

# iOS & OS X



Operating System	OS X	iOS
Device	Mac	iPhone, iPad, iPod, Apple Watch
Framework	Cocoa	Cocoa Touch

- June 2007 - First iPhone Launch
- Early 2008 – Native Development Kit released
- July 2008 – Apple App Store available to users
  - 500 third-party apps
  - Over ten million titles downloaded first week

# Why Objective-C for iOS



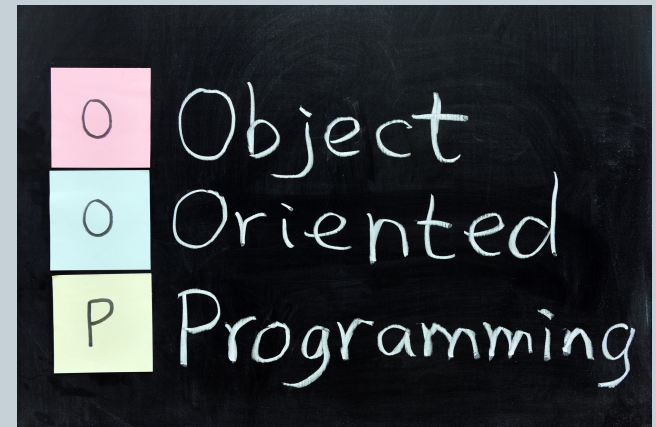
- Steve Jobs started NeXT when he left Apple in 1985
- NeXT initially made computers & developed software
- In 1993 NeXT became a software company
- NeXT had developed the NeXTSTEP operating system
  - Objective-C was selected as the main language by NeXT
    - ÷ for its NeXTSTEP operating system
- In 90s Apple was looking for a replacement for Mac OS
- In 1996 Apple purchased NeXT for \$429 million
- NeXTSTEP OS was used as the basis for Mac OS
- Purchase of NeXT brought Steve Jobs back to Apple

# Native Mobile Apps



- Built specifically for a particular OS

- iOS apps
  - ÷ Written in Objective-C & Swift
- Android apps
  - ÷ Written in Java



- What do Objective-C, Swift and Java have in common?
  - They are all ***Object Oriented Programming Languages***

# Main Object Oriented Programming Principles

- **Class**

- Template or blueprint

- **Object**

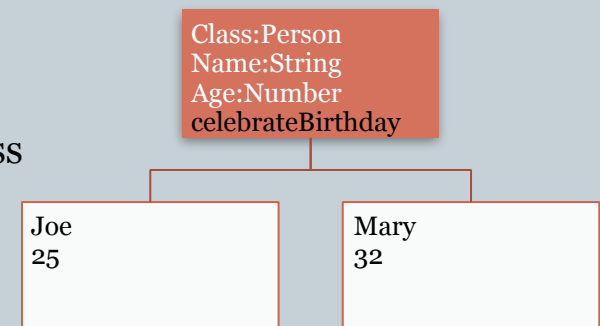
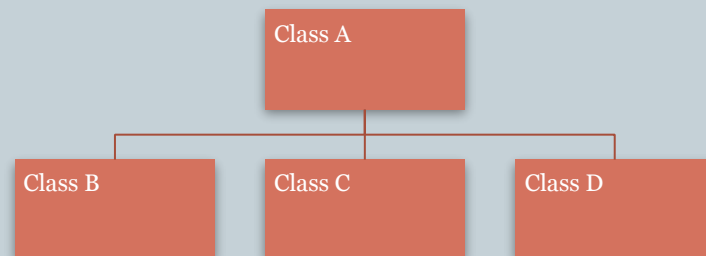
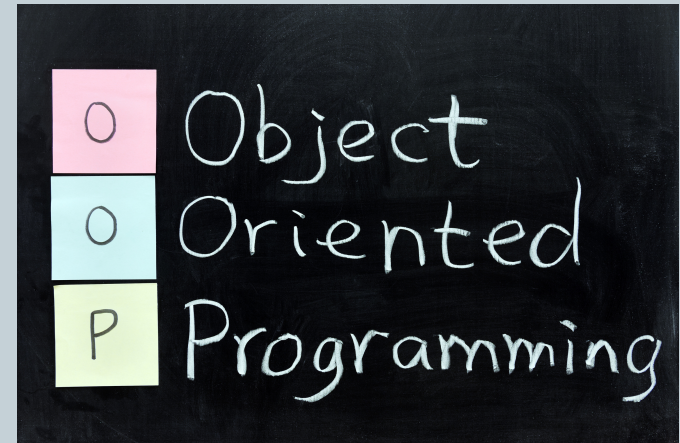
- Instance of a class
  - ÷ Properties = Attributes
  - ÷ Methods = Actions = Functions = Events

- **Message Passing**

- Objects communicate through messages

- **Inheritance**

- The child inherits all property and methods of the parent class





# Development Requirements: Xcode

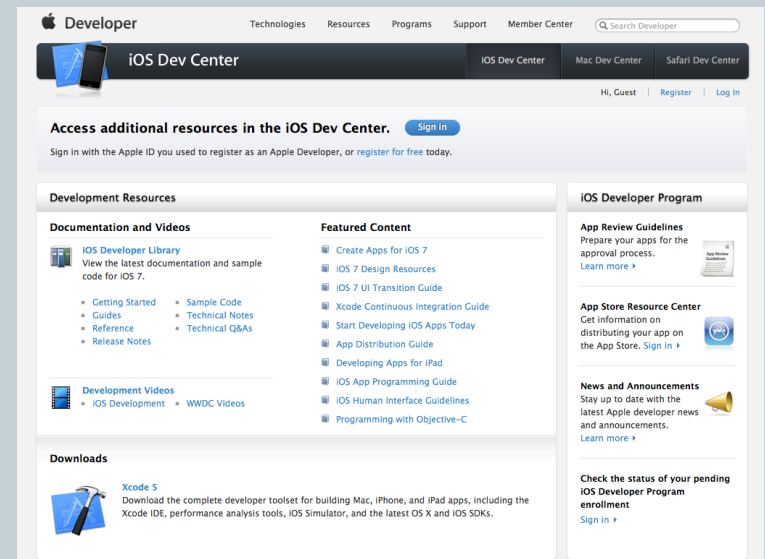


- System: Have an Intel-based Mac
- Download Xcode from the Mac App Store
  - Xcode is an integrated development environment (IDE)
- Xcode provides all of the tools you need to create and manage your iOS projects and source files
- Xcode includes:
  - iOS Simulator
  - latest version of the iOS SDK
    - ÷ SDK (Software Development Kit)



# Development Requirements: iOS Dev Center

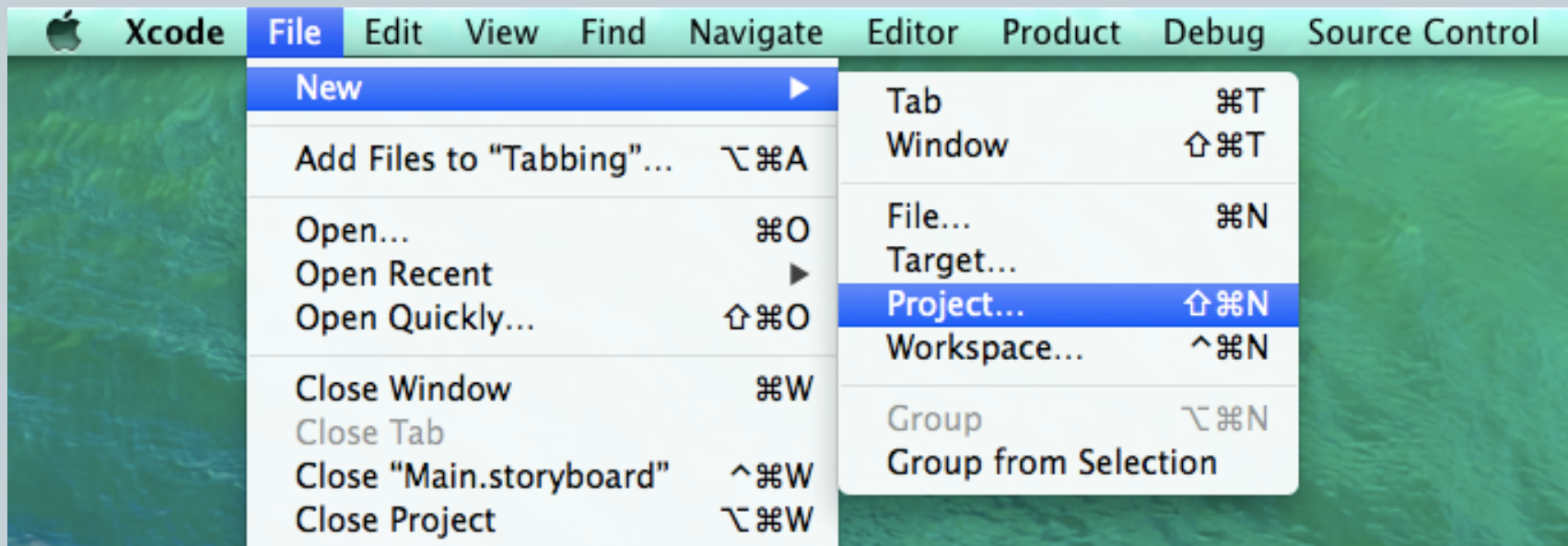
- Register as an Apple Developer on iOS Dev Centre
  - <https://developer.apple.com/devcenter/ios/index.action>
- iOS Dev Centre gives access to resources:
  - The latest version of the SDK
  - Pre-release documentation
  - Videos
  - Sample code



# App Creation



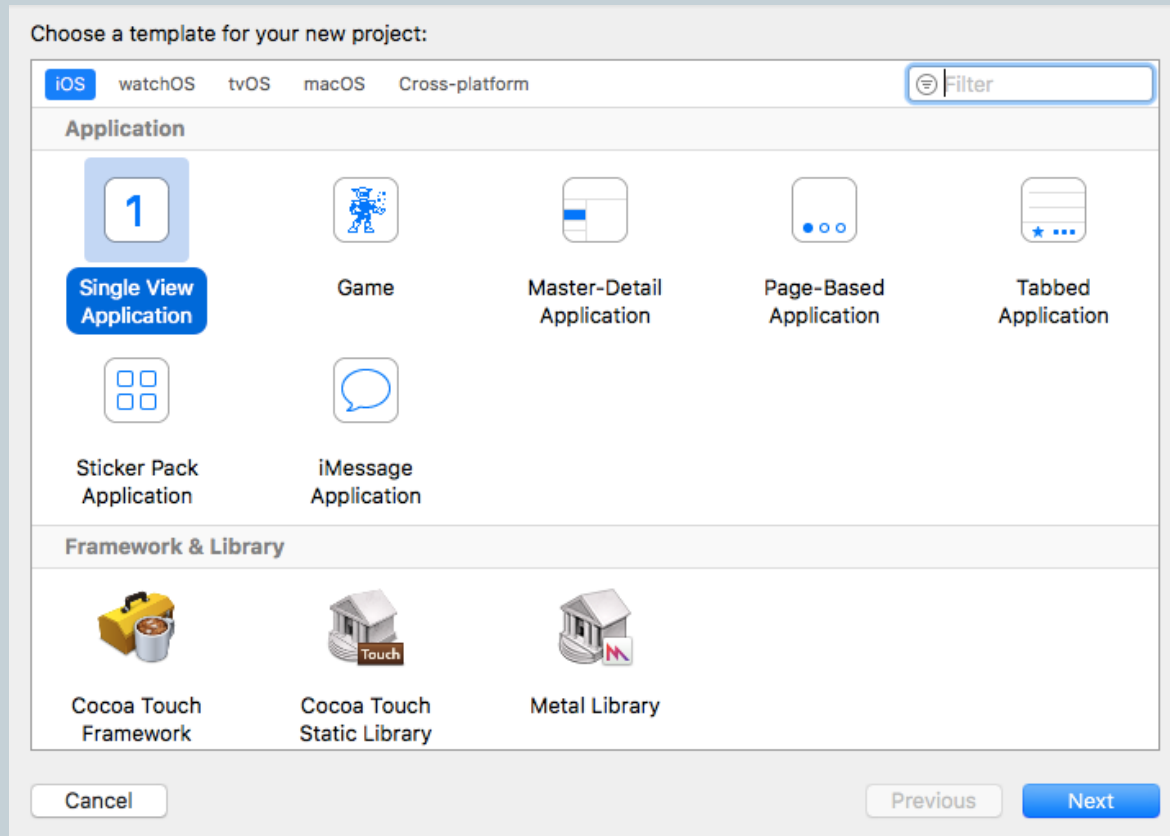
- Launch Xcode
- File → New → Project



# App Creation



- iOS → Application → Single View Application



# App Creation



- Product Name is the name of your app

Choose options for your new project:

Product Name:

Team:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

Devices:

☒ Use Core Data  
☐ Include Unit Tests  
☐ Include UI Tests

# App Creation

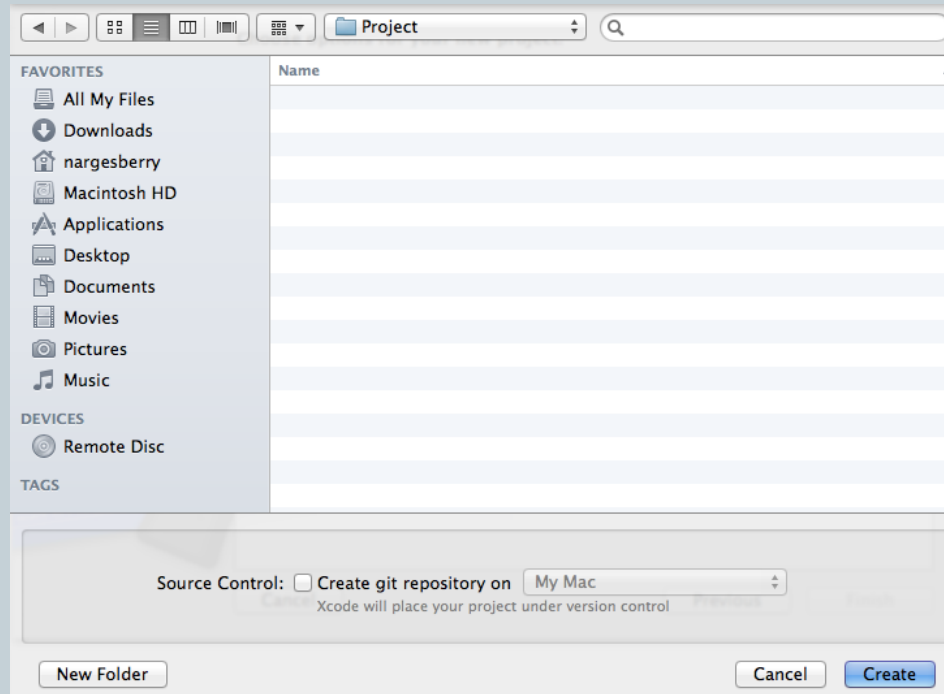


- **Product Name**
  - The name of your app
- **Organization Name**
  - The name given here will appear in the copyright section of the comments
- **Organization Identifier**
  - This should be unique identifier unique to you
- **Bundle Identifier = “Product Name” + “Company Identifier”**
  - Unique to your app
- **Class Prefix**
  - Swift
  - Objective-C
- **Devices**
  - iPhone
  - iPad
  - Universal

# Select Directory



- Once the fields have been filled
  - Click Next
    - ÷ You will be prompted to select a directory to save the project into



# Project Created



The screenshot displays the Xcode IDE interface for a newly created project named "ProjectOne". The top status bar indicates the project is "Ready" and the time is "Today at 10:50". The left sidebar shows the project's file structure, including source files like AppDelegate.swift and ViewController.swift, as well as assets and storyboards. The main editor area is divided into two panes: the left pane shows the "PROJECT" and "TARGETS" sections, with "ProjectOne" selected under "TARGETS"; the right pane shows the "General" tab of the project settings. The "General" tab is further divided into sections: "Identity", "Signing", and "Deployment Info". The "Identity" section shows the Display Name as "ProjectOne", Bundle Identifier as "com.bermotech.ProjectOne", Version as "1.0", and Build as "1". The "Signing" section shows that "Automatically manage signing" is checked, and the Team is "BERMOTECH LIMITED". The "Deployment Info" section shows the Deployment Target as "10.1", Devices as "iPhone", Main Interface as "Main", Device Orientation as "Portrait" (checked), "Landscape Left" (checked), and "Landscape Right" (checked), and Status Bar Style as "Default". The right sidebar shows a "Quick Help" section with "No Quick Help" and a "Search Documentation" button. At the bottom of the right sidebar, there is a "No Matches" message.

ProjectOne: Ready | Today at 10:50

ProjectOne

PROJECT

ProjectOne

TARGETS

ProjectOne

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

▼ Identity

Display Name ProjectOne

Bundle Identifier com.bermotech.ProjectOne

Version 1.0

Build 1

▼ Signing

☒ Automatically manage signing  
Xcode will create and update profiles, app IDs, and certificates.

Team BERMOTECH LIMITED

Provisioning Profile Xcode Managed Profile ⓘ

Signing Certificate iPhone Developer: Narges Berry Noubar (6C94W...

▼ Deployment Info

Deployment Target 10.1

Devices iPhone

Main Interface Main

Device Orientation ☒ Portrait  
☐ Upside Down  
☒ Landscape Left  
☒ Landscape Right

Status Bar Style Default

☐ Hide status bar  
☐ Requires full screen

Quick Help

No Quick Help

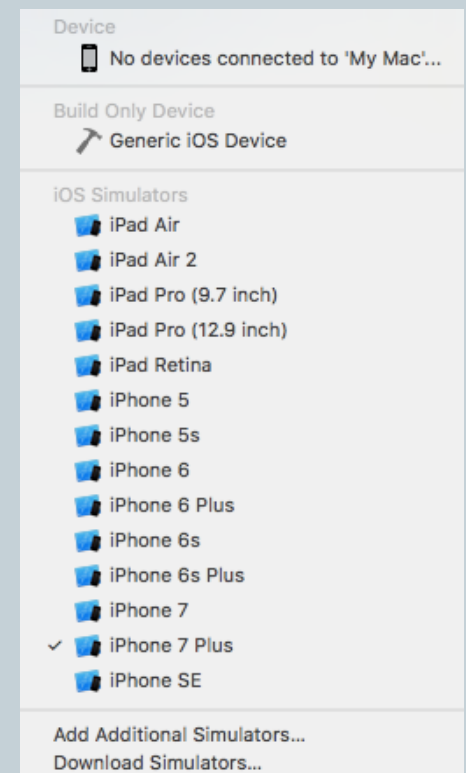
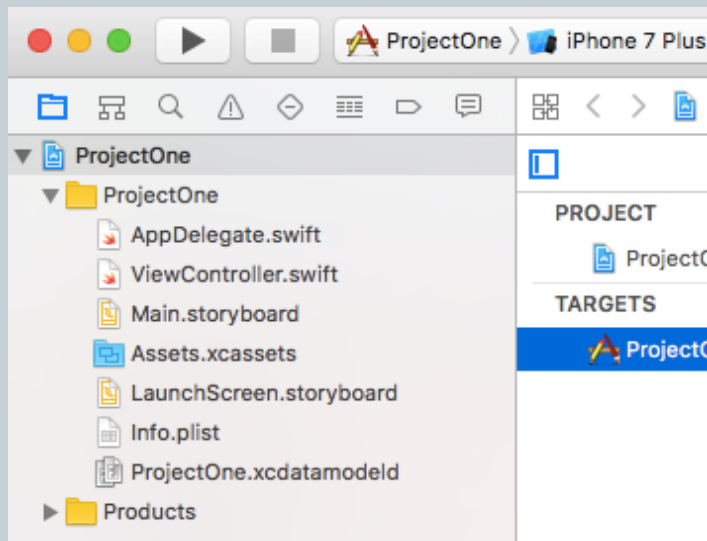
Search Documentation

No Matches



# Simulator for Testing

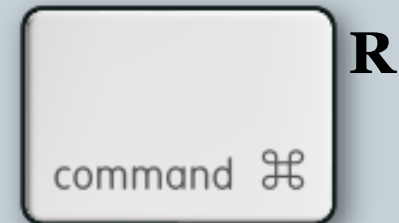
- Choose iOS Simulator from the drop-down list



# Compile and Run



- **Program**: *Set of instructions*
- **Compile**: *Translate a program*
  - ÷ From human language
  - ÷ To computer language
- **Run**: *Execution of the translated instructions by the computer*
- There are 2 ways to compile and run the program
  - Click on the “Run” button at the top left corner
  - Press and hold “command” button, then press “R”
- Run the Project
  - Once the project is run, the simulator will appear
  - At this point it will be blank



# File & Folders in the Navigator Area

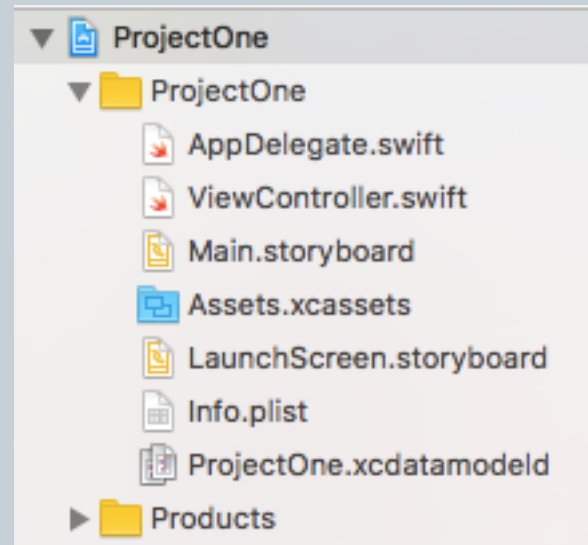


- **Main.storyboard**

- Allows you to design entire user interface in 1 place
  - ÷ Shows app screens and the connection between them

- **AppDelegate.swift**

- **ViewController.swift**



# Comments and Import Statements



- **Comments**

- Comments are ignored by the compiler
- **// Single Line Comment**
- **/\* Multiline Comment \*/**

- **import statements**

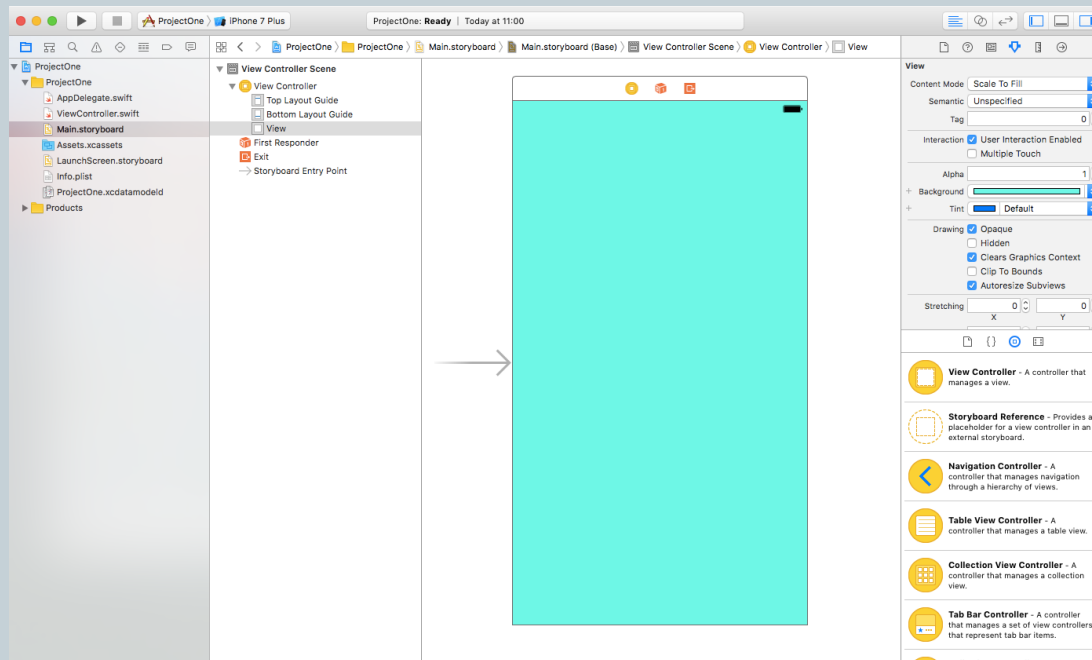
- Brings in the definition in the file
  - ÷ The import statement precompiles all the classes once
    - Example

```
import UIKit
```

# What is a storyboard?



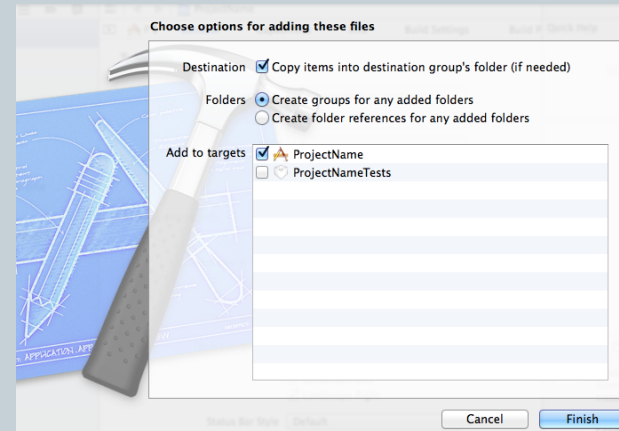
- Visual representation of the
  - *User interface* of an iOS application
- Shows **screens** and **connection** between them
- Each screen is a **ViewController**



# Adding files to the “Supporting Files” folder

- Drag and Drop

- Put the image or image folder on the desktop
- Drag it to the project folder



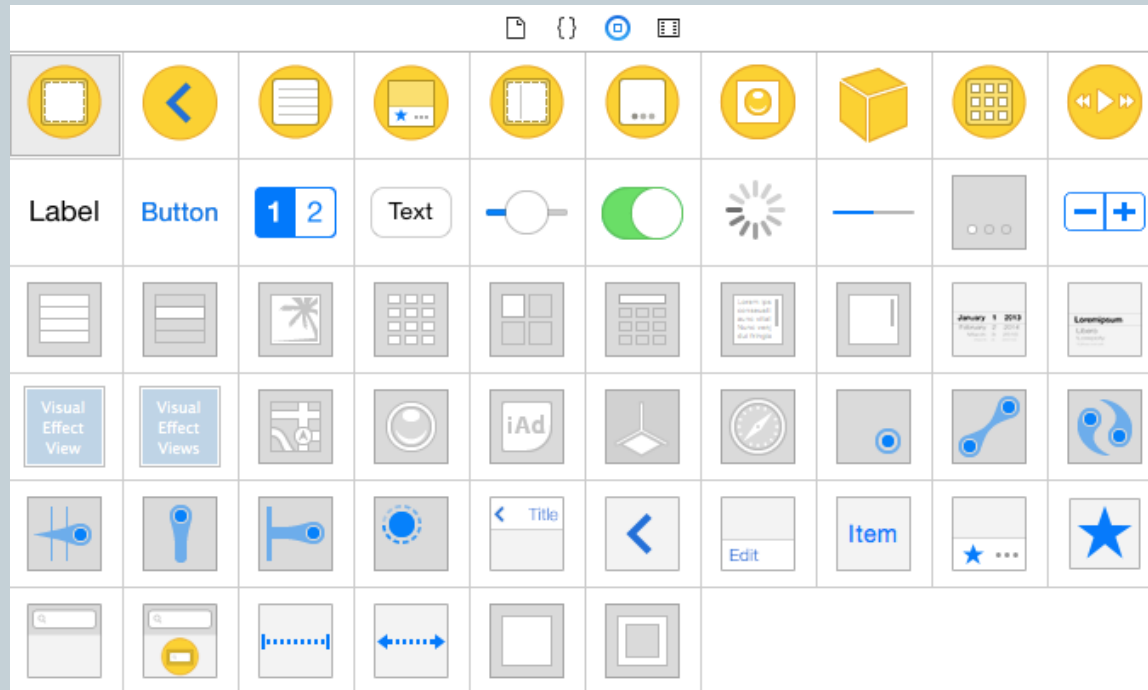
- Make sure **Destination** is **checked**
  - This will add the images to the project
  - Otherwise they will be referenced

Destination ☒ Copy items into destination group's folder (if needed)

# Access Xcode Objects



- To access Xcode Objects → Click on Object Library
  - at the bottom right corner
  - ÷ Xcode Objects will be displayed



# alpha



- Every view has a property called alpha
- **What is alpha and what do its values mean?**
- Alpha defines the **transparency** of view objects.

1 is fully opaque.

0 is fully transparent



# Custom Colour



Example:

```
UIColor(red: 178/255, green: 178/255, blue: 122/255, alpha: 1)
```



```
@IBAction func randomButton(_ sender: Any)
{
    let r = arc4random()%3

    if(r==0)
    {
        self.view.backgroundColor = UIColor.purple
    }

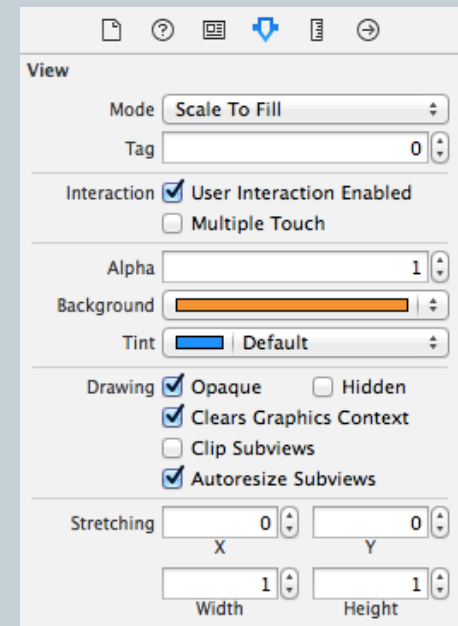
    if(r==1)
    {
        self.view.backgroundColor = UIColor.green
    }
    if(r==2)
    {
        self.view.backgroundColor = UIColor(red: 252/255, green: 178/255, blue: 178/255, alpha: 1 )
    }
}
```

# Xcode Objects

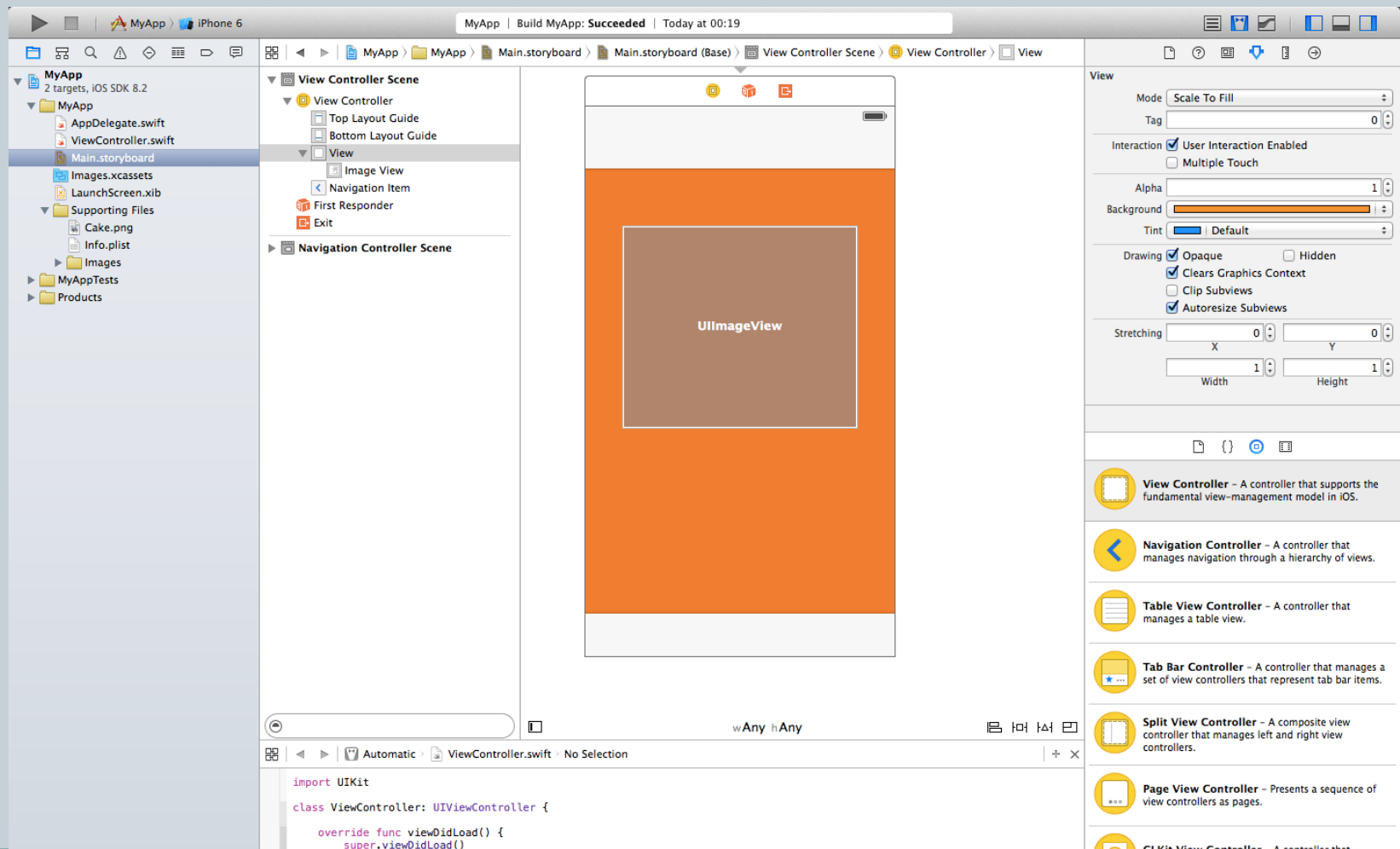
## Image View & Label



- Create a new “Single View Application” project
- Click on Main.Storyboard
- Click on view on the storyboard
- → Attributes Inspector
- Change view Background colour

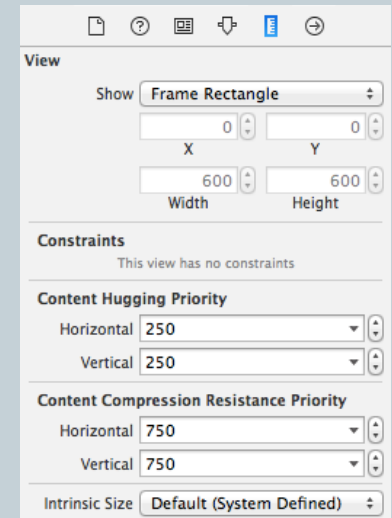
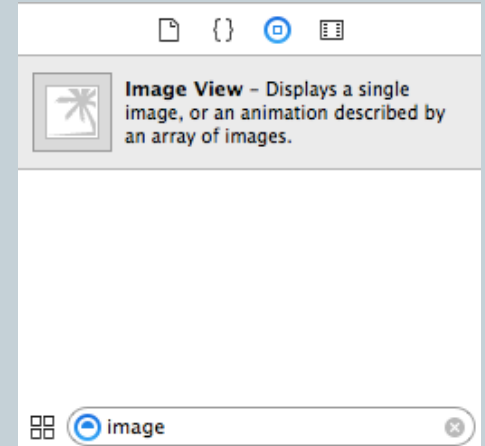


# Add an Image to the Storyboard



# Add an Image to the Storyboard

- Click on Main.storyboard
- Click on the Objects Library
- Drag an “Image View” to the storyboard
- Change the width and height of the image
  - Click on the Image View on the storyboard
    - ✎ → Click on the Size Inspector (ruler)
      - Change Width to 200
      - Change Height to 200



# Adding an Image



- Add an image to the “Image View”

1. Add an image to your project

2. Click on the “Image View”

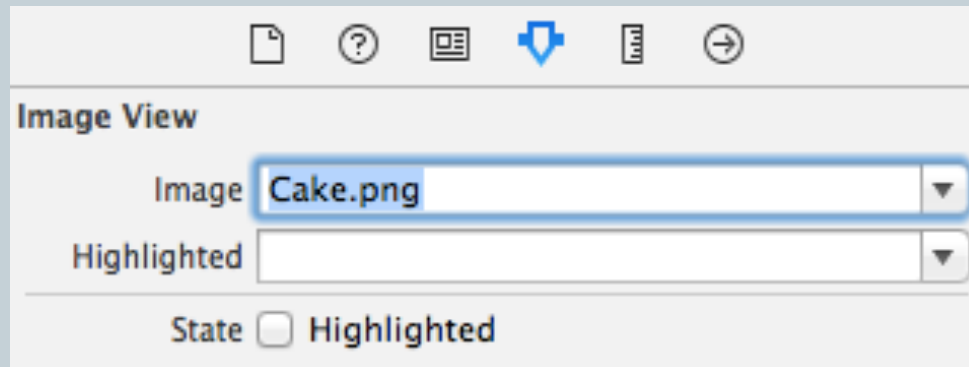


→ Attribute inspector



→ Image View

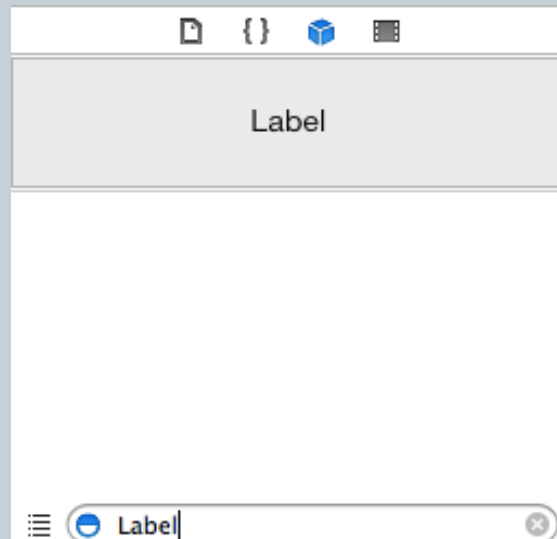
- → Select Image



# Add a Label to the Storyboard



- Click on Main.storyboard
- Click on the Objects Library
- Drag a “Label” to the storyboard
- Double click on the label and change its text



# Customise Label Font

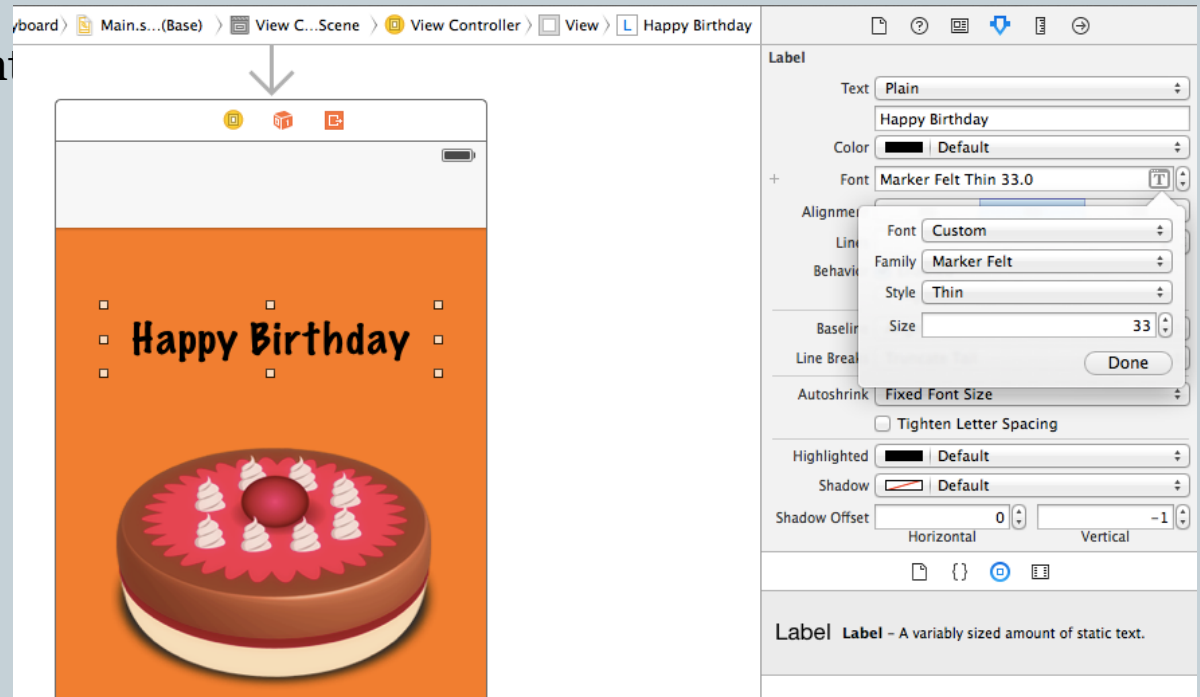


- Click on the label

- → Attribute Inspector
- → Label
- → Click on T
- → Customise label font



- Run the App!





# Useful Code Snippets



- Change the background colour  
`self.view.backgroundColor = UIColor.orange`
- Set Image of an Image View  
`myImage.image = UIImage(named: "image.png")`
- Changing the text of a label  
`myLabel.text = "New Text";`
- Making an object invisible  
`myObject.alpha = 0`
- Animate button to fade  
`UIView.animateWithDuration(10.0, animations:  
{self.fadeButton.alpha = 0})`
- Change the colour of label  
`myLabel.textColor = UIColor.red`

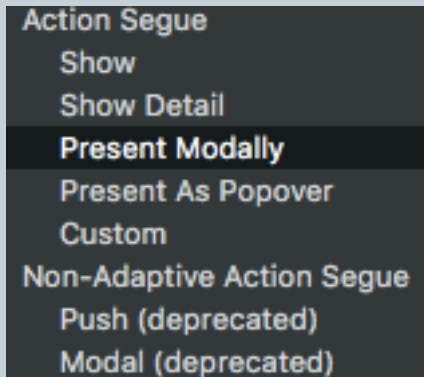
# Navigation Action Segue (modal)



# Present Modally



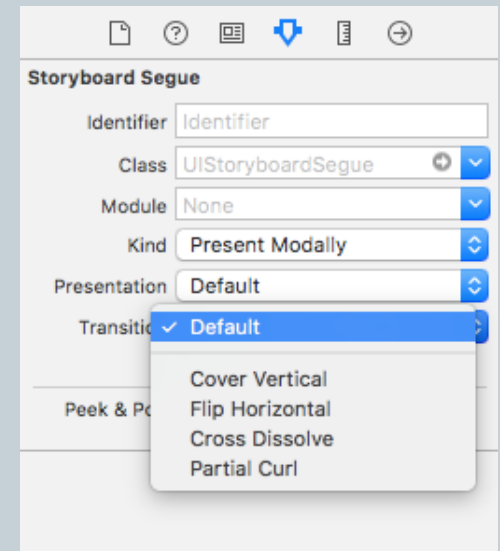
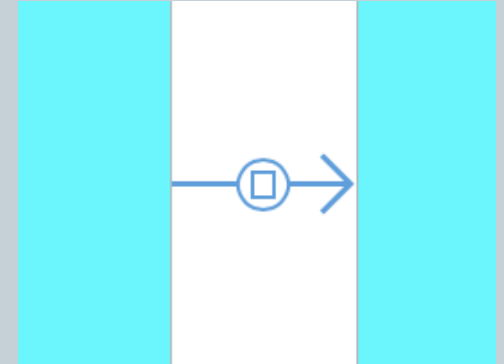
- Add button to first ViewController
- Add a second ViewController
- Click on Button on first ViewController
  - Press Control
    - Drag to second ViewController and release
      - Select Present Modally from the drop down



# Select Transition



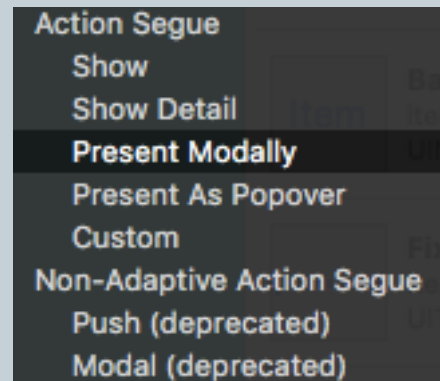
- Click on Segue
- Go to Attribute Inspector
  - Transition
    - Select from:
      - *Cover Vertical*
      - *Flip Horizontal*
      - *Cross Dissolve*
      - *Partial Curl*



# Page Navigation



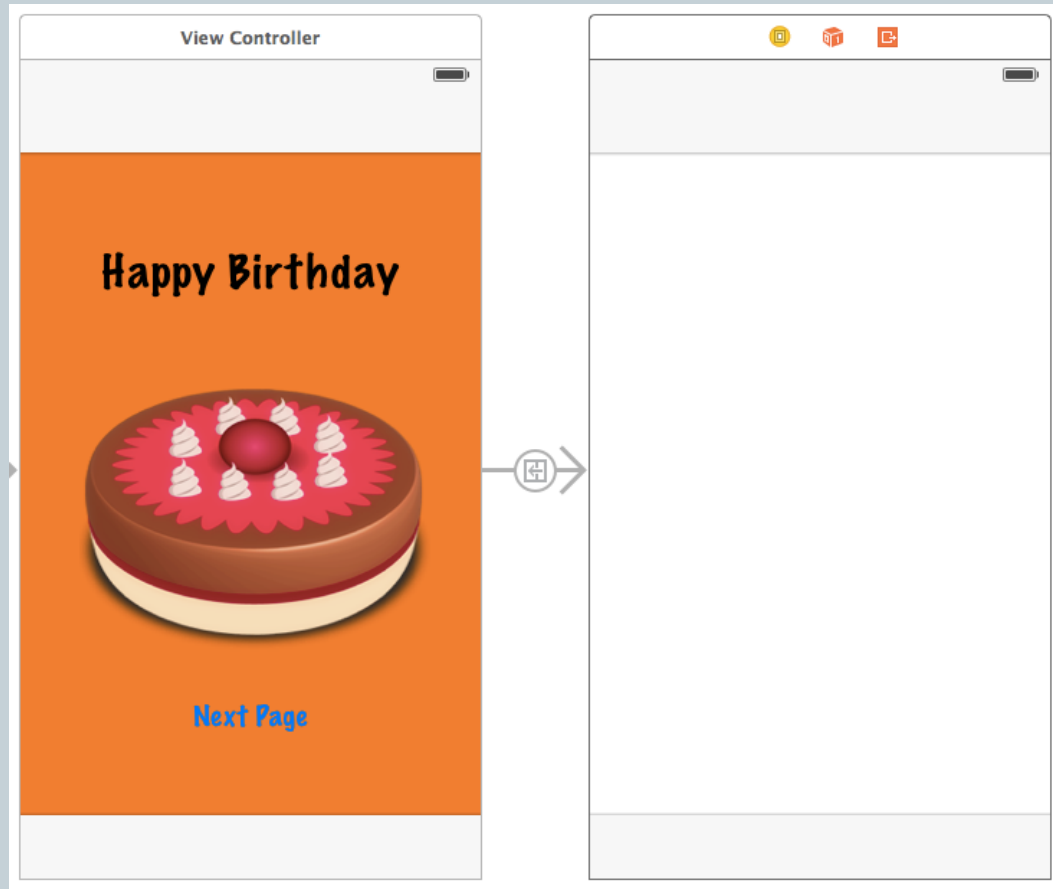
- **How to navigate from one page to another and back?**
- Add a *button* to view controller from object library
- Add another view controller to the storyboard
- Click on the *button* on page 1
  - Press Ctrl
  - Drag to second view controller
  - select **show**



# Navigation implemented



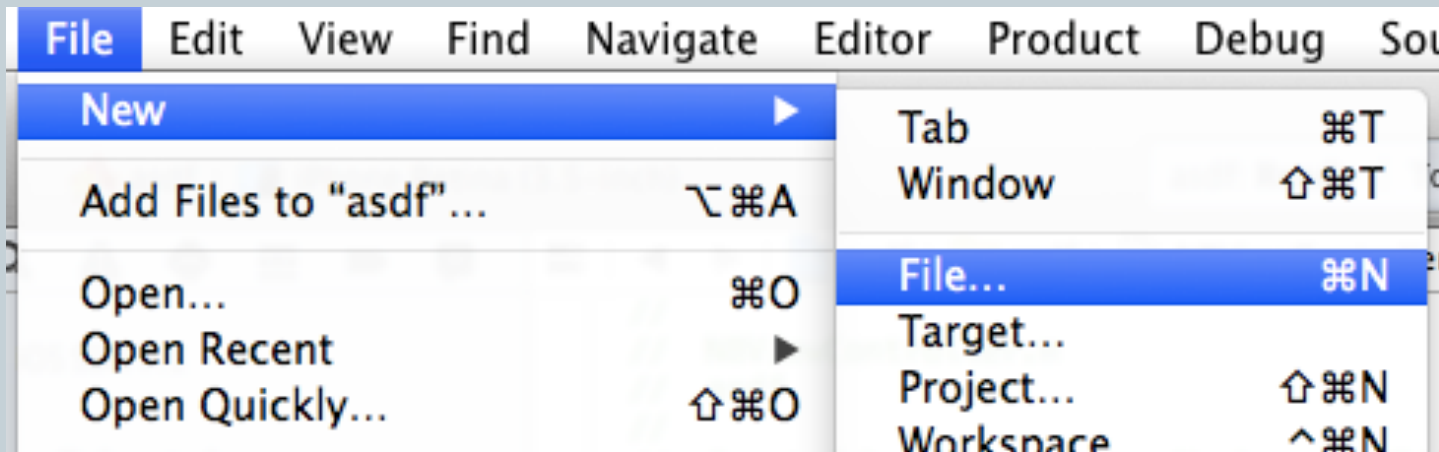
- Clicking the button will navigate to the 2<sup>nd</sup> page



# View Controller Classes

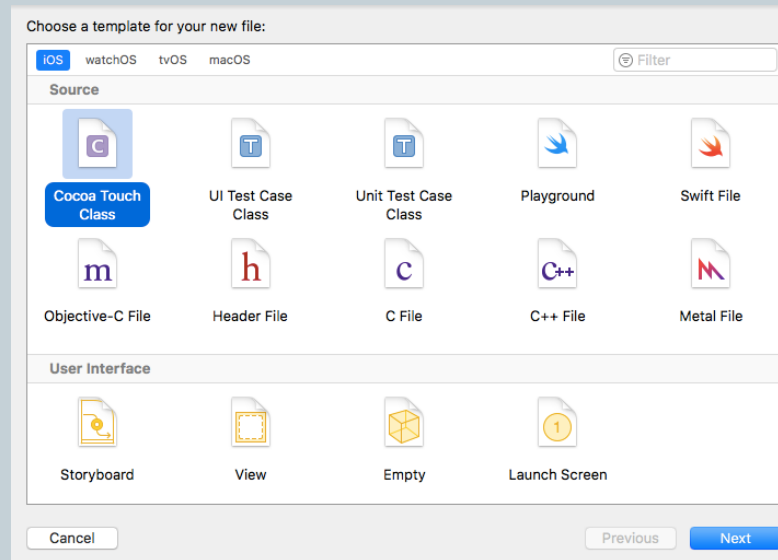


- *Each View Controller in the storyboard must have a “**ViewController**” class assigned to it*
- Creating a View Controller Class
  - File → New → File



# Creating a “ViewController” Class

- iOS → Cocoa Touch Class → Next





# Creating a “ViewController” Class



- Subclass of: UIViewController

A screenshot of the Xcode class creation dialog. It shows a form with the following fields: 'Class:' with the text 'SecondViewController' entered; 'Subclass of:' with a dropdown menu showing 'UIViewController'; an unchecked checkbox labeled 'Also create XIB file'; and 'Language:' with a dropdown menu showing 'Swift'. The 'Class:' field is highlighted with a blue border.

Class: SecondViewController

Subclass of: UIViewController

☐ Also create XIB file

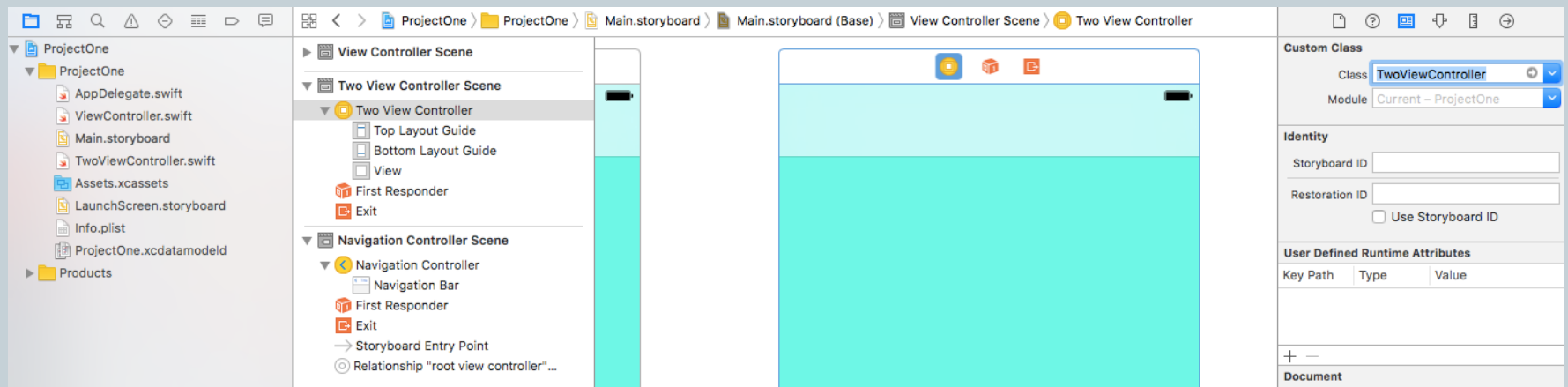
Language: Swift

- → Next → Create

# View Controller Class Assignment



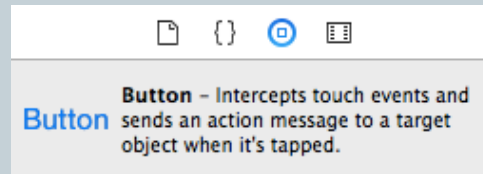
- Storyboard View Controller
  - Must be linked to the created ViewController class
- → Click on View Controller on Storyboard → Identity Inspector → Select class from dropdown



# Xcode Object Button

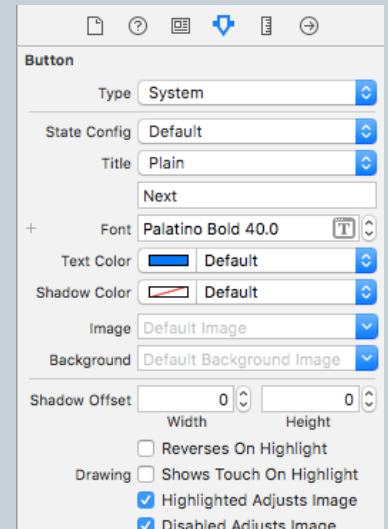
- How to add a button to the View Controller?

- ÷ Click on “Objects Library”
- ÷ Type “Button” in the search field
- ÷ Drag a “Button” to the storyboard




- Go to Attribute Inspector

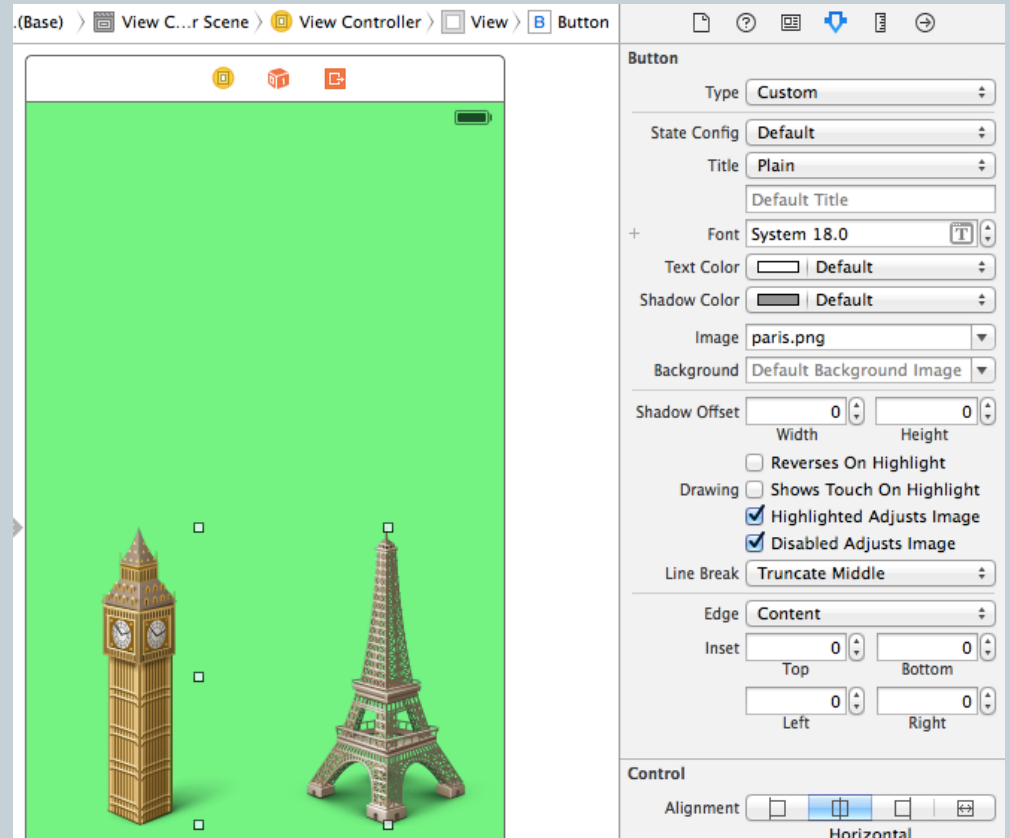
Customise your button by changing the title, text colour, font type & image.



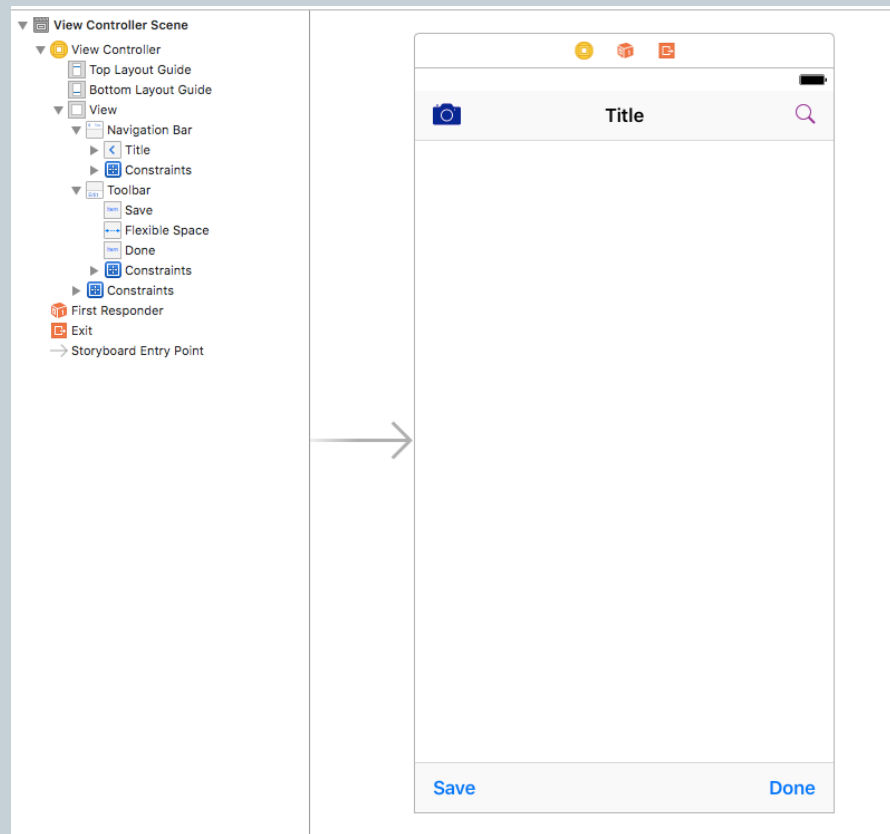
# Button Image



- Click on Button
- Go to  Attribute Inspector
- Button
  - Image
    - ÷ Select your image



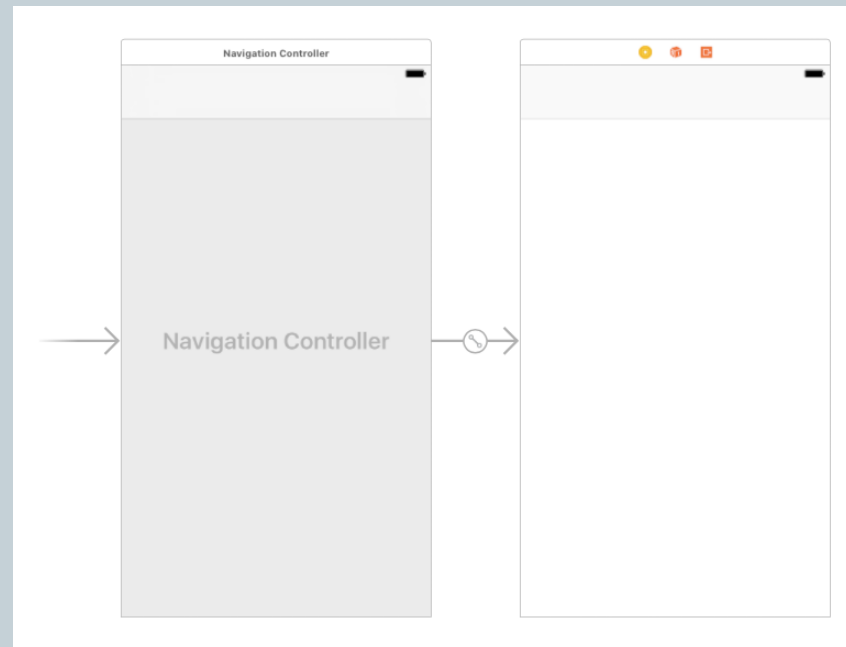
# Navigation Bar and Toolbar



# Navigation Bar and Toolbar



- Click on the ViewController on the Storyboard
- Editor - Embed In - Navigation Controller
- Navigation Controller and Top Navigation bar added



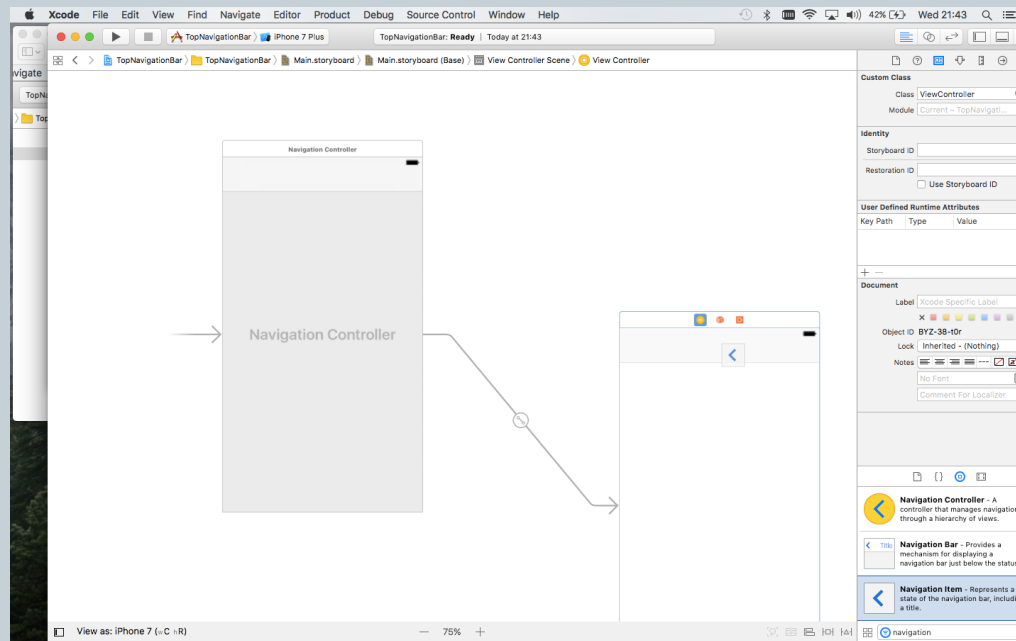
# Navigation Bar and Toolbar

43

- From Object Library
- Add a Navigation Item to the top of the bar
  - Double click on it and change the title



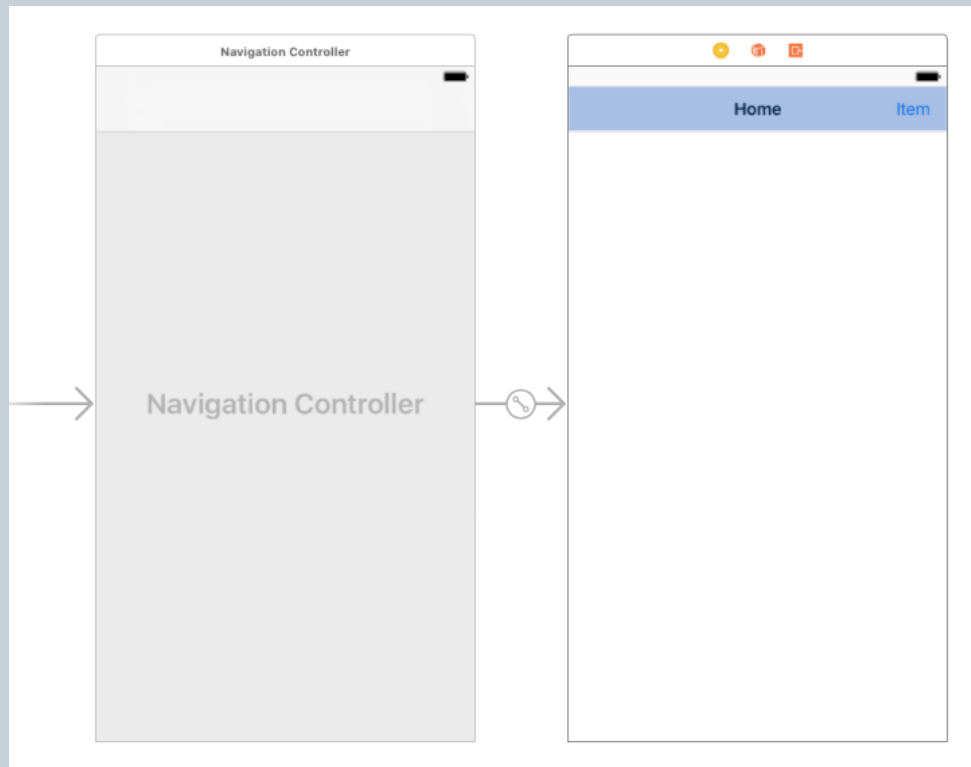
**Navigation Item** - Represents a state of the navigation bar, including a title.



# Navigation Bar and Toolbar



- From Object Library
- Add a Bar Button Item

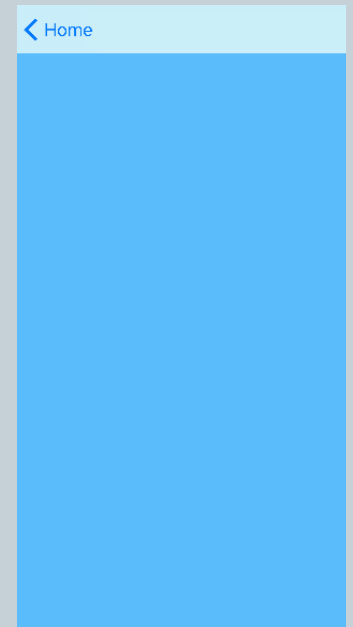




# Navigation Bar and Toolbar



- Add another ViewController to your Storyboard
- Click on the Bar Button Item
  - Press Ctrl
    - Drag to the next ViewController
    - Click on Show
- Back Button will be added automatically



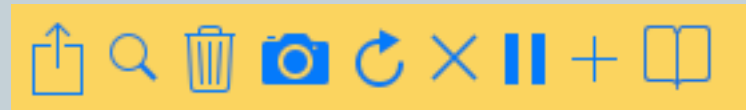
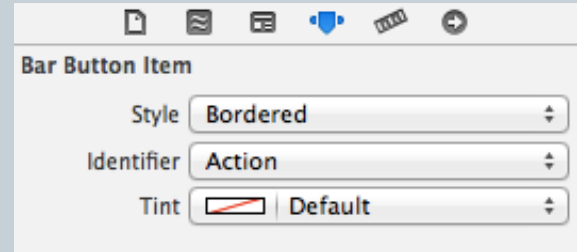
# Bar Button Item



- Bar Button Items are used for

- Toolbar
- Navigation Bar

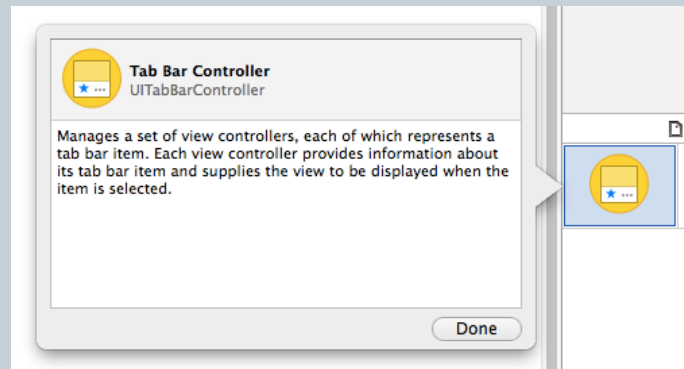
- Click on “Objects Library”
- Type “Bar Button Item” in the search field
- Drag a “Bar Button” to the storyboard
- Click on the button
- → Attributes Inspector
  - Bar Button Item
    - ÷ Identifier
      - Try different identifiers and see what happens



# Tab Bars



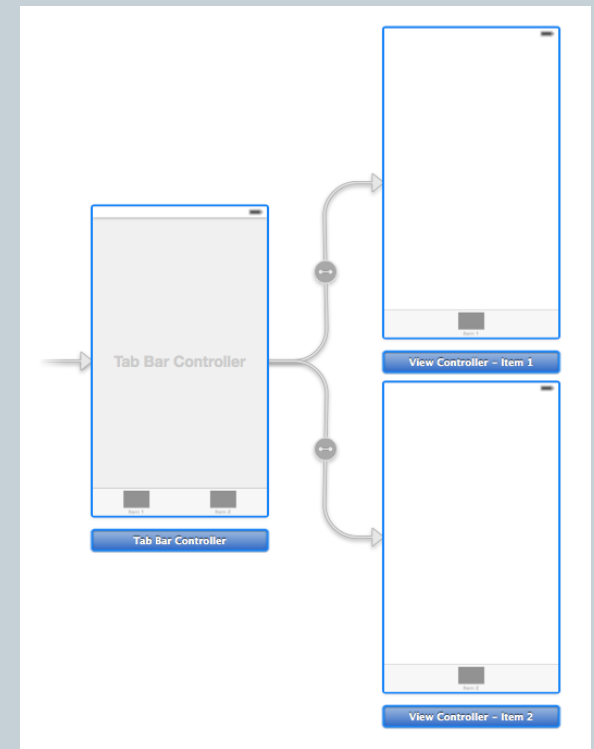
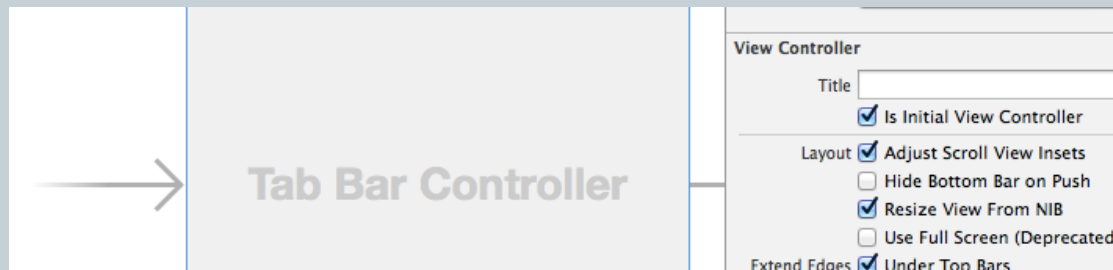
- **We are going to create an app with a tab bar**
- Delete the current View Controller
  - →Click on it →Press “delete”
- Go to the “Objects Library” & enter “tab bar” in the search field
- Drag the “Tab Bar Controller” to Main.storyboard



# Tab Bar Controller Added



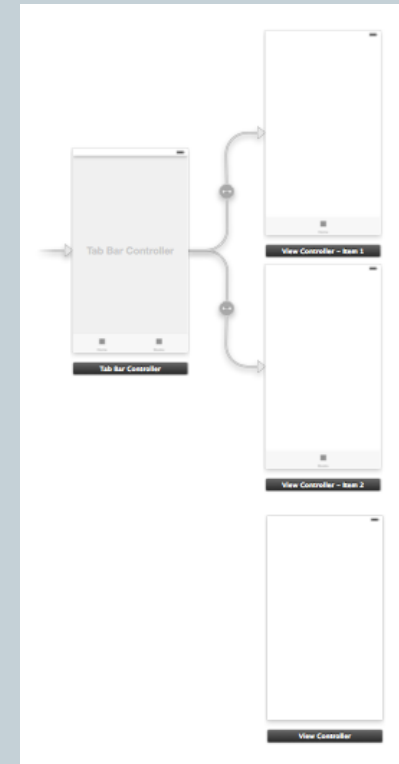
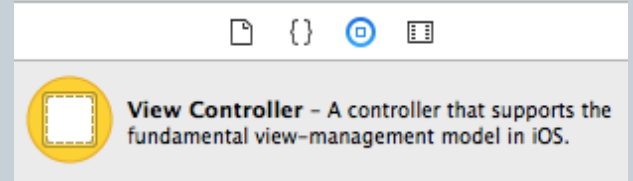
- Tab Bar Controller has been added and consists of
  - Tab Bar Controller
    1. View Controller Item 1
      - **This is tab 1**
    2. View Controller Item 2
      - **This is tab 2**
- Click on Tab Bar Controller
  - Check “is Initial View Controller”



# Add third Tab



- → Go to the Object Library
- Click on the View Controller object
  - Drag to the storyboard
- Third View Controller
  - ÷ Added
    - Not connected



# Link the new tab



- Add the third View Controller to Tab Bar Controller
  - Click on the Tab Bar Controller
  - Press “Control”
  - With the mouse
    - ÷ Link the Tab Bar Controller to the third View Controller
    - ÷ Let the mouse go
    - ÷ Select “**view controllers**”
    - ÷ from Relationship Segue

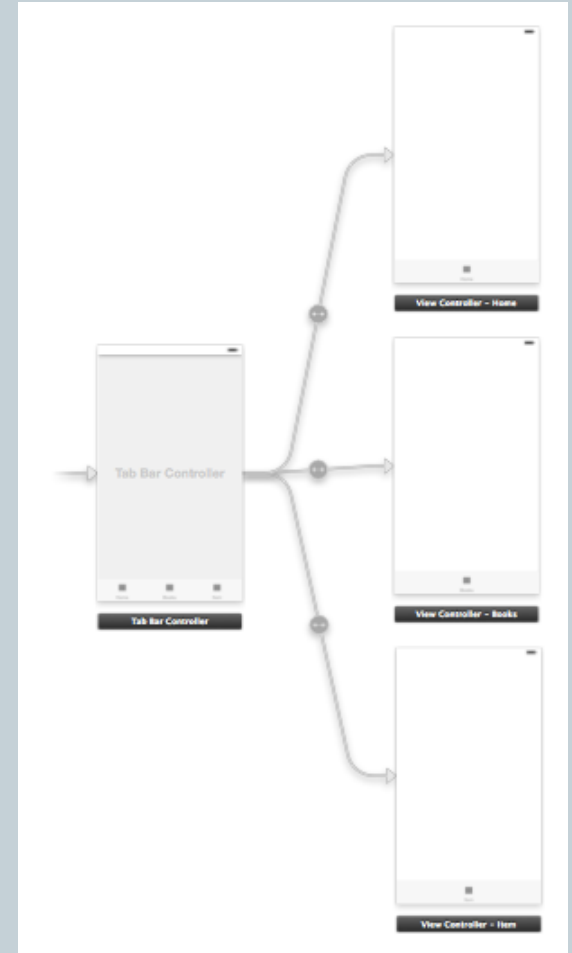
A screenshot of the segue menu that appears in an iOS storyboard when the 'Control' key is pressed. The menu is dark gray with white text. It lists several segue types under three main categories: Manual Segue, Relationship Segue, and Non-Adaptive Manual Segue. The 'view controllers' option under Relationship Segue is highlighted, indicating it is the selected segue type.

- Manual Segue
  - show
  - show detail
  - present modally
  - popover presentation
  - custom
- Relationship Segue
  - view controllers**
- Non-Adaptive Manual Segue
  - push (deprecated)
  - modal (deprecated)

# The third Tab has been added

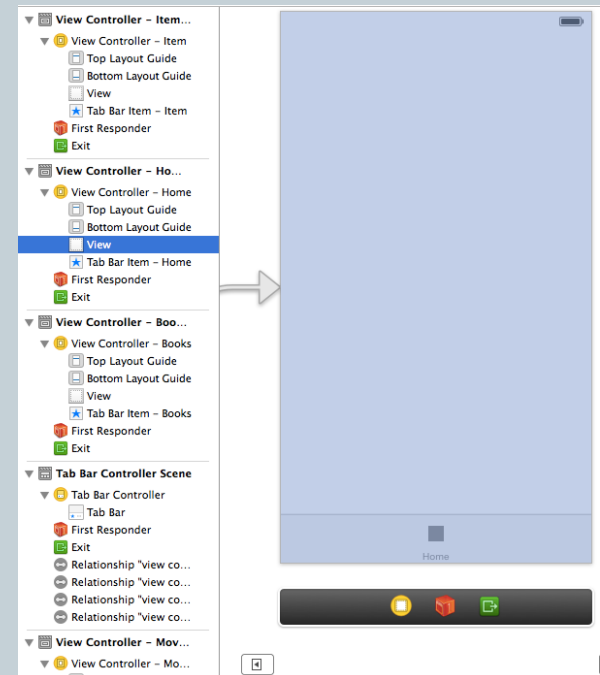


- **Change the name of Tab Bar**
  - Click on Item
  - Enter new name
- **Exercise**
  - Add three more tabs
  - Name them



# Change the background colour of each view

- Click on the first View Controller
- Click on Show Document Outline
  - If “Document Outline” not already displayed
- Click on **View**
- Click on Attribute Inspector
- Change Background Colour
  - From White to another colour
- Let's run the app!



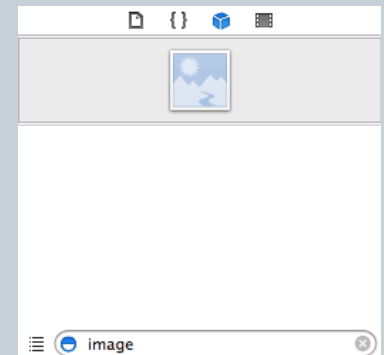
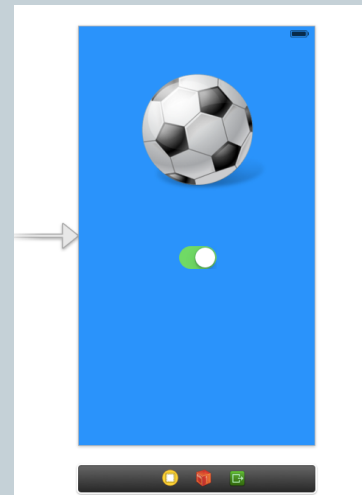
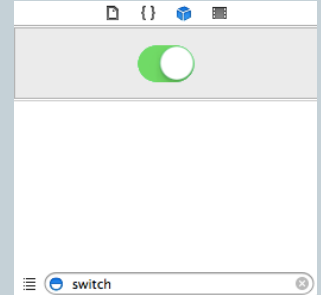


# Xcode Object Switch



# Add objects to the Storyboard

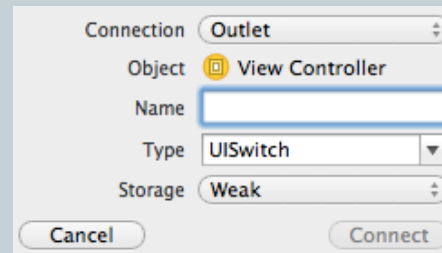
- Create a new “Single View Application” project
- Click on Main.storyboard
- Click on the Objects Library
  - Drag the following to View Controller
    - *Switch*
    - *Image View*
    - *Label*
- Add an image to the “Image View”



# Create IBOutlet Connection



- Click on the “Assistance Editor”
  - This will display the `ViewController.swift`
- Click on the switch on storyboard
  - Press “control”
  - Drag to the `ViewController.swift` file
    - ÷ The IBOutlet connection window will appear
      - Populate Name field
- Create IBOutlet connection for image and label



# Create IBOutlet Connection



- Create connection for image and label
  - Click on “Image View” on storyboard
    - ÷ Press “control”
    - ÷ Drag to the ViewController.swift file
      - The IBOutlet connection window will appear
        - Populate Name field
  - Click on “Label” on storyboard
    - ÷ Press “control”
    - ÷ Drag to the ViewController.swift file
      - The IBOutlet connection window will appear
        - Populate Name field

# Create IBOutlet Connection



- IBOutlet connections have been created

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var myLabel: UILabel!
    @IBOutlet weak var myImage: UIImageView!
    @IBOutlet weak var mySwitch: UISwitch!
}
```

# Insert and Implement Switch Action

- → Storyboard
- Click on switch → Ctrl → Drag to ViewController.swift file
- Connection window will appear
- Enter switchName
- Select Action from the Connection dropdown

- Switch has 2 Boolean states

- On
- Off

```
@IBAction func switchAction(_ sender: Any)
{
    if mySwitch.isOn
    {
        myImage.alpha = 1
        myLabel.text = "Goal"
    }
    else
    {
        myImage.alpha = 0
        myLabel.text = "No Goal"
    }
}
```

The screenshot shows the Xcode Connection Inspector window. The 'Connection' dropdown is set to 'Action'. The 'Object' is 'View Controller' with a yellow icon. The 'Name' field is empty and highlighted with a blue border. The 'Type' dropdown is set to 'id'. The 'Event' dropdown is set to 'Value Changed'. The 'Arguments' dropdown is set to 'Sender'. At the bottom, there are 'Cancel' and 'Connect' buttons.

# Xcode Object Slider



# Steps



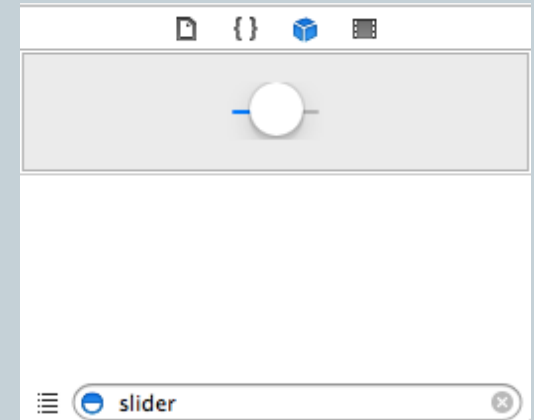
- Add a slider to the storyboard
- Change the view background colour to orange
- Create the IBOutlet connection for the slider
- Create IBAction connection for the slider
- Write slider method implementation



# Add a slider to the Storyboard



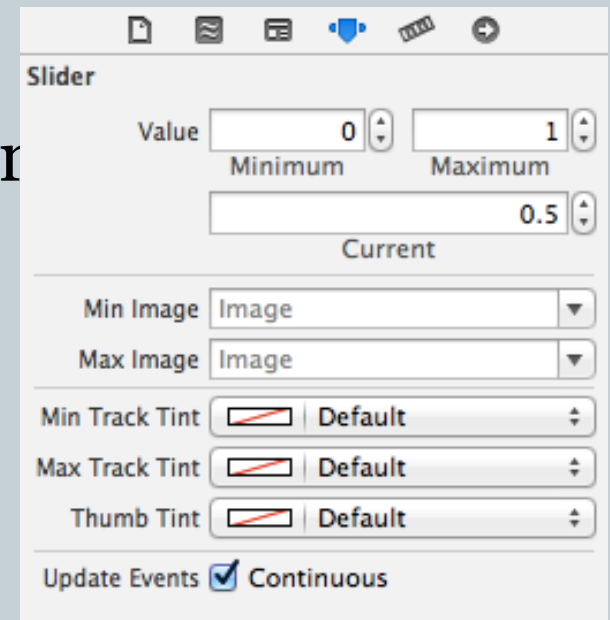
- Create a new “Single View Application” project
- Click on Main.storyboard
- Click on the Objects Library
- Drag a slider to the storyboard
- Change View Background colour



# Slider Values



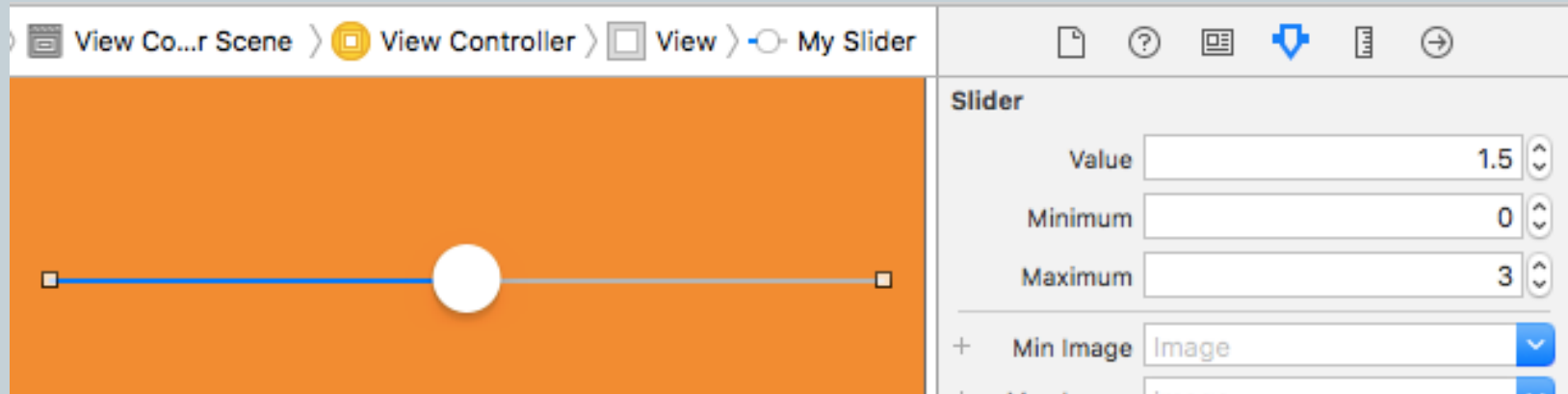
- Slider has the following values:
  - Minimum
  - Maximum
  - Current
- Slider values can be set
- → Storyboard → Click on slider
  - → Attributes Inspector



# Slider Properties



- Click on the Slider on the storyboard
- Go to Attribute Inspector
- Edit following fields:
  - Value
  - Minimum
  - Maximum



# Create IBOutlet Connection for Slider



- Click on the “Assistance Editor”
- Click on the slider on story board
  - Press “control”
  - Drag to the ViewController.swift file
    - ÷ The IBOutlet connection window will appear
- Populate the Name field
  - sliderName

A screenshot of the 'IBOutlet Connection' dialog box in Xcode. The 'Connection' dropdown is set to 'Outlet'. The 'Object' dropdown is set to 'View Controller'. The 'Name' field is empty and highlighted with a blue border. The 'Type' dropdown is set to 'UISlider'. The 'Storage' dropdown is set to 'Weak'. At the bottom, there are 'Cancel' and 'Connect' buttons.

Connection: Outlet

Object: View Controller

Name:

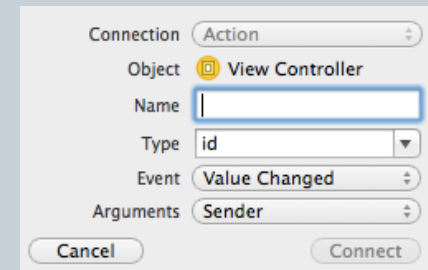
Type: UISlider

Storage: Weak

Buttons: Cancel, Connect

# Slider IBOutlet Connection & Implement

- → Storyboard
- Click on slider → Ctrl → Drag to ViewController.swift
- Connection window will appear
- Enter sliderName
- Make IBAction connection



- Select Action from Connections dropdown

```
@IBAction func changeColour(_ sender: Any)
{
    if mySlider.value < 1
    {
        self.view.backgroundColor = UIColor.green
    }
    else if mySlider.value > 2
    {
        self.view.backgroundColor = UIColor.red
    }
    else
    {
        self.view.backgroundColor = UIColor.orange
    }
}
```

# Auto Layout

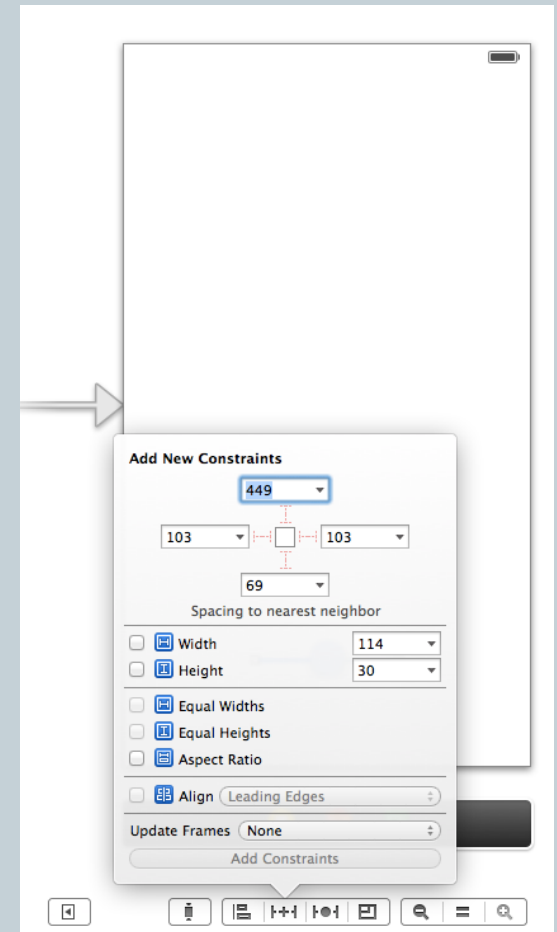


Introduced in iOS 6

# Pin Menu



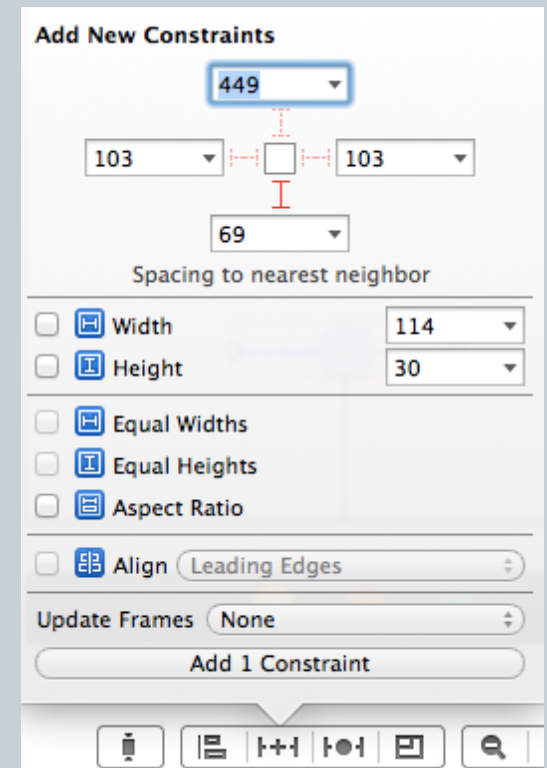
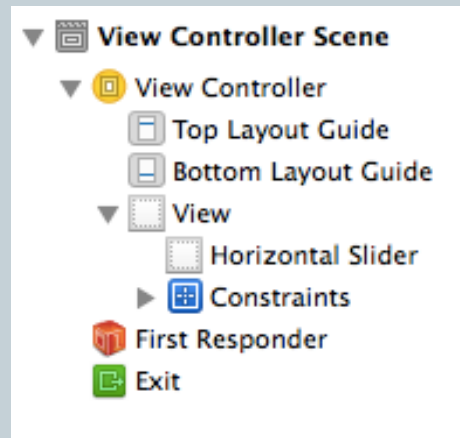
- Click on the object
- Select the Pin Menu



# Pin Menu



- Adjust relative distance to bottom of the screen
  - Add New Constraints
  - Spacing to nearest neighbour
  - Add 1 Constraint
- You will notice
  - Constraint has been added to
    - ÷ Document Outline

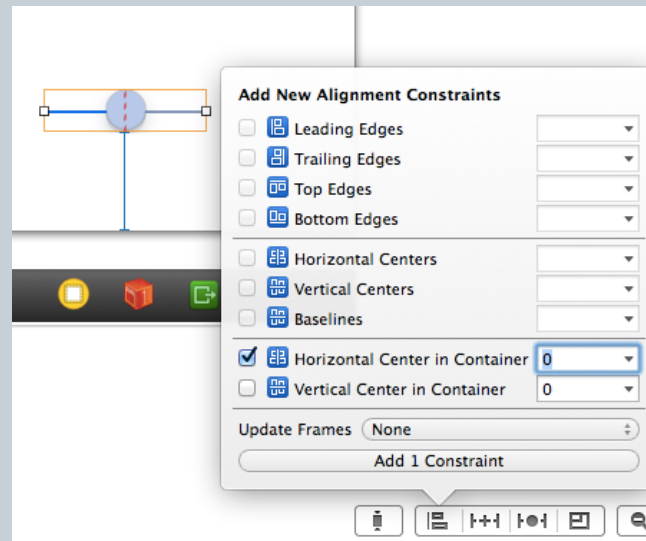




# Alignment Menu




- Click on the Object
- Click on the alignment menu
- Check horizontal Centre in Container
- Click on Add 1 Constraint

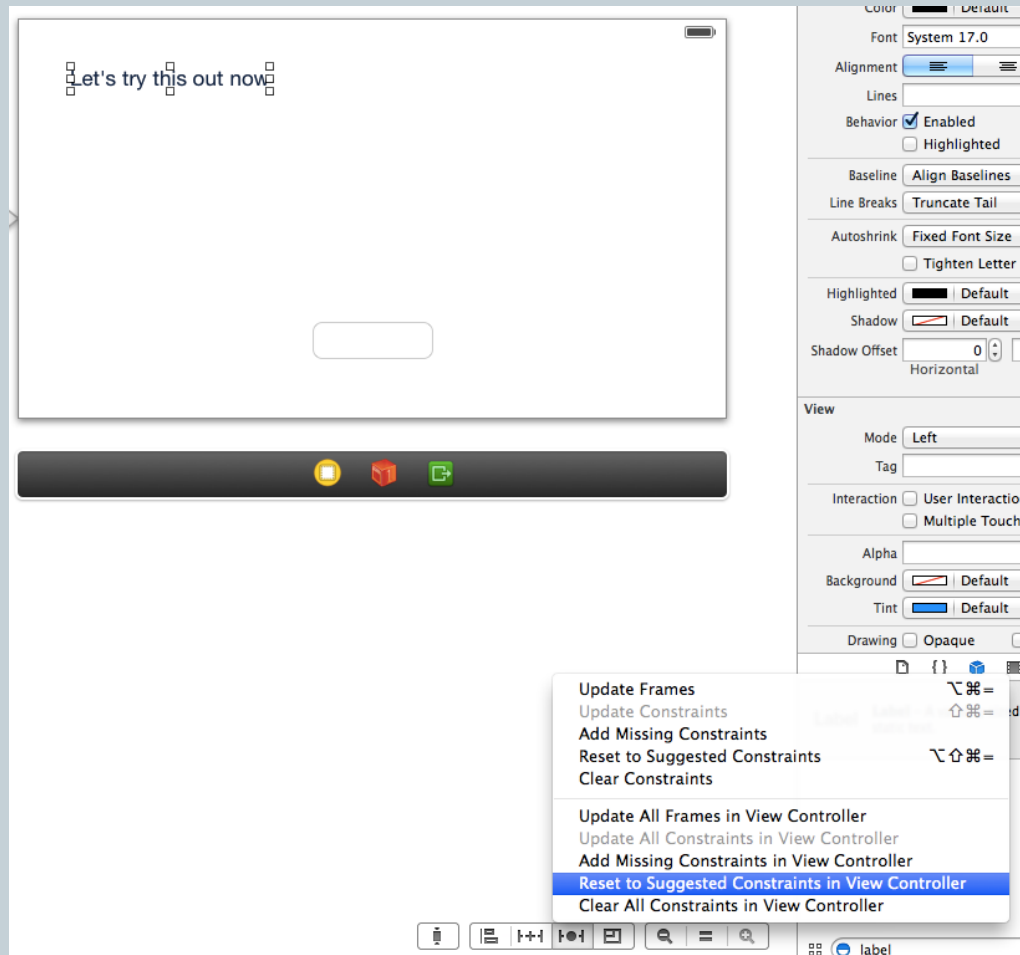


# Resolve Auto Layout issues



- Add label to top left
- Click on label
- Click on Resolve Auto Layout Issues icon 
- Reset to suggested Constraints in View Controller

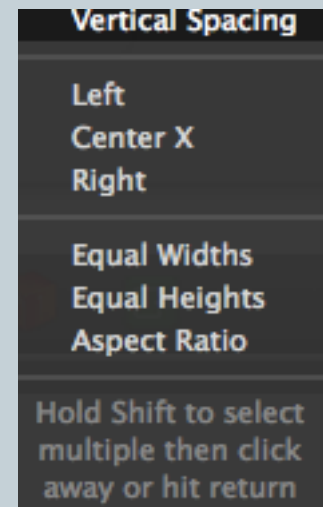
# Resolve Auto Layout issues



# Aligning Objects with each other



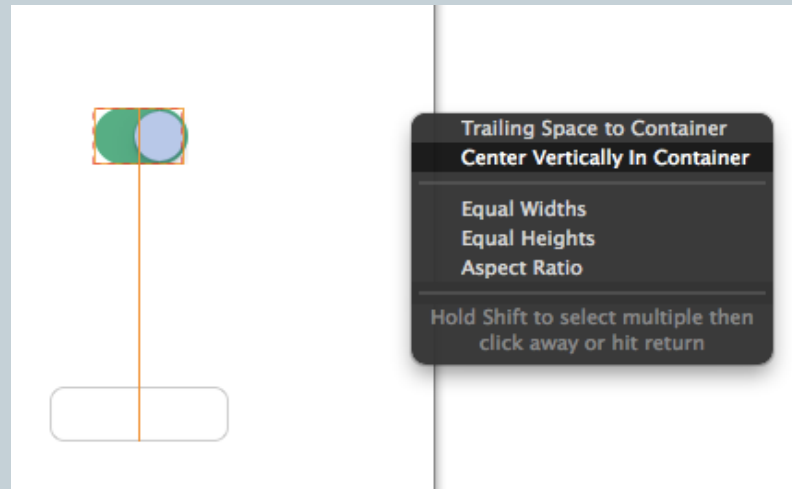
- Sometimes you want to align objects with each other
- Click on one object
- Press control
- Drag to the second object and let go
- Black menu will appear
- Click on Centre X



# Can also align vertically




- Click on Object
- Press Control
- Drag to the right and let go
- Centre Vertically in Container



# Resolve Auto Layout Issues



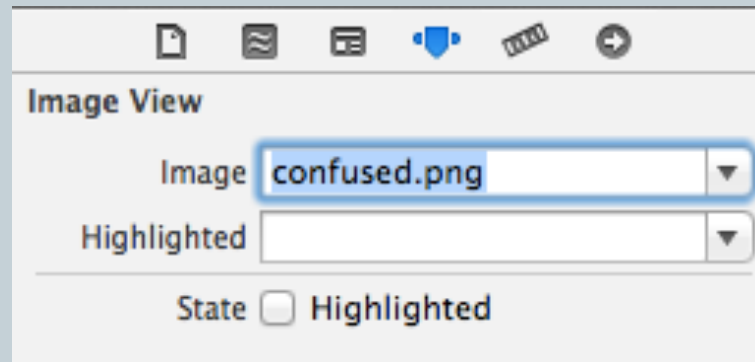
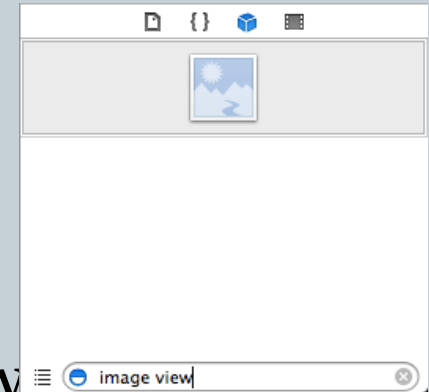
- Try again and you find objects are not centered again
- Click on object
  - Orange box indicates where object should be
- Resolve Auto Layout Issues 
- Update Frames

# Gesture Recognizers



# Add an Image View to the Storyboard

- Click on Main.storyboard
- Click on the Objects Library
- Drag an Image View to the storyboard
- Click on the Image View
  - Attribute Inspector → Image View
    - Select image from the drop down list
      - ÷ This is the image that will initially appear





# Enable Interaction and Multiple Touch

## ● Enable Interaction and Multiple Touch

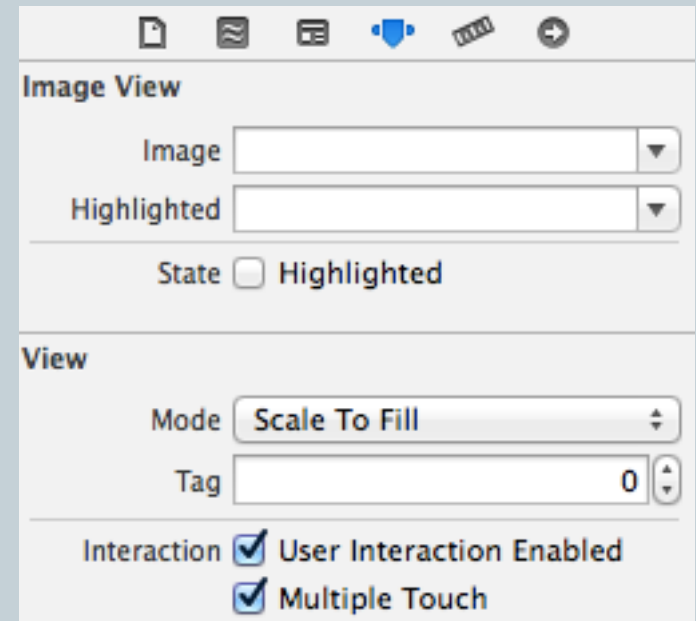
- Click on the Image View on the storyboard

- → Attributes Inspector → View

- ÷ Ensure following fields are checked:

- **User Interaction Enabled**

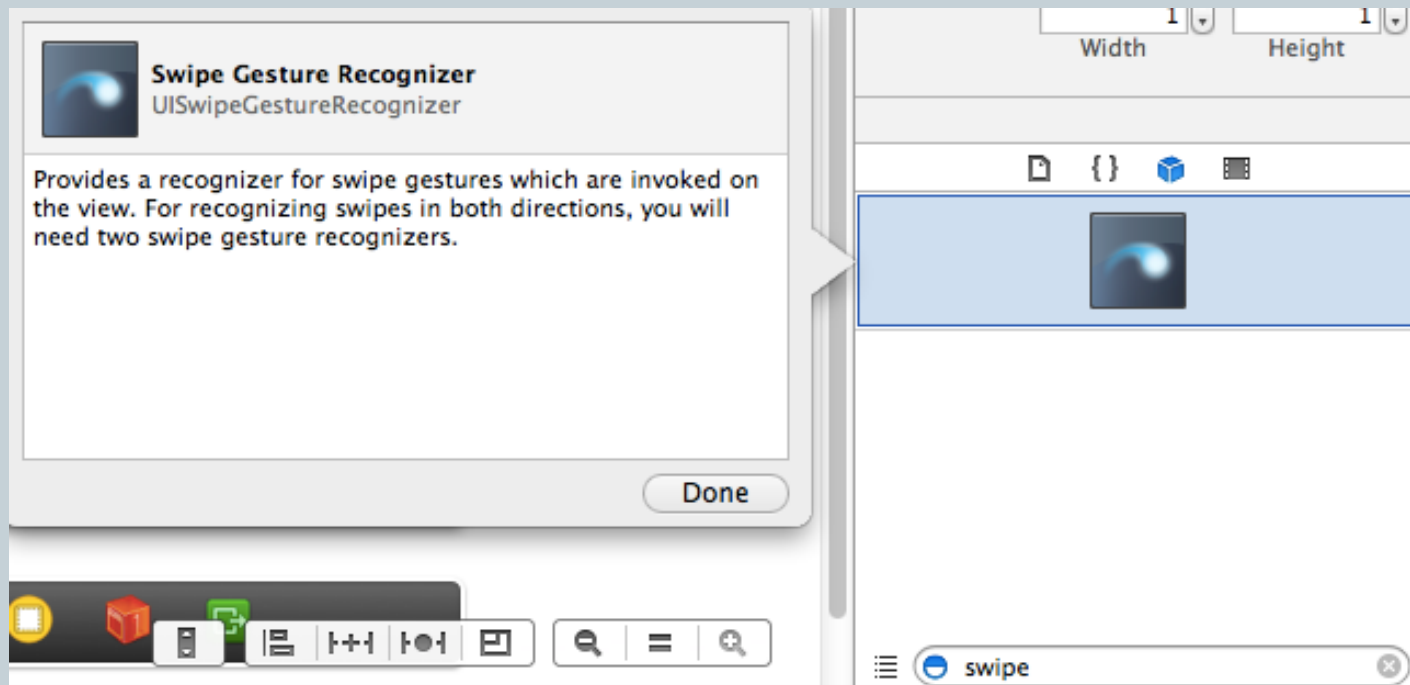
- **Multiple Touch**



# Add the swipe gestures

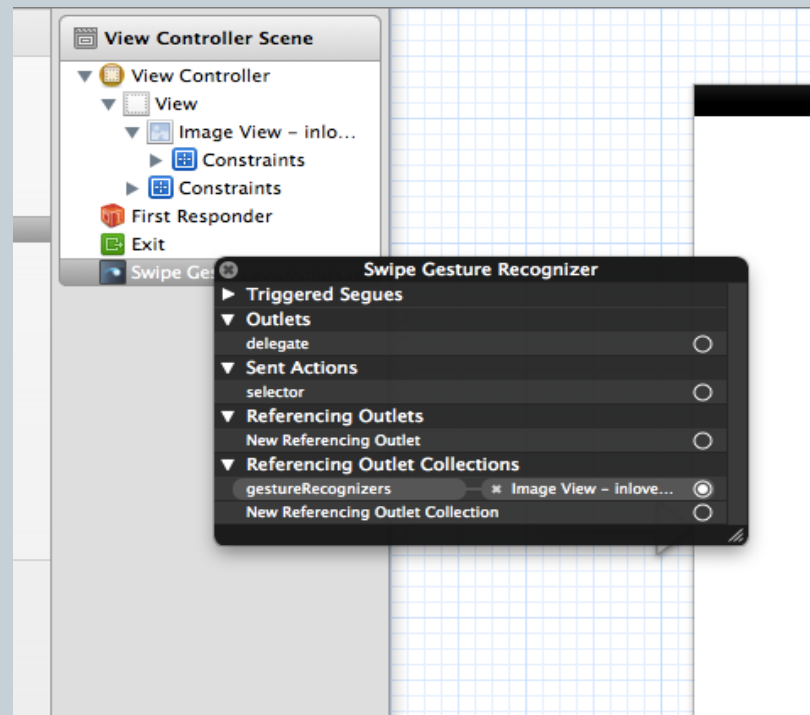


- → Objects Menu
- Select a “Swipe Gesture Recognizer”
- Drag and place it on top of the Image View



# Right “Swipe Gesture Recognizer” added

- You will notice that the “Swipe Gesture Recognizer” appears under the “Document Outline” and if you right click on it, there should be a connection to the view as shown below:



# Right “Swipe Gesture Recognizer” added



- If you click on the “Swipe Gesture Recognizer” on the Document Outline and then click on the Attribute Inspector, you will notice that it is a **Right** Swipe



# Left “Swipe Gesture Recognizer” added

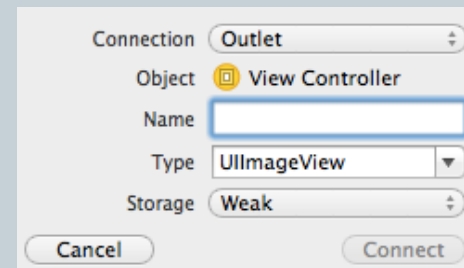
- We also need to be able to do a left swipe
- → Objects Menu
  - drag another “Swipe Gesture Recognizer over the image View
- This will then appear under “Document Outline
  - → Click on it → Attribute Inspector
    - ✎ → Change Swipe attribute to “Left”



# Create IBOutlet Connection for Image View

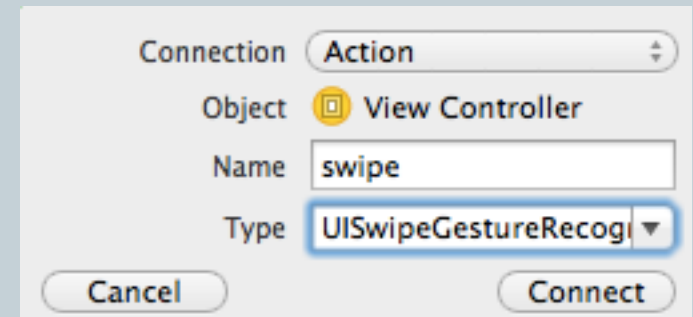
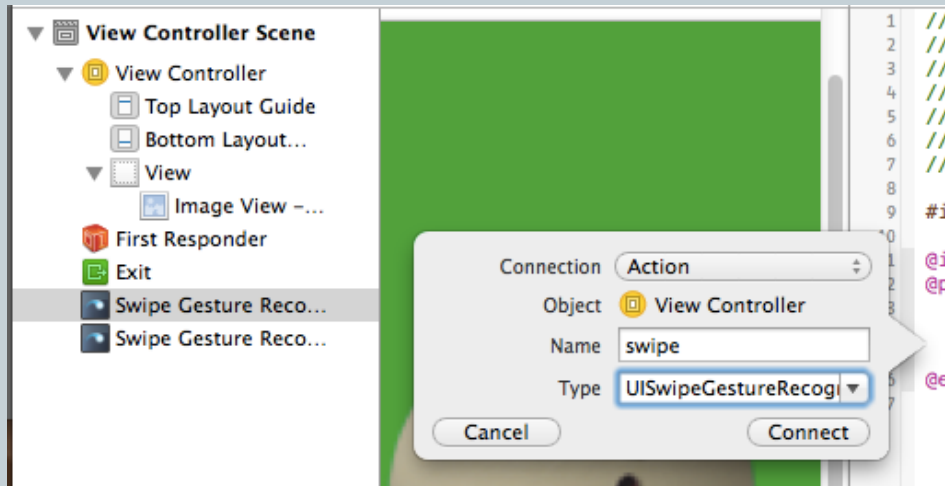


- Click on the “Assistance Editor”
  - This will display the ViewController.swift file
  - If not, the file can be manually selected
- Click on the image view on storyboard
  - Press “control”
  - Drag to the Header File
    - ÷ The IBOutlet connection window will appear
- Populate the Name field
  - imageViewName



# Create the Gesture IBOutlet Connection

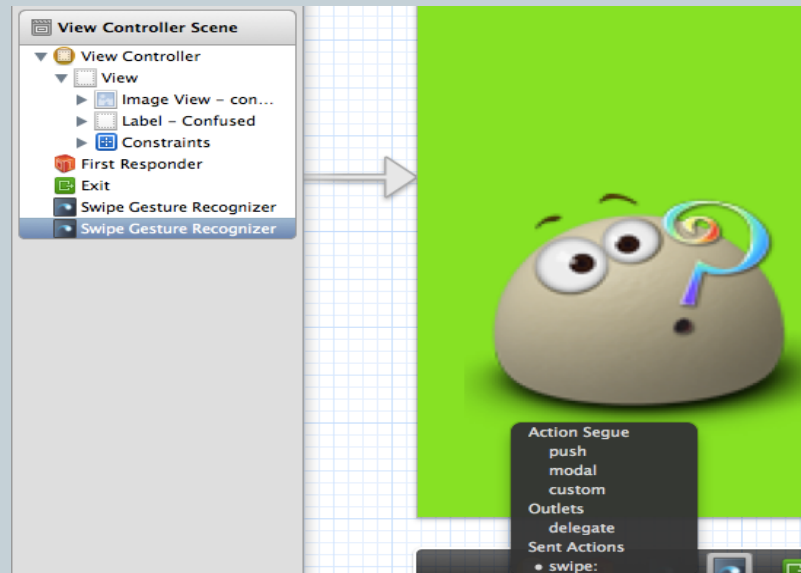
- → Document Outline
- → Click on the Right “Swipe Gesture Recognizer”
- Press “Ctrl” → Drag to the Header File
  - Connection: Action
  - Type: UISwipeGestureRecognizer



# Connect Left “Swipe Gesture Recognizer”



- → Document Outline
- → Click on the Left “Swipe Gesture Recognizer” → Press “Ctrl” → Drag to the “View Controller”
  - Select the Sent Action method implemented as shown below





# Swipe Method Implementation



- Swipe method implementation

```
class ViewController: UIViewController
{
    @IBOutlet weak var swipeImage: UIImageView!
    @IBOutlet weak var myLabel: UILabel!

    @IBAction func rightSwipe(_ sender: AnyObject)
    {
        swipeImage.image = UIImage (named: "happy.png")
        myLabel.text = "Happy"
    }

    @IBAction func leftSwipe(_ sender: AnyObject)
    {
        swipeImage.image = UIImage (named: "angry.png")
        myLabel.text = "Angry"
    }
    override func viewDidLoad()
    {
        super.viewDidLoad()
    }
}
```

# Shake Gesture



# Shake Functions



- Function called when shake begins

```
override fun motionBegan(_ motion: UIEventSubtype, with event: UIEvent?) {  
}
```

- Function called when shake ends

```
override fun motionEnded(_ motion: UIEventSubtype, with event: UIEvent?) {  
}
```

# Random Number Generator

A decorative circle with a teal outline and a white center, positioned below the title and above the list.

- **arc4random()%n**
- **let** a = arc4random()%**10**
  - Generate a random number between 0 and 9

# arc4random()%n Example

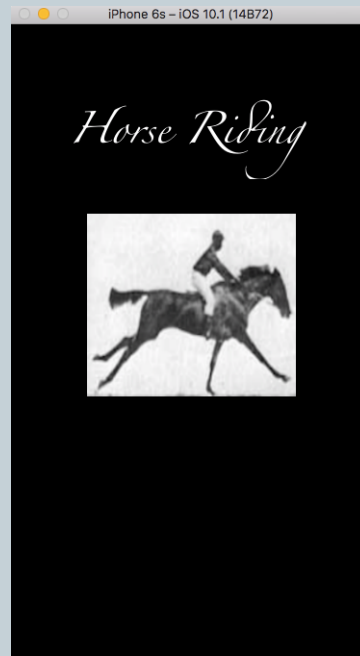


- Example:

```
let a = arc4random()%5

if a == 0
{
    self.view.backgroundColor = UIColor.green
}
if a == 1
{
    self.view.backgroundColor = UIColor.blue
}
if a == 2
{
    self.view.backgroundColor = UIColor.purple
}
if a == 3
{
    self.view.backgroundColor = UIColor.yellow
}
if a == 4
{
    self.view.backgroundColor = UIColor.cyan
}
```

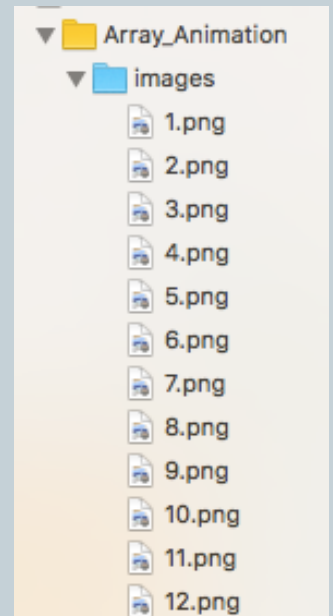
# Animation



# Array Animation Steps



- Steps:
  1. Add array image to the project
  2. Storyboard: Add an UIImageView to the ViewController
  3. Make UIImageView IBOutlet Connection
  4. ViewController Implementation



# ViewController Implementation




```
class ViewController: UIViewController {  
  
    @IBOutlet weak var animationImage: UIImageView!  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
  
        animationImage.animationImages=[  
            UIImage (named: "images/1.png")!,  
            UIImage (named: "images/2.png")!,  
            UIImage (named: "images/3.png")!,  
            UIImage (named: "images/4.png")!,  
            UIImage (named: "images/5.png")!,  
            UIImage (named: "images/6.png")!,  
            UIImage (named: "images/7.png")!,  
            UIImage (named: "images/8.png")!,  
            UIImage (named: "images/9.png")!,  
            UIImage (named: "images/10.png")!,  
            UIImage (named: "images/11.png")!,  
            UIImage (named: "images/12.png")!  
        ]  
  
        animationImage.animationDuration = 1.1  
        animationImage.animationRepeatCount = 0  
        animationImage.startAnimating()  
    }  
}
```





# Array Animation



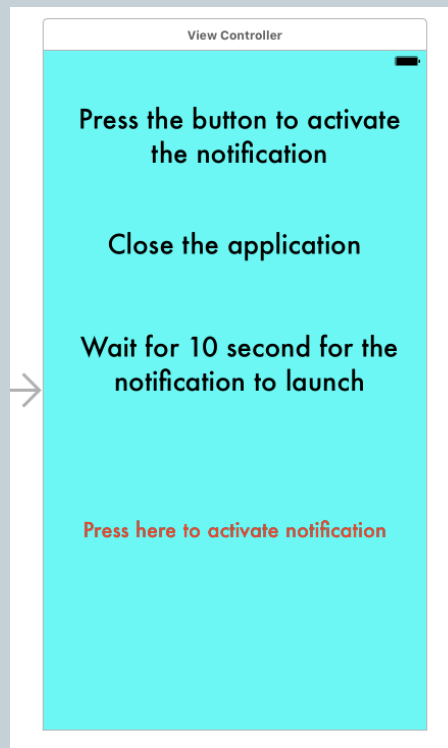
- Animation code added to viewDidLoad
- Animation will start as soon as view is loaded
- *Exercise:*
  - Click on animationRepeatCount
    - Click on i to launch Quick Help 

```
animationImage.animationRepeatCount = 0
```

**Quick Help**  
  
Declaration `var animationRepeatCount: Int { get set }`  
Description Specifies the number of times to repeat the animation.  
The default value is 0, which specifies to repeat the animation indefinitely.  
Availability iOS (8.0 and later), tvOS (9.0 and later)  
Declared In [UIKit](#)  
More [Property Reference](#)

# Local Notification



# Local and Push Notifications



- One application can be active in the foreground at any time
- Local notifications and push notifications are ways
  - for an application that isn't running in the foreground
  - ÷ to let its users know it has information for them
- The information could be:
  - a message
  - an impending calendar event
  - or new data on a remote server

# Local and Push Notifications



- When presented by the operating system
  - Local and Push Notifications
    - ÷ Look and sound the same
- When users are notified
  - they can launch the application and see the details
  - They can also choose to ignore the notification
    - ÷ in which case the application is not activated

# Local and Push Notifications



- Add button to storyboard
- Make IBAction connection to ViewController
- import frameworks
  - `import` UserNotifications
- implement Protocols
  - `class` ViewController: UIViewController, UNUserNotificationCenterDelegate
- Add boolean variable
  - `var` isGrantedNotificationAccess:Bool = `false`
- Edit ViewDidLoad method - See next slide
- Implement IBAction function

# viewDidLoad



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

    UNUserNotificationCenter.current().requestAuthorization(
        options: [.alert, .sound, .badge],
        completionHandler: { (granted, error) in
            self.isGrantedNotificationAccess = granted
        }
    )
}
```

# IBAction function



```
@IBAction func setNotification(_ sender: Any)
{
    if isGrantedNotificationAccess
    {
        //add notification code here

        //Set the content of the notification
        let content = UNMutableNotificationContent()
        content.title = "Notification Title"
        content.subtitle = "Notification Subtitle"
        content.body = "Notification Message"

        //Set the trigger of the notification -- here a timer.
        let trigger = UNTimeIntervalNotificationTrigger(
            TimeInterval: 10.0,
            repeats: false)

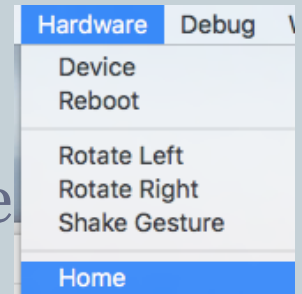
        //Set the request for the notification from the above
        let request = UNNotificationRequest(
            identifier: "10.second.message",
            content: content,
            trigger: trigger
        )

        //Add the notification to the currnet notification center
        UNUserNotificationCenter.current().add(request, withCompletionHandler: nil)
    }
}
```

# Close App and see what happens



- Run the app!
- Close the app and wait 10 seconds
  - Click on Simulator → Hardware → Home
- Notification will appear after 10 seconds
- Clicking on the notification message, will open the app





# Audio



# Steps



1. Import framework
2. Add the audio file to the project
3. Add Switch to storyboard
4. Make IBOutlet and IBAction connections for Switch
5. Start Audio play in ViewDidLoad
6. Write implementation code to switch audio on and off

# Import Framework

108

```
import UIKit
import AVFoundation

class ViewController: UIViewController
{
    var musicPlayer = AVAudioPlayer()

    @IBOutlet weak var switchAudio: UISwitch!

    override func viewDidLoad() {
        super.viewDidLoad()

        do
        {
            let url = Bundle.main.path(forResource: "high-energy", ofType: "wav")
            self.musicPlayer = try AVAudioPlayer(contentsOf: URL(fileURLWithPath: url!), fileTypeHint: nil)
            musicPlayer.prepareToPlay()
            self.musicPlayer.numberOfLoops = -1
            musicPlayer.play()
        }

        catch
        {
        }
    }
}
```

# Switch Function Implementation



```
@IBAction func turnOnOff(_ sender: AnyObject)
{
    if switchAudio.isOn
    {
        musicPlayer.play()
    }
    else
    {
        musicPlayer.pause()
    }
}
```

# Stop Audio



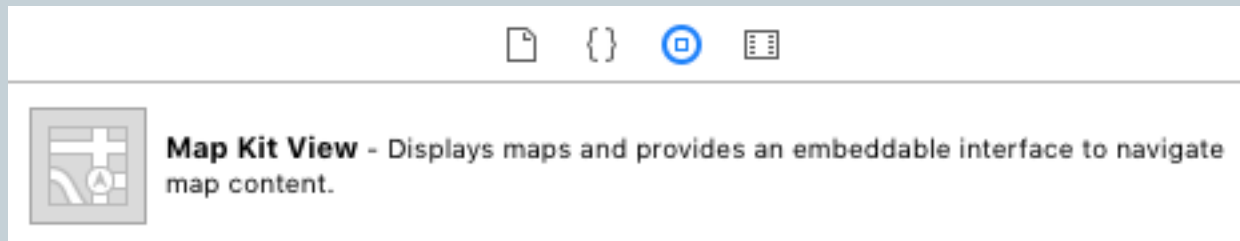
- Audio will be stopped in viewWillDisappear function

```
override fun viewWillDisappear(_ animated: Bool) {  
    mediaPlayer.stop()  
}
```

# Xcode Object Map View



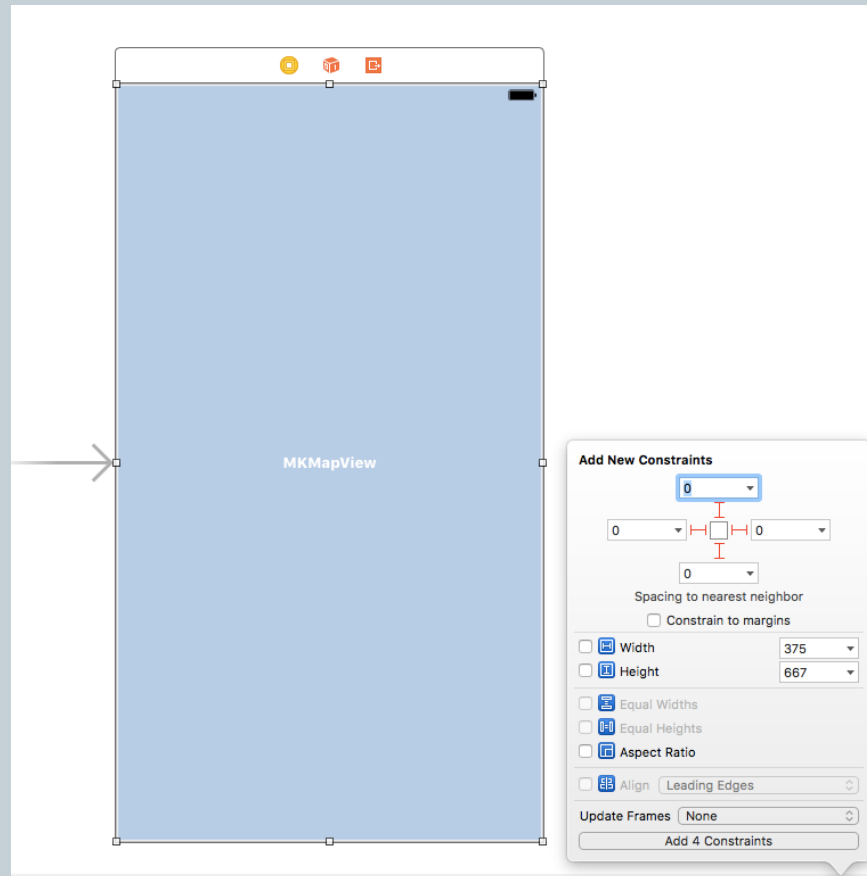
- Go to Object Library
- Add Map Kit View to the Storyboard



# AutoLayout



- Click on MapView and add 4 constraints as follows:





# Attribute Inspector



- Click on MapView
- Go to Attribute Inspector
- Enable **User Location, Scale** and **Traffic**
- Choose Map Type (*Map, Satellite or Hybrid*)

**Map View**

Type

Allows	<input checked="" type="checkbox"/> Zooming	<input checked="" type="checkbox"/> Scrolling
	<input checked="" type="checkbox"/> Rotating	<input checked="" type="checkbox"/> 3D View
Shows	<input checked="" type="checkbox"/> Buildings	<input checked="" type="checkbox"/> Compass
	<input checked="" type="checkbox"/> Scale	<input checked="" type="checkbox"/> Traffic
	<input checked="" type="checkbox"/> Points of Interest	
	<input checked="" type="checkbox"/> User Location	

# MapViewController Implementation



```
import UIKit
import MapKit
import CoreLocation

class ViewController: UIViewController{

    @IBOutlet weak var myMap: MKMapView!

    override func viewDidLoad() {
        super.viewDidLoad()

        let location = CLLocationCoordinate2DMake(51.546785, -0.179111)
        let mySpan = MKCoordinateSpanMake(0.003, 0.003)
        let region = MKCoordinateRegion(center: location, span: mySpan)

        myMap.setRegion(region, animated: true)

        let myAnnotation = MKPointAnnotation()
        myAnnotation.coordinate = location
        myAnnotation.title = "Bermotech Office"
        myMap.addAnnotation(myAnnotation)
    }
}
```

# Exercise



- Aim of exercise

- Change map type with button
- Clicking on the buttons will change the type of map displayed
- Add 3 buttons
  - ÷ Default
  - ÷ Satellite
  - ÷ Hybrid

```
myMap.mapType = .standard  
myMap.mapType = .satellite  
myMap.mapType = .hybrid
```

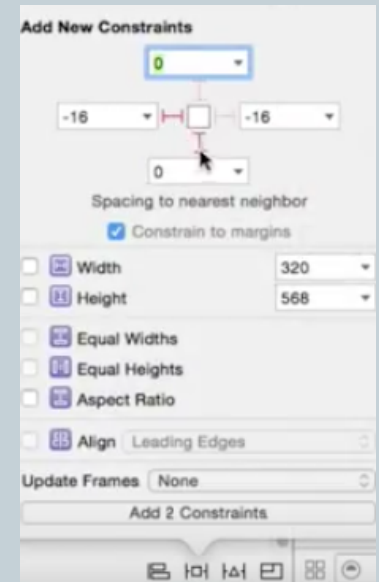
# Scroll View



# Scroll View Steps in Storyboard



- Storyboard
  - Add ScrollView to your added View
    - Add missing constraints
  - Add Label in Centre
    - Add missing constraints
  - Make ScrollView IBOutlet connection



# ScrollView ViewController Implementation



```
class ViewController: UIViewController
{

    var screenWidth : CGFloat!
    var screenHeight : CGFloat!
    @IBOutlet weak var myScrollView: UIScrollView!

    override func viewDidLoad()
    {
        super.viewDidLoad()

        screenWidth = self.view.frame.size.width
        screenHeight = self.view.frame.size.height
        print("\(screenWidth) , \(screenHeight)")

        myScrollView.contentInset = UIEdgeInsetsMake(0, 0, screenHeight, screenWidth)
    }
}
```

# WebView



# Web View Steps



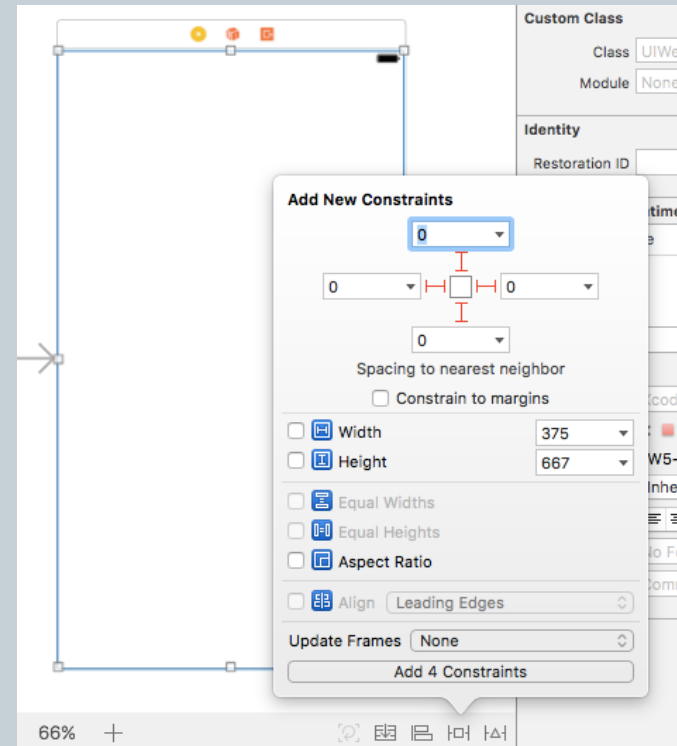
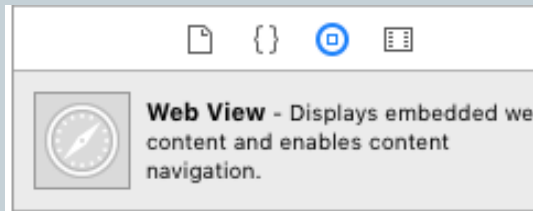
- Storyboard
  - Add WebView
    - Implement AutoLayout
      - Make IBOutlet Connection
- ViewController.swift
  - Load WebView URL request in ViewDidLoad
- Info.plist
  - Custom iOS Target Properties
    - App Transport Security Settings
      - Allow Arbitrary Load - set to YES



# Storyboard



- From the Object Library
  - Add a WebView to the ViewController
- Autolayout
  - Add the 4 constraints



# ViewController



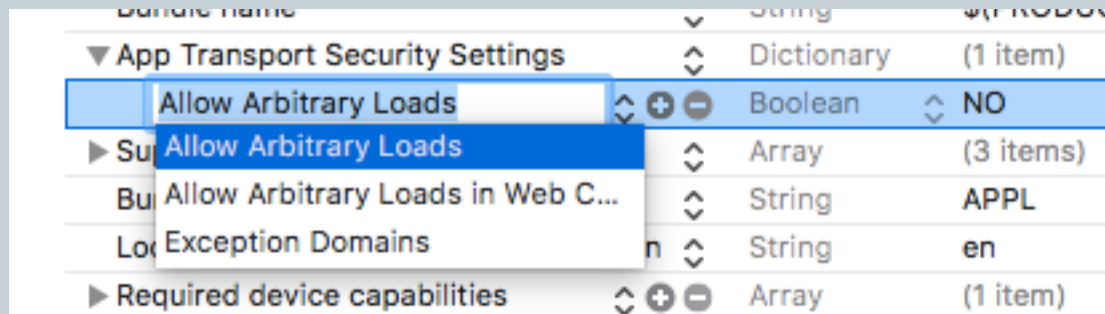
```
class ViewController: UIViewController
{
    @IBOutlet weak var myWebView: UIWebView!
    override func viewDidLoad()
    {
        super.viewDidLoad()

        let str1 = "http://www.google.com"
        let url1 = URL(string: str1)
        let req1 = URLRequest(url: url1!)
        myWebView.loadRequest(req1)
    }
}
```

# Info.plist



- Add row
  - *App Transport Security Settings*
- Expand *App Transport Security Settings*
  - Click on +
    - Add Allow Arbitrary Loads
      - Set to YES



# Info.plist



## ▼ Custom iOS Target Properties

Key		Type	Value
Bundle versions string, short	↕	String	1.0
Bundle identifier	↕	String	\$(PRODUCT_BUNDLE_IDENTI
InfoDictionary version	↕	String	6.0
Main storyboard file base name	↕	String	Main
Bundle version	↕	String	1
Launch screen interface file base name	↕	String	LaunchScreen
Executable file	↕	String	\$(EXECUTABLE_NAME)
Application requires iPhone environm...	↕	Boolean	YES ↕
Bundle name	↕	String	\$(PRODUCT_NAME)
▶ Supported interface orientations	↕	Array	(4 items)
▼ App Transport Security Settings	↕	Dictionary	(1 item)
Allow Arbitrary Loads	↕	Boolean	YES ↕
Bundle OS Type code	↕	String	APPL
Localization native development region	↕	String	en ↕
▶ Required device capabilities	↕	Array	(1 item)

# Localisation



# Steps

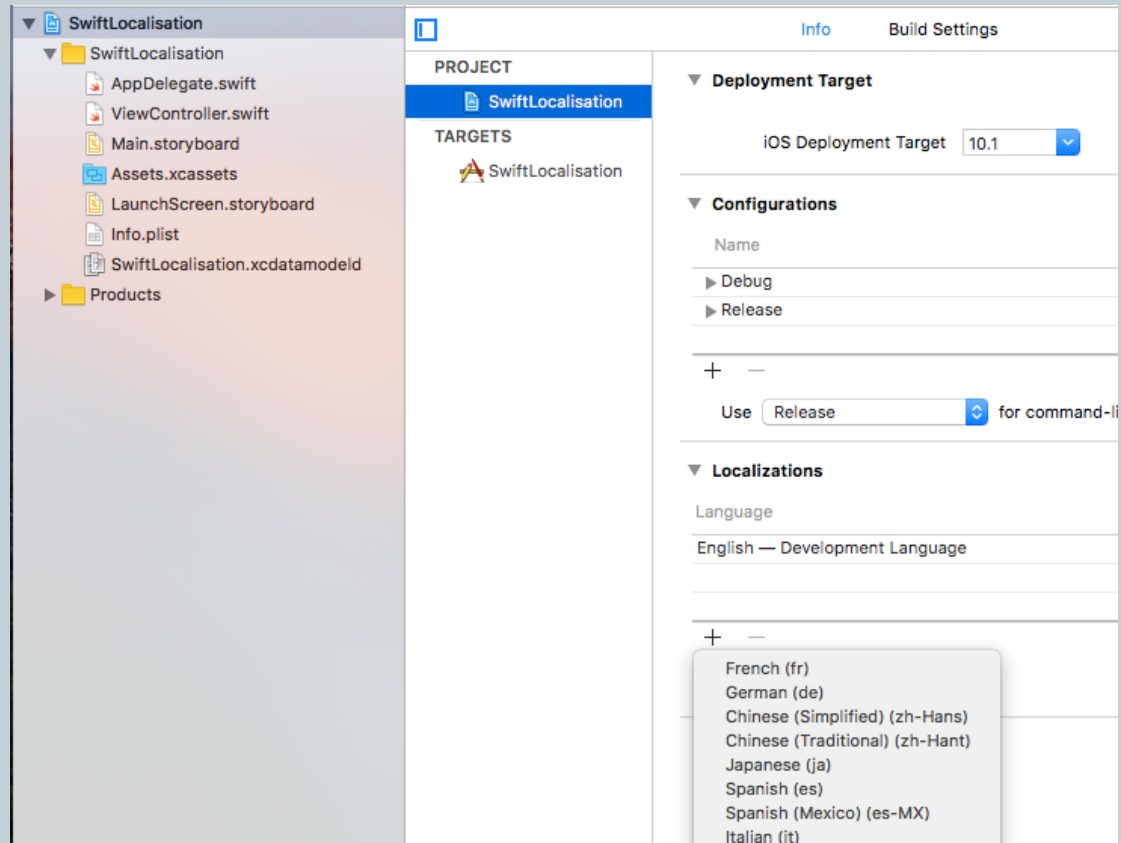


- Create New SingleView Project
- Add labels to ViewController
- Make IBOutletConnections

# Add Localization



- Add language under localisation



# Localization added

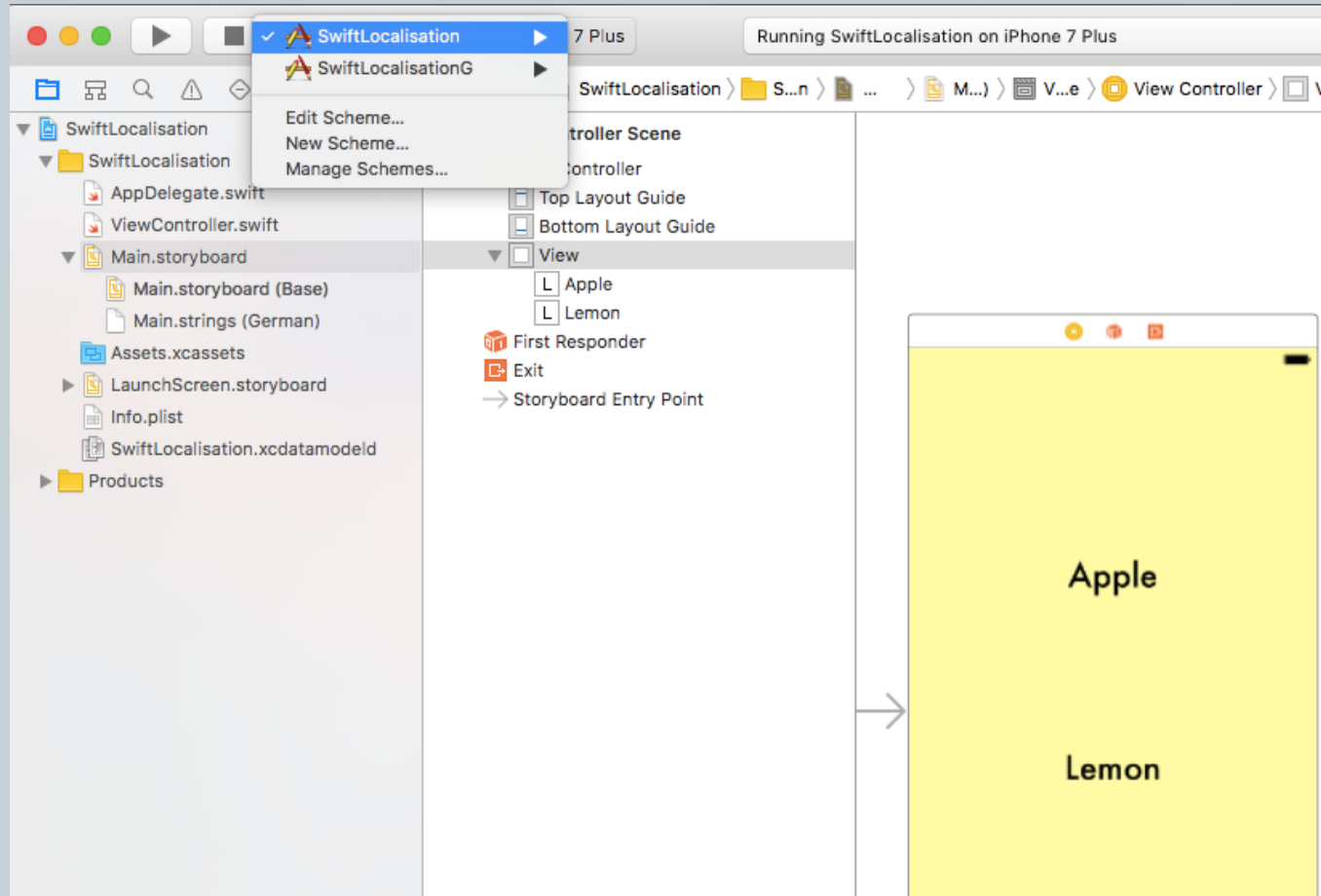


## ▼ Localizations

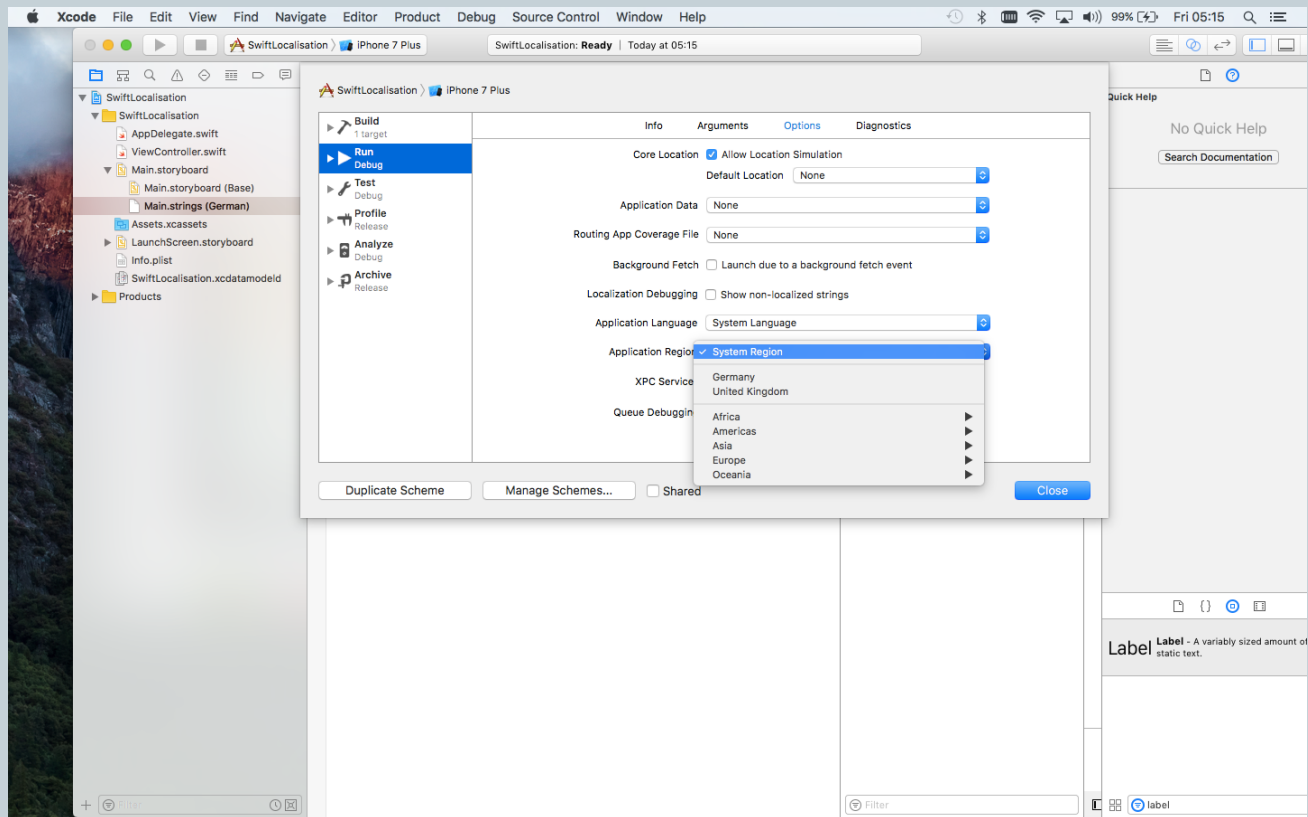
Language	Resources
English — Development Language	2 Files Lo...
German	2 Files Lo...



# Edit Scheme




# Change Application Language



# Application Region







- Can also change Application Region
- This will not change the language!!

Application Language	<input type="text" value="German"/>	
Application Region	<input type="text" value="Germany"/>	

# Picker View



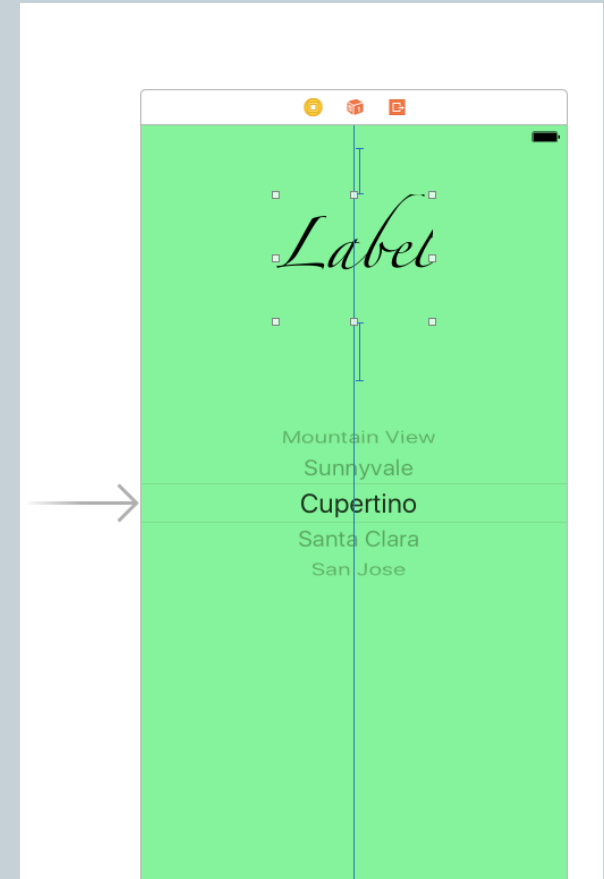
Placeholder image for the Picker View component, showing a small square with placeholder text.

**Picker View** - Displays a spinning-wheel or slot-machine motif of values.

# PickerView Steps




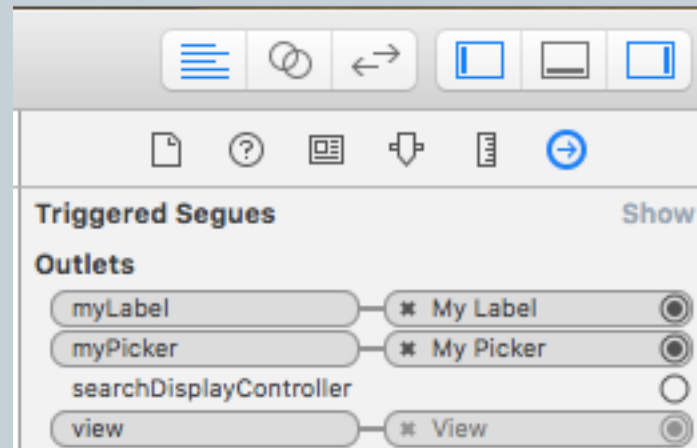
- Storyboard:
  - From Object Library add:
    - Label
    - UIPickerView
  - Complete AutoLayout
  - Make IBOutlet Connections for
    - UIPickerView and Label



# PickerView Steps



- Click on ViewController on Storyboard
- Click on Connections Inspector 
- Connect Label and PickerView to ViewController
  - ViewController becomes the ***delegate***



# ViewController Class



- Protocol Implementations:
  - UIPickerViewDelegate
  - UIPickerViewDataSource

```
class ViewController: UIViewController, UIPickerViewDelegate, UIPickerViewDataSource
```

# ViewController Class



- IBOutlet Connection:

```
@IBOutlet weak var myLabel: UILabel!  
@IBOutlet weak var myPicker: UIPickerView!
```



# ViewController Class



- Array declaration and initialisation
- The array will hold all the font types
  - Font types will be displayed in the picker list
  - Font types will also be used to set Label font type

```
var myPickerArray = ["Futura", "Arial",  
"Courier", "Chalkduster", "Zapfino",  
"AppleColorEmoji", "Didot", "MarkerFelt-Thin",  
"Papyrus", "Symbol"]
```

# ViewController Class



- viewDidLoad Implementation:

```
override func viewDidLoad() {  
    super.viewDidLoad()  
  
    myPicker.dataSource = self  
    myPicker.delegate = self  
  
    myLabel.text = myPickerArray[0]  
    myLabel.font = UIFont(name: myPickerArray[0], size: 40)  
}
```

# ViewController Class

195

- UIPickerView method Implementations:

```
func numberOfComponents(in pickerView: UIPickerView) -> Int
{
    return 1
}
```

```
func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) -> String?
{
    return myPickerArray[row]
}
```

```
func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int)
{
    myLabel.text = myPickerArray[row]
    myLabel.font = UIFont(name: myPickerArray[row], size: 40)
}
```

```
func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int
{
    return myPickerArray.count
}
```

# pickerView numberOfRowsInComponent



- Write method implementation for:
  - pickerView numberOfRowsInComponent

```
return myPickerArray.count
```

# numberOfComponentsInPickerView



- Write method implementation for:
  - numberOfComponentsInPickerView

```
return 1
```

# pickerView didSelectRow inComponent



- Write method implementation for:

- pickerView didSelectRow inComponent

```
myLabel.text = myPickerArray[row]  
myLabel.font = UIFont(name: myPickerArray[row], size: 40)
```

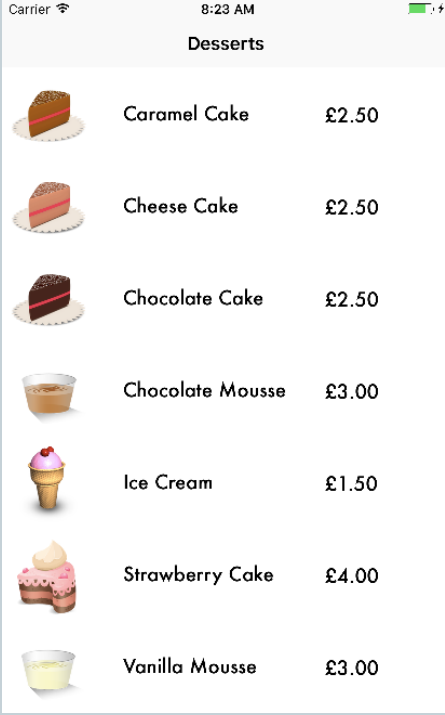
# pickerView titleForRow forComponent










- Write method implementation for:
  - pickerView titleForRow forComponent

```
return myPickerArray[row]
```

# TableView



A mobile app mockup displaying a TableView of desserts. The status bar at the top shows 'Carrier', signal strength, '8:23 AM', and battery level. The title bar is labeled 'Desserts'. The table lists seven items, each with an emoji icon, the item name, and the price.

Desserts		
	Caramel Cake	£2.50
	Cheese Cake	£2.50
	Chocolate Cake	£2.50
	Chocolate Mousse	£3.00
	Ice Cream	£1.50
	Strawberry Cake	£4.00
	Vanilla Mousse	£3.00



# Steps



- Storyboard
- Implementation:
  1. Initialise the arrays
  2. Add number of sections in
    - 1. numberOfSectionsInTableView method:
  4. Return number of rows in
    - numberOfRowsInSection method
    - Configure the cell in

# Steps



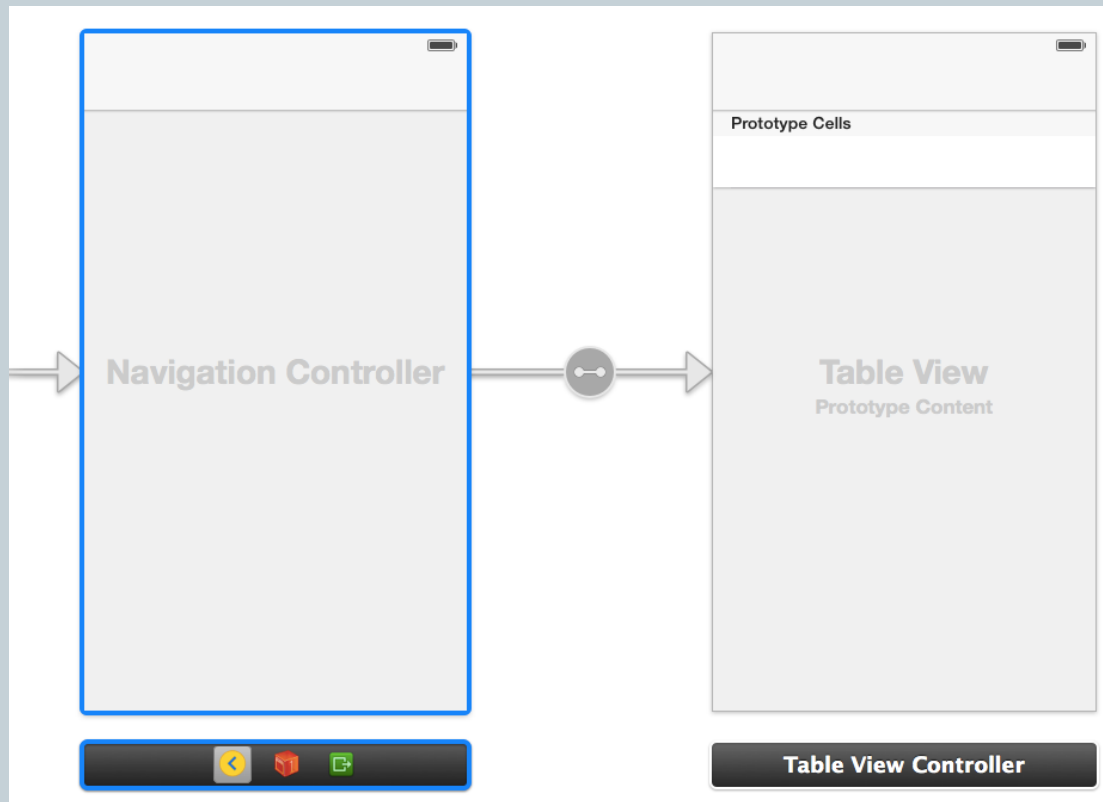
- **Storyboard**

- Delete the existing View Controller
- From the Objects Menu drag a “Table View Controller”
- Editor
  - ÷ Embed In
    - Navigation Controller

# Storyboard



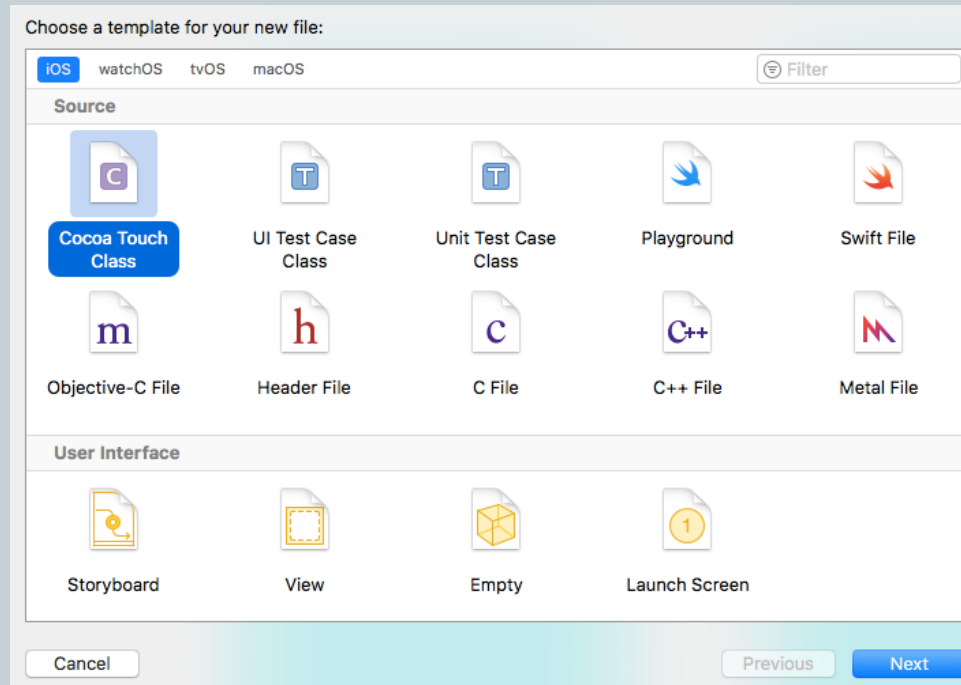
- Storyboard should appear as follows



# Create TableViewController Class



- Next need to create a Table View Controller Class.
- File → New → File
  - iOS – Cocoa Touch Class – Next



# Create TableViewController Class



- Enter Class Name
  - Subclass of “UITableViewController”
    - Next → Create

Choose options for your new file:

Class:

Subclass of:  ▼

☐ Also create XIB file

Language:  ▼

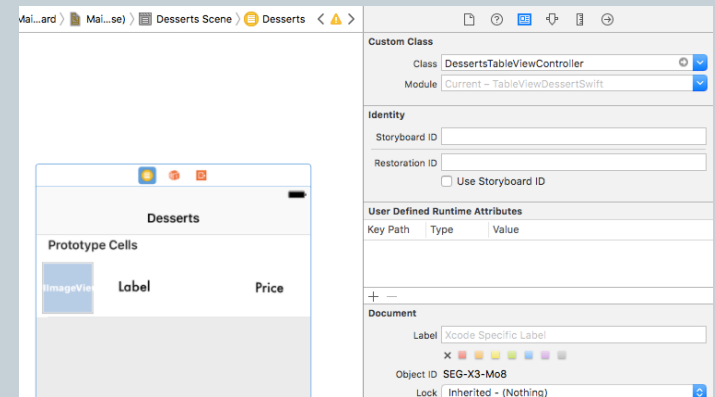
Cancel Previous Next

# Assign TableViewController Class

- Assign the created TableViewController class

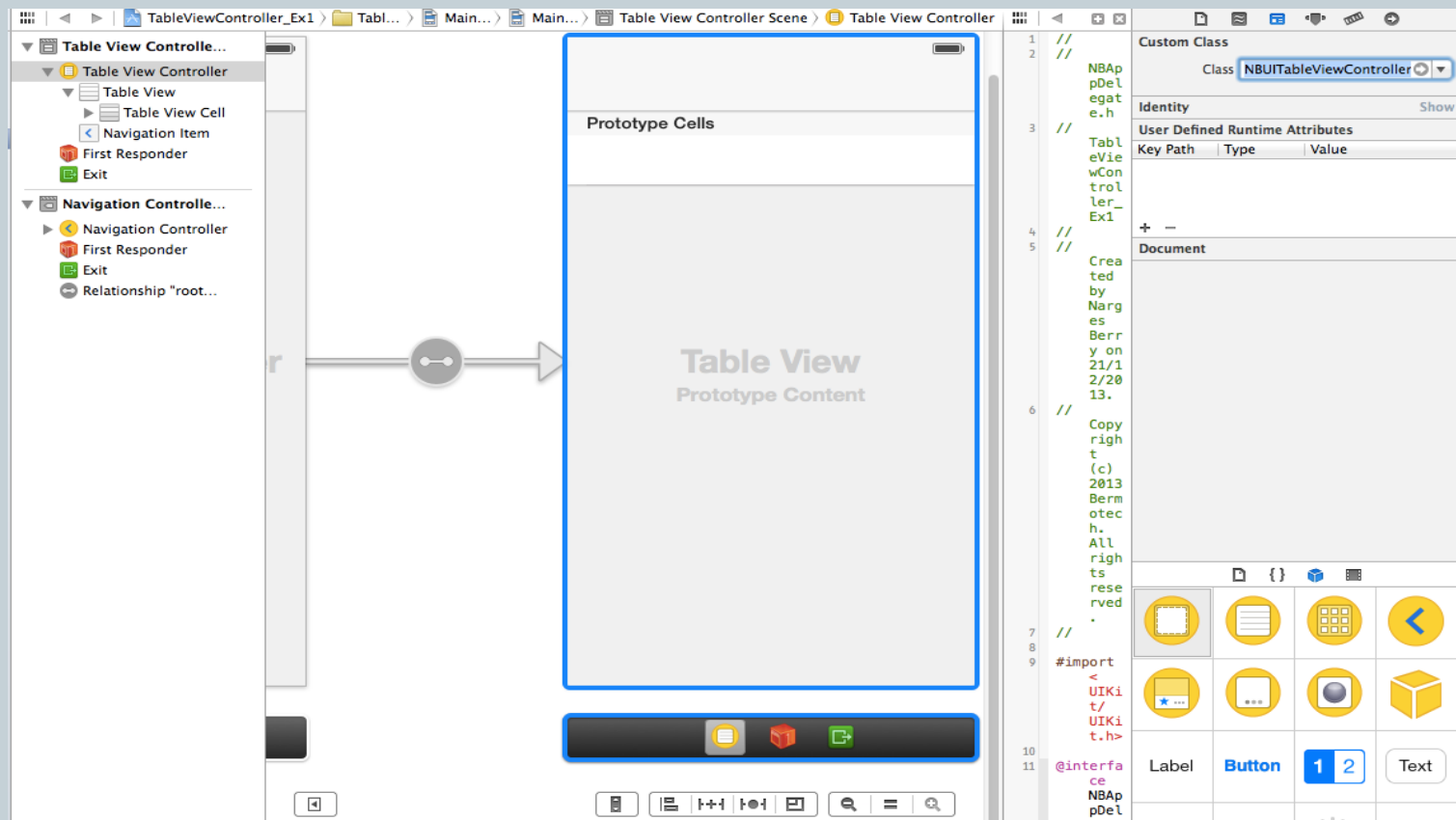
- In storyboard

- → MainStoryboard.storyboard
- → Show “Document Outline”
- → Click on the Table View Controller
- → Identity Inspector
  - → Custom Class → Select the correct “Class”  
÷ (the one just created) from the drop-down



# Assign TableViewController Class

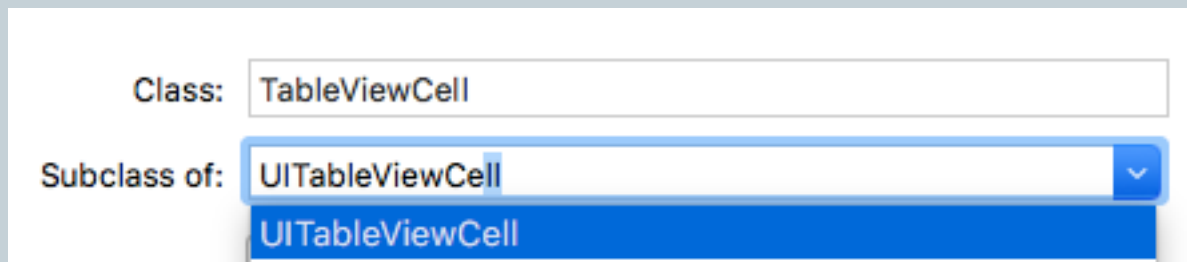
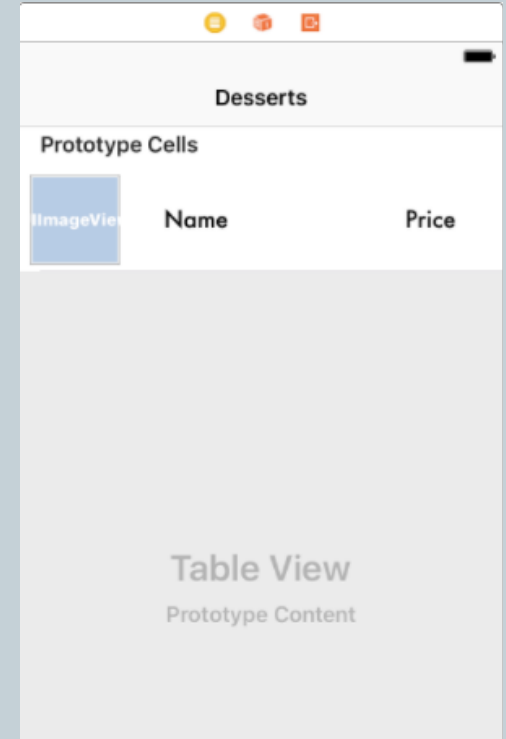
- TableViewController Class assigned



# TableViewCell



- Storyboard
  - Add an UIImageView and two Labels
- Create a TableViewCell Class
  - File - New - File
    - iOS - Cocoa Touch Class - Next
    - Subclass of **UITableViewCell**



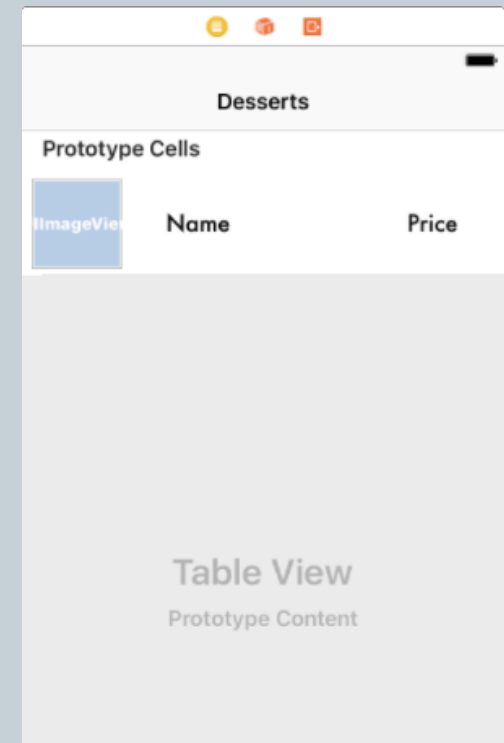


# TableViewCell



- Assign the created class to cell through ID Inspector
- Make IBOutlet connections for the image and 2 labels

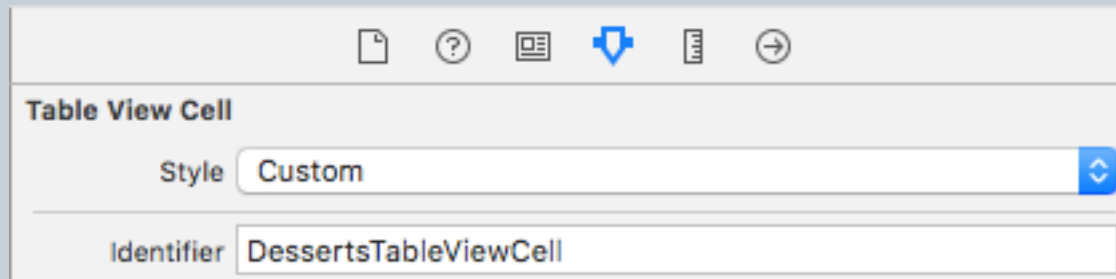
```
class DessertsTableViewCell: UITableViewCell {  
    @IBOutlet weak var myImage: UIImageView!  
    @IBOutlet weak var myLabel: UILabel!  
    @IBOutlet weak var myPriceLabel: UILabel!
```



# Set the “Cell Identifier”

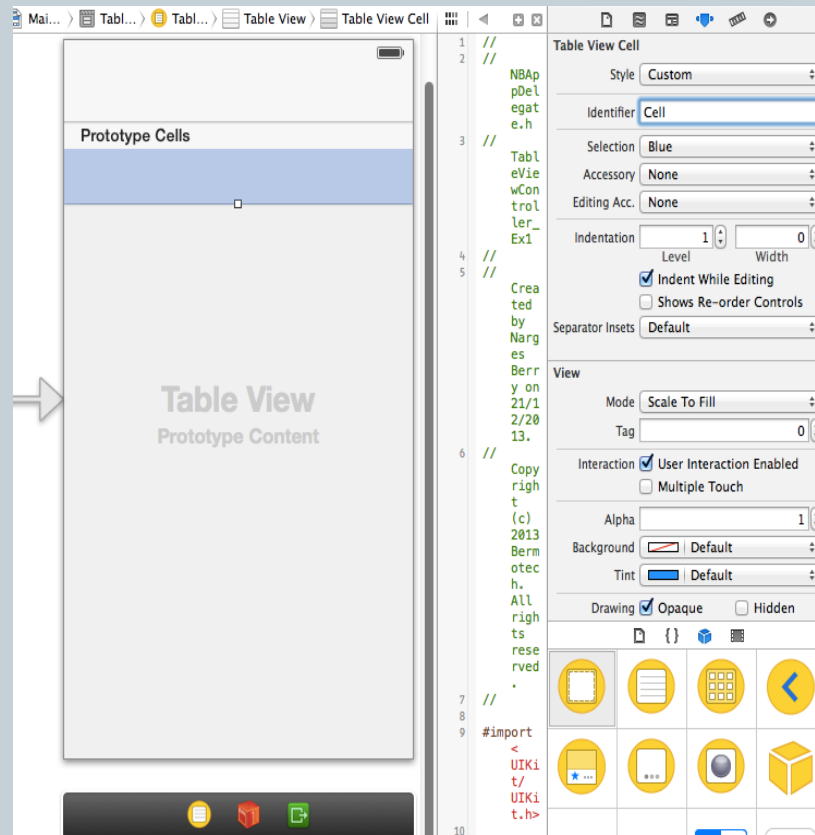


- Set the “Cell Identifier” to **DessertsTableViewCell**
- → Click on the Prototype Cell
- → Attribute Inspector
- → Table View Cell
- → Identifier
- → Set to **DessertsTableViewCell**



# Set the “Cell Identifier”

- “Cell Identifier” set



# TableViewController Class

158

- Initialise Arrays:

```
var desserts =  
["Caramel Cake", "Cheese Cake", "Chocolate Cake", "Chocolate  
Mousse", "Ice Cream", "Strawberry Cake", "Vanilla Mousse"]
```

```
var images =  
["CaramelCake.png", "CheeseCake.png", "ChocolateCake.png", "ChocolateMou  
sse.png", "IceCream.png", "StrawberryCake.png", "VanillaMousse.png"]
```

```
var prices =  
["£2.50", "£2.50", "£2.50", "£3.00", "£1.50", "£4.00", "£3.00"]
```

# TableViewController Class



- Add number of sections in
  - numberOfSectionsInTableView method:

```
override func numberOfSectionsInTableView(tableView: UITableView) -> Int
{
    return 1
}
```

# TableViewController Class



- Return number of rows
  - which is equal to the number items in the array

```
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int
{
    return desserts.count
}
```

# TableViewController Class



- Configure the cell in the following method to display the array item corresponding to the row:

```
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell
{
    let myCell = tableView.dequeueReusableCell(withIdentifier: "DessertsTableViewCell") as!
    DessertsTableViewCell

    let row = indexPath.row

    myCell.myLabel.text = desserts[row]

    myCell.myImage.image = UIImage(named: images[row])

    myCell.myPriceLabel.text = prices[row]

    return myCell
}
```

# TableView with DetailView





# DetailedViewController



- When user clicks on the table row
  - The DetailedViewController will appear
- Steps:
  1. Add a new ViewController to the Storyboard
  2. Add Image and Label to this ViewController
  3. Create a new DetailViewController Class
    - Subclass of UIViewController
      - `class DetailViewController: UIViewController`
  4. Connect the class through ID Inspector
  5. Make IBOutlet connections for Image and Label
    - `@IBOutlet weak var myLabel: UILabel!`
    - `@IBOutlet var ivDisplayImage : UIImageView?`

# cellForRowAtIndexPath

158

```
func tableView(_ tableView: UITableView, cellForRowAtIndexPath indexPath: IndexPath) -> UITableViewCell
{
    var cell : SampleTableViewCell! = tableView.dequeueReusableCell(withIdentifier: "Cell") as!
SampleTableViewCell
    if(cell == nil)
    {
        cell = Bundle.main.loadNibNamed("Cell", owner: self, options: nil)?[0] as! SampleTableViewCell;
    }
    let stringTitle = carName[indexPath.row] as String //NOT NSString
    let strCarName = car[indexPath.row] as String
    cell.lblTitle.text=stringTitle
    cell.ivPhoto.image = UIImage(named: strCarName)
    return cell as SampleTableViewCell
}
```

# prepare for segue

159

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?)
{
    if segue.identifier == "DetailSegue"
    {
        let detailViewController = ((segue.destination) as! DetailViewController)
        let indexPath = self.tvCars!.indexPathForSelectedRow!
        let strImageName = car[indexPath.row]
        let strCarName = carName[indexPath.row]
        detailViewController.strImageName = strImageName
        detailViewController.strCarName = strCarName
        detailViewController.title = strImageName
    }
}
```

# DetailedViewController Implementation

160



```
import UIKit

class DetailViewController: UIViewController
{
    @IBOutlet weak var myLabel: UILabel!
    @IBOutlet var ivDisplayImage : UIImageView?

    var strImageName: String!
    var strCarName: String!

    override func viewDidLoad()
    {
        super.viewDidLoad()
        myLabel.text = strCarName
        self.ivDisplayImage?.image=UIImage(named : strImageName)
    }

    @IBAction func btnBackClicked(_ sender : AnyObject)
    {
        self.dismiss(animated: true, completion: nil)
    }
}
```

# Collection View



## Introduced in iOS 6



**Collection View** - Displays data in a collection of cells.

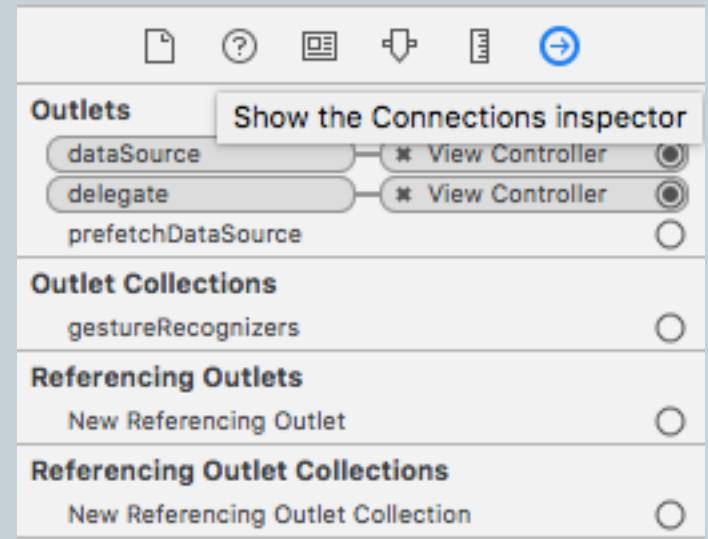


**Collection View Cell** - Defines the attributes and behavior of cells in a collection view.

# CollectionView Steps



- Storyboard:
  - Add CollectionView to ViewController
  - Add Collection View Cell to CollectionView
  - Add UIImageView and Label to the Cell
  - Click on ViewController
    - Go to Connection Inspector
      - Connect:
        - Label and UIImageView



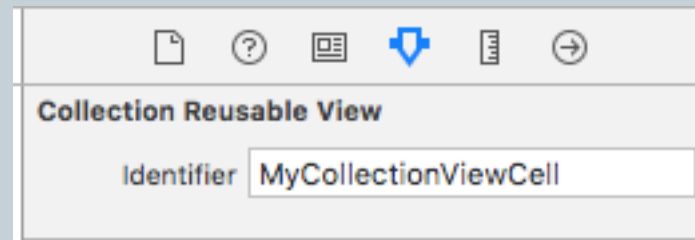
# CollectionView Cell



- Extends:  
UICollectionViewCell

```
class MyCollectionViewCell: UICollectionViewCell {  
  
    @IBOutlet weak var cellLabel: UILabel!  
    @IBOutlet weak var cellImage: UIImageView!  
}
```

- ID set in Storyboard



# ViewController Implementation



```
class ViewController: UIViewController, UICollectionViewDelegate, UICollectionViewDataSource {

    let dessertsArray = ["CaramelCake", "CheeseCake", "ChocolateCake", "MousseCake"]
    let dessertsLabelArray = ["Caramel Cake", "Cheesecake", "Chocolate Cake", "Chocolate Mousse"]

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

    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
        return dessertsArray.count
    }

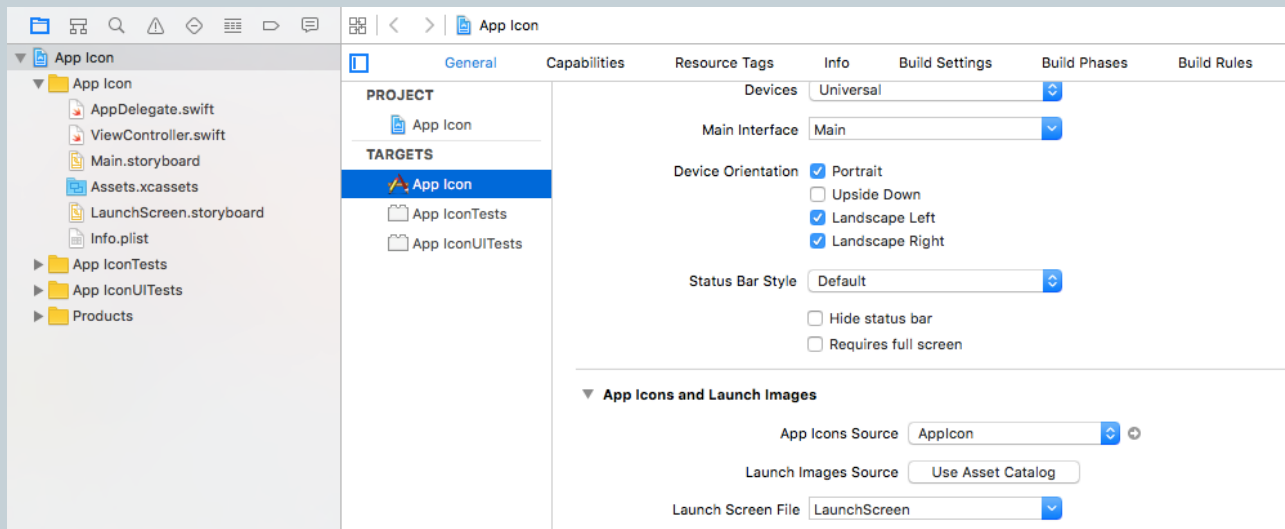
    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {
        let cell : MyCollectionViewCell = collectionView.dequeueReusableCell(withReuseIdentifier: "MyCollectionViewCell", for:indexPath) as! MyCollectionViewCell

        cell.cellImage.image = UIImage(named:dessertsArray[indexPath.row] )
        cell.cellLabel.text = dessertsLabelArray[indexPath.row]

        return cell
    }
}
```



# App Icon



# App Icon Sizes



- 40
- 60
- 58
- 87
- 80
- 120
- 120
- 180

## Appicon



iPhone  
Spotlight - iOS 5,6  
Settings - iOS 5-9  
29pt

iPhone Spotlight  
iOS 7-9  
40pt



iPhone App  
iOS 7-9  
60pt



iPad Settings  
iOS 5-9  
29pt

iPad Spotlight  
iOS 7-9  
40pt



iPad App



iPad Pro App



- <https://www.cocoacontrols.com/controls?language=2-swift>