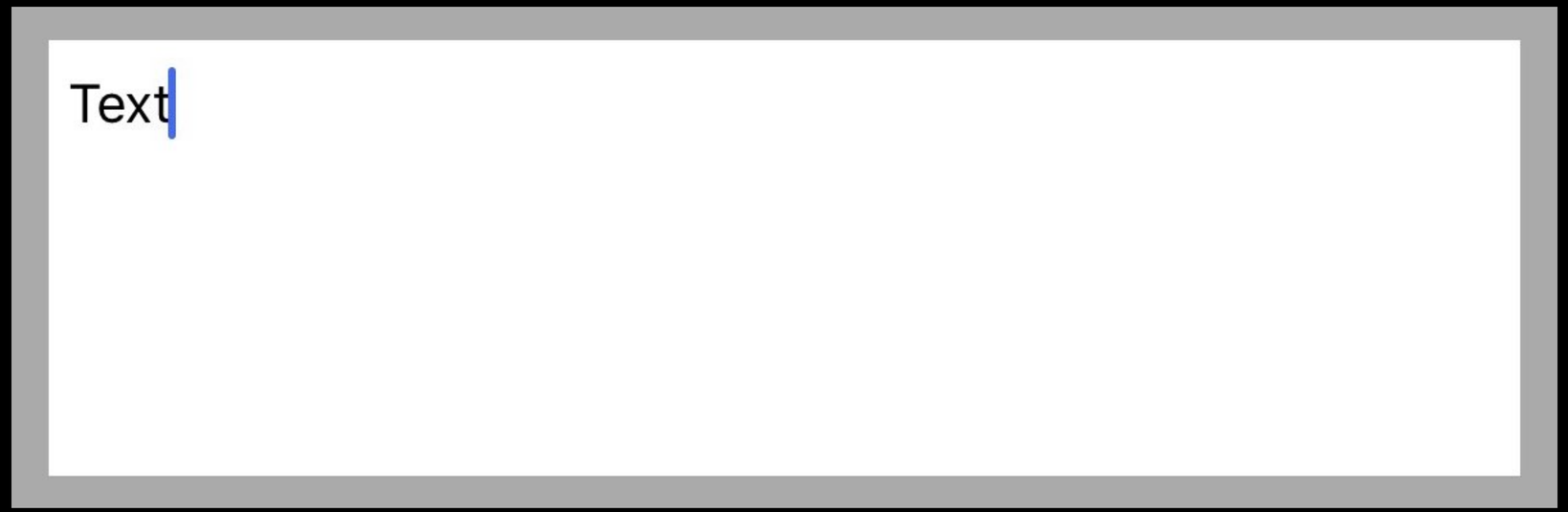
UITextField

- An object that displays an editable text area in your interface

Text

UITextView

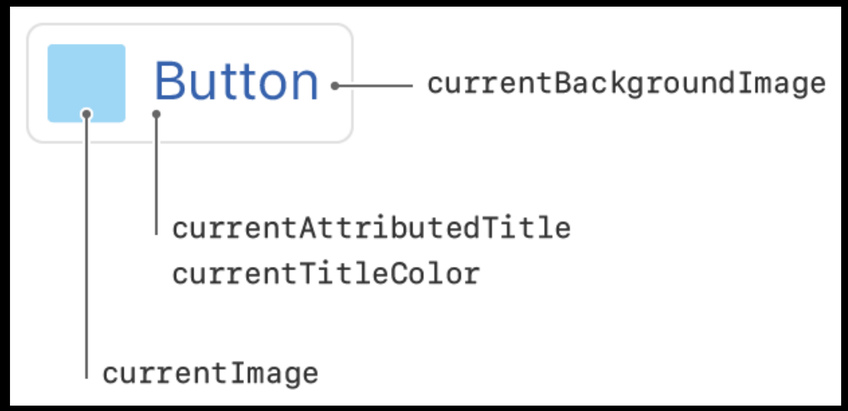
- A scrollable, multiline text region



Controls

UIButton

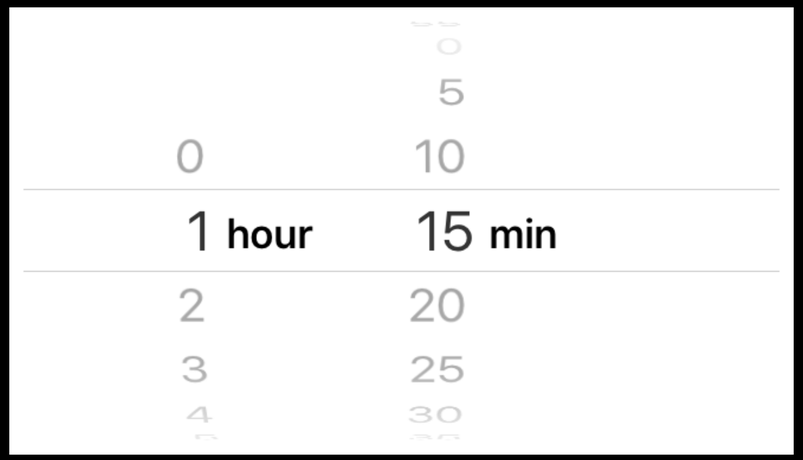
- A control that executes your custom code in response to user interactions
- Set type, title, image,…
- Handling events with @IBAction
- @IBAction func doSomething(sender: UIButton)

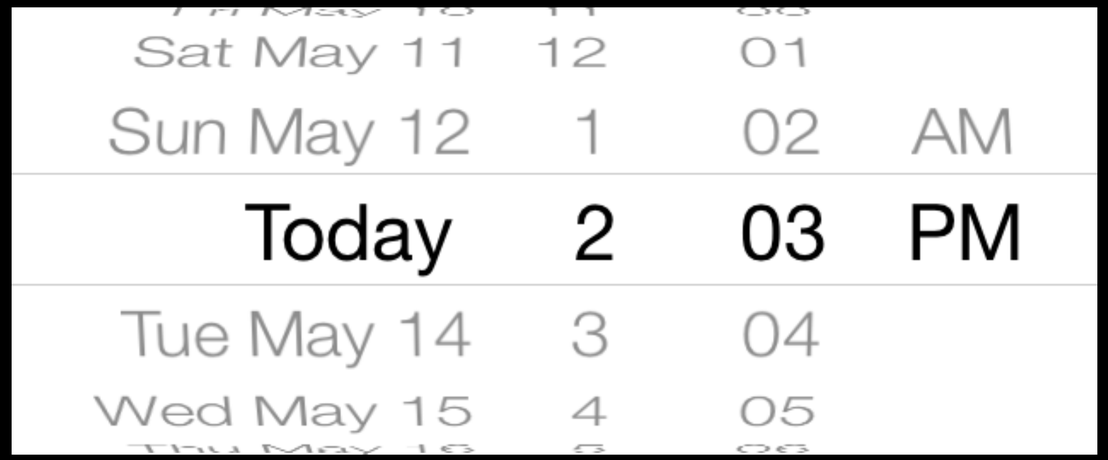


Controls

UIDatePicker

- A control used for the inputting of date and time values





Controls

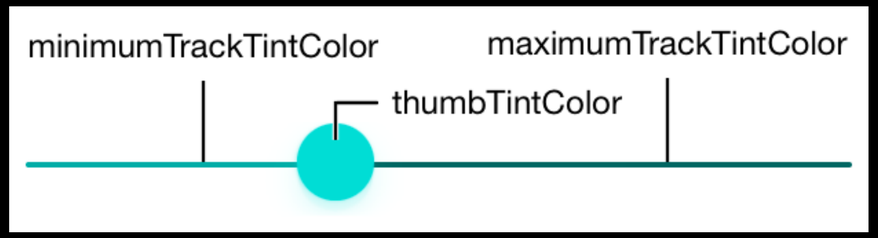
UISegmentedControl

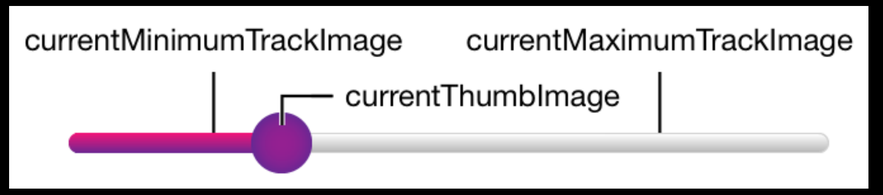
- A horizontal control made of multiple segments, each segment functioning as a discrete button

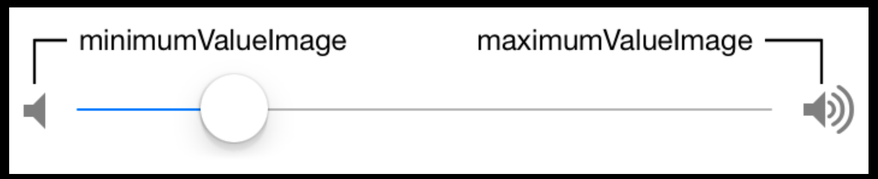
Controls

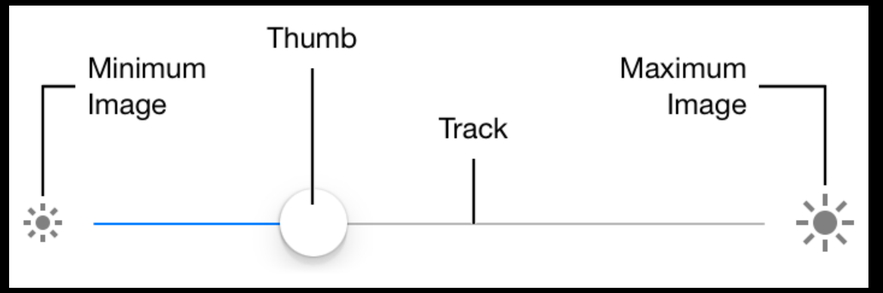
UISlider

- A control used for selecting a value from range









Controls

UIStepper

- A control used to increment or decrement a value

Controls

UISwitch

- A control that offers a binary choice, such as On/Off

Content views

UIActivityIndicatorView

- A view that shows that a task is in progress

Content views

UIImageView

- An object that displays a single image or a sequence of animated images in your interface

Content views

UIPickerView

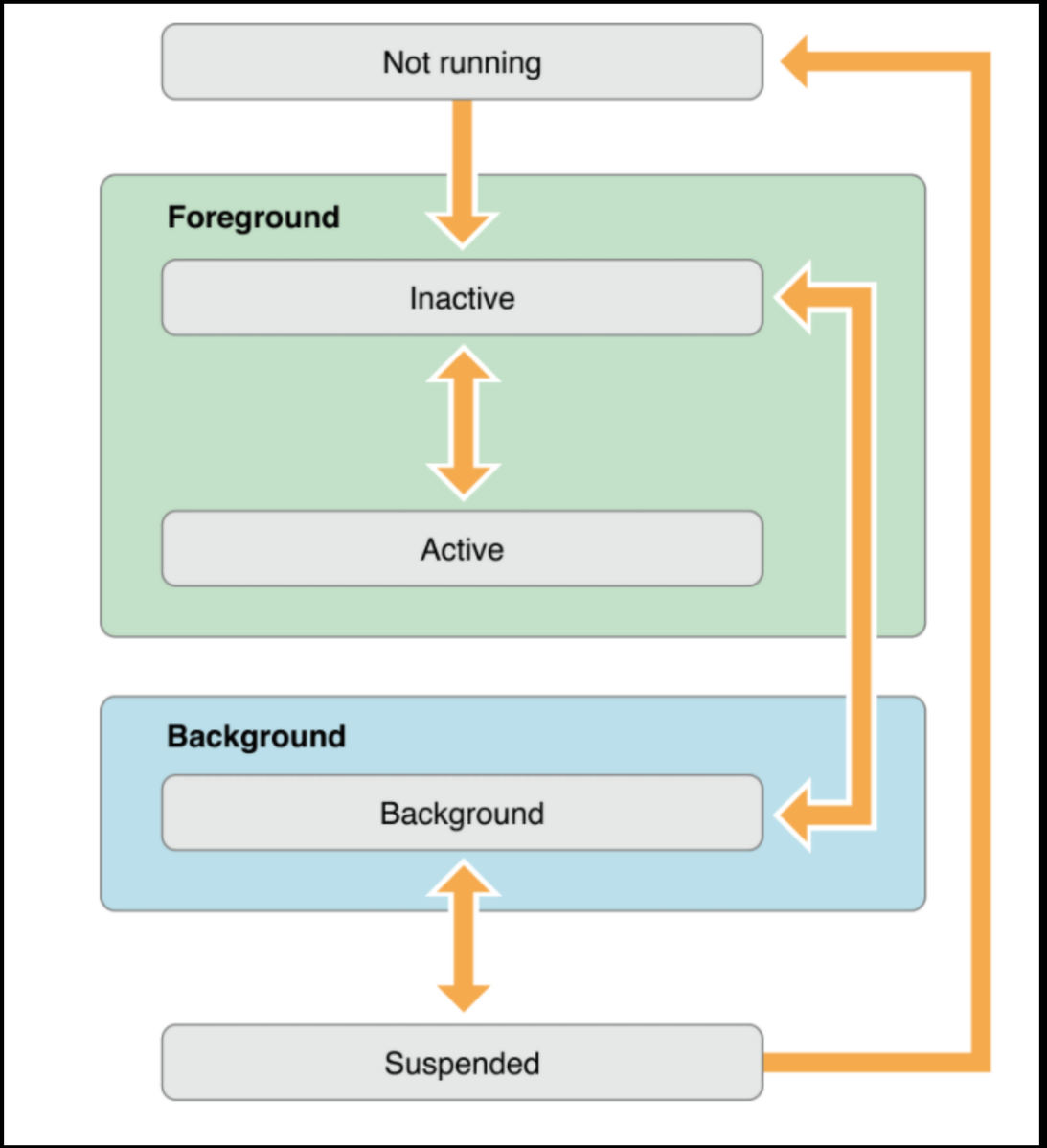
- A view that uses a spinning-wheel or slot-machine metaphor to show one or more sets of values

Content views

UIProgressView

- A view that depicts the progress of a task over time

Workshop 2 - Part 4:
App lifecycle



application:willFinishLaunchingWithOptions:

application:didFinishLaunchingWithOptions:

applicationDidBecomeActive:

applicationDidEnterBackground:

applicationWillEnterForeground:

applicationWillTerminate:

…

Workshop 2 - Part 5: Positioning UI Elements

Positioning UI Elements

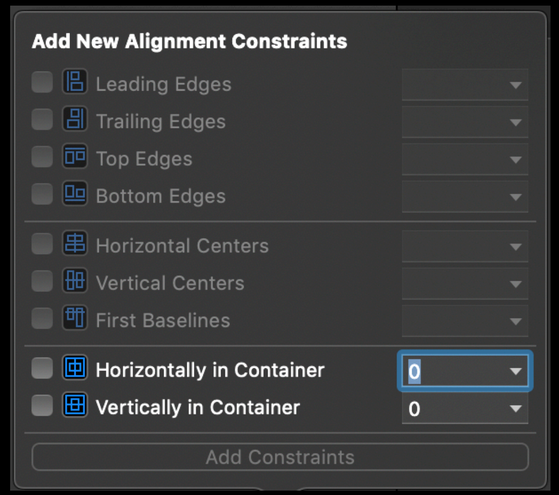
Types

- Constraints
- Embedding

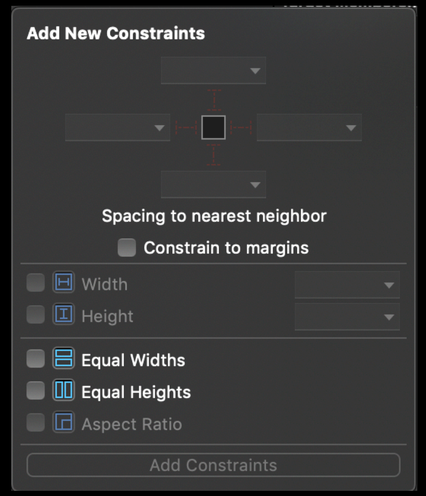
Positioning UI Elements

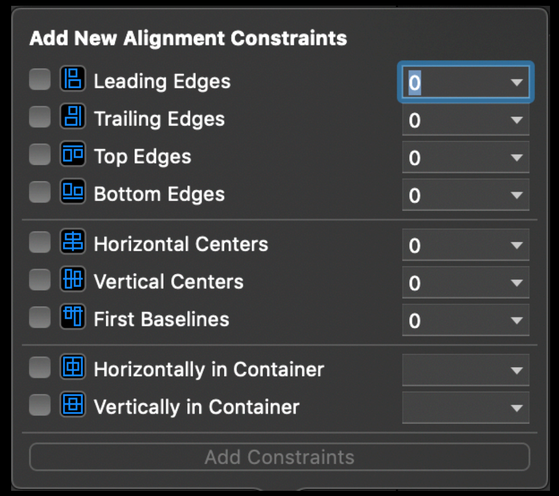
Constraints

- Single element constraints



- Multiple element constraints

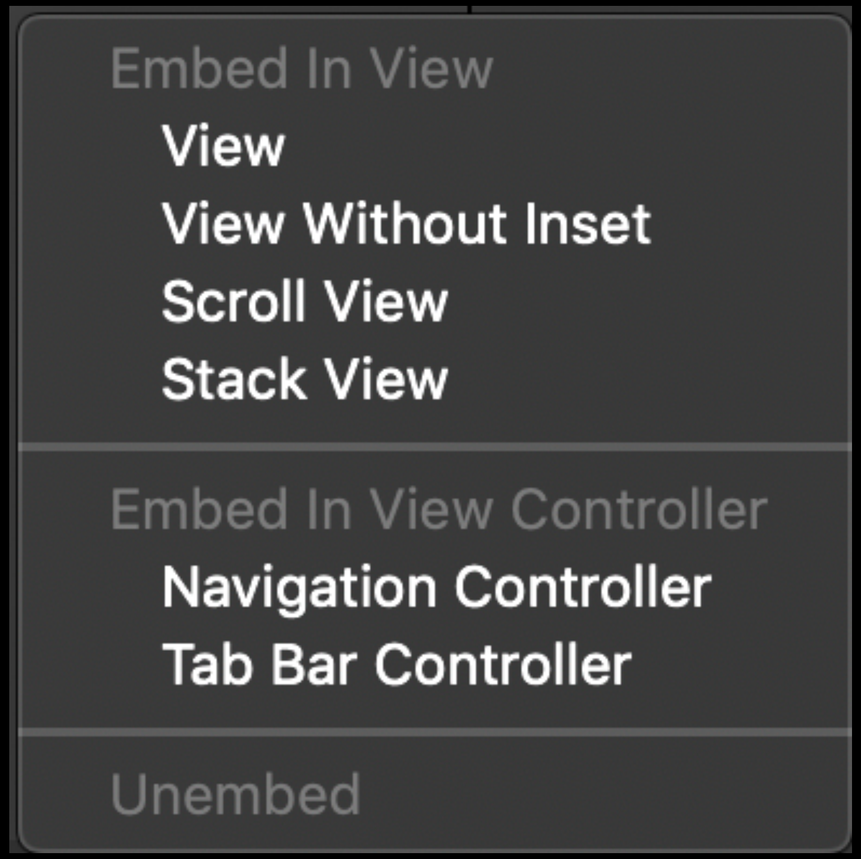




Positioning UI Elements

Embedding

- Embed in View
- Embed in View Controller



Workshop 2 - Part 6: Connecting UI to code

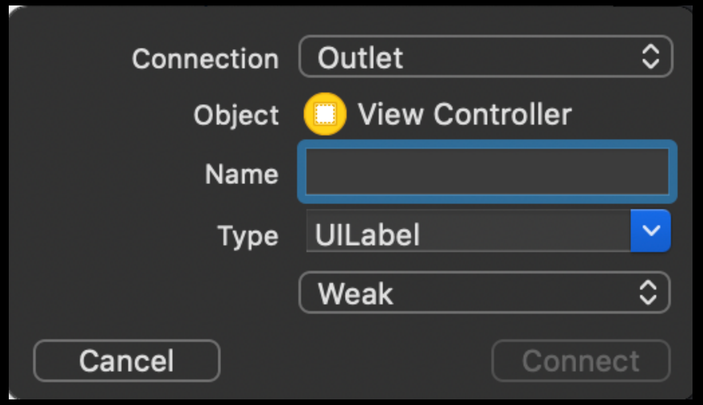
Connecting UI to code

Types

- IBOutlet
- IBAction

Connecting UI to code

IBOutlet



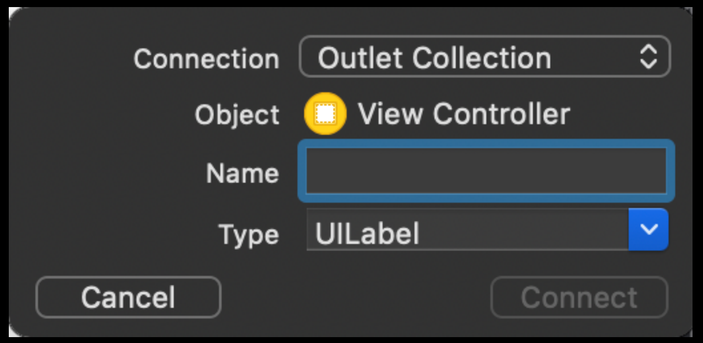
- IBOutlet
- IBOutletCollection

- A property accessible inside the corresponding View

- Can apply custom edits in code

- Done to most of the UI Elements

```
@IBOutlet weak var timeLabel: UILabel!
```



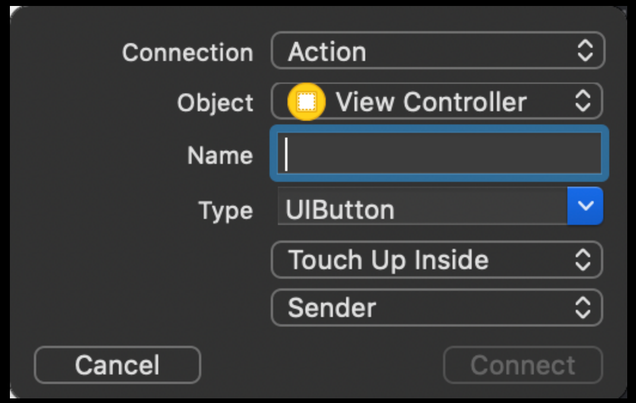
Connecting UI to code

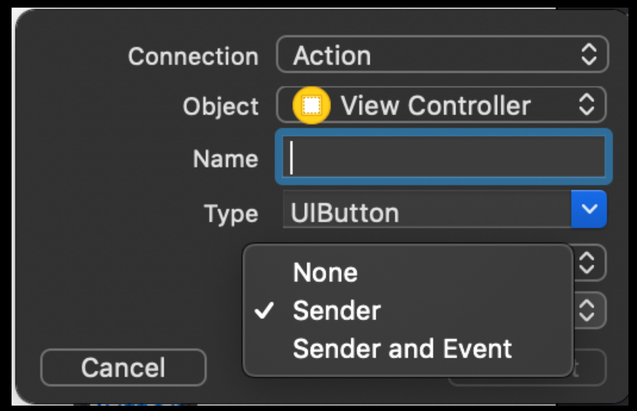
IBAction

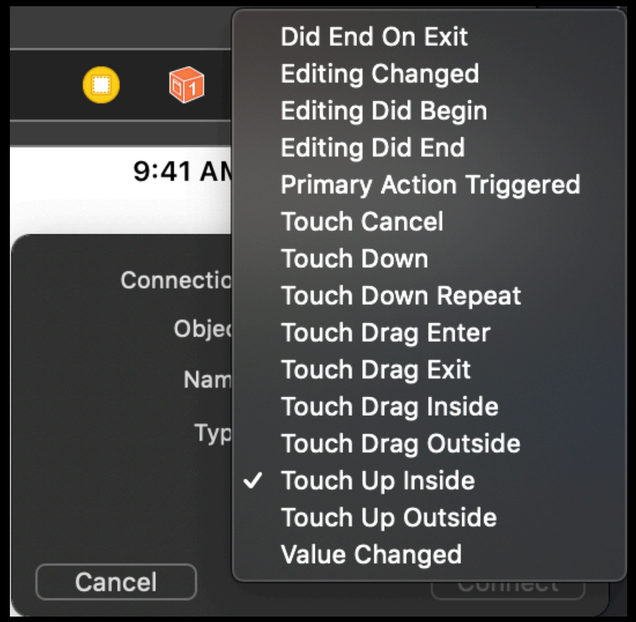
- Can be done just to some elements
- Action that is triggered when specified event occurs

```
@IBAction func showTime(_ sender: UIButton) {

}
```







Workshop 2 - Part 7:
Navigating between the screens

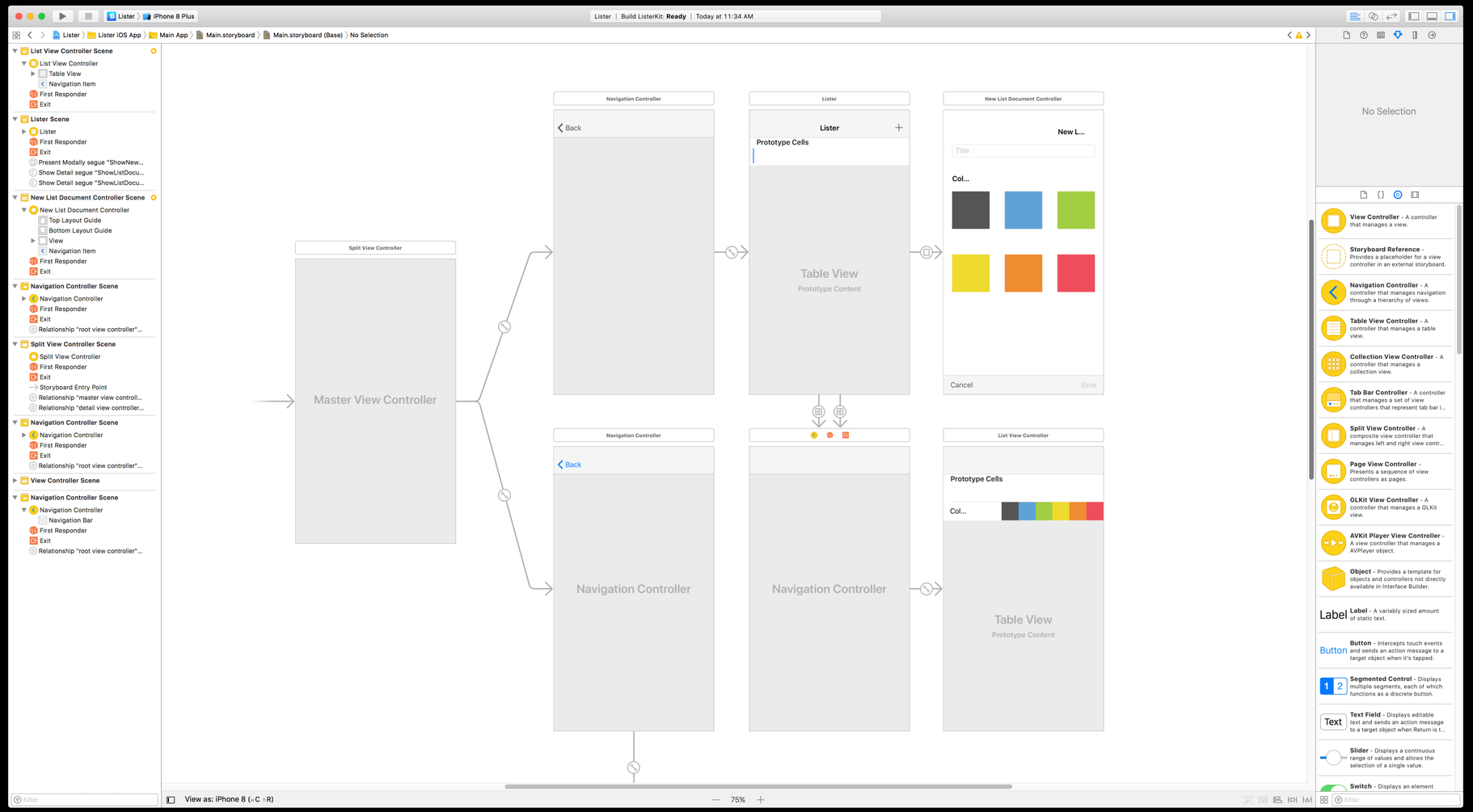
Navigating between the screens

Options

- Segues in IB
- Code

Navigating between the screens

Segues in IB



Navigating between the screens

Segues in IB

- Control-drag from source to destination
- Give identifier, if edits in code needed
- Handle the segue inside :
  `override func prepare(for segue: UIStoryboardSegue, sender: Any?)`
- Segue is initiated automatically (if connected to touchable object) or manually in code by:
  `performSegue(withIdentifier: "ShowDetailSegue", sender: nil)`

Navigating between the screens

Code

- Segues are nice for small and easy apps, but can become obsolete in just a bit larger ones

- Branching in code much easier than in Segues (you control every case)

- Almost always the preferred way of navigation

```
    let vc = storyboard?.instantiateViewController(withIdentifier:
String(describing: DetailViewController.self)) as! DetailViewController
    vc.image = images[indexPath.row]
    navigationController?.pushViewController(vc, animated: true)
```

Navigating between the screens

Types of navigation

- Push
  - Hierarchy of views inside UINavigationController (pushing and popping views in the stack)

- Present
  - New flow of the app starts
  - Showing popover

- Custom
  - If custom animated transitions are needed