GENERAL ASSEMBLY

MOBILE DEVELOPMENT

INTRO TO SWIFT

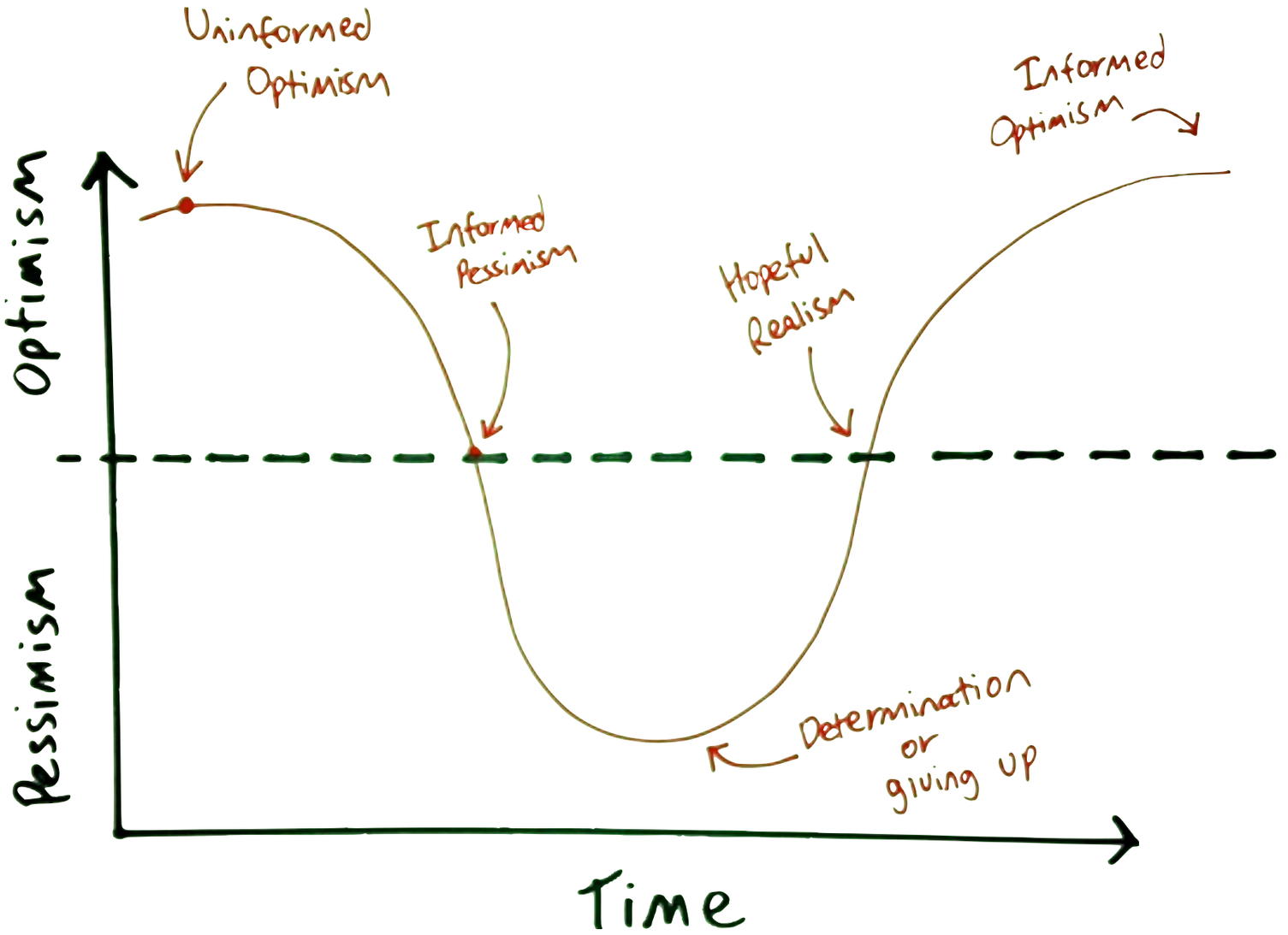
Kishin Manglani

ADMIN

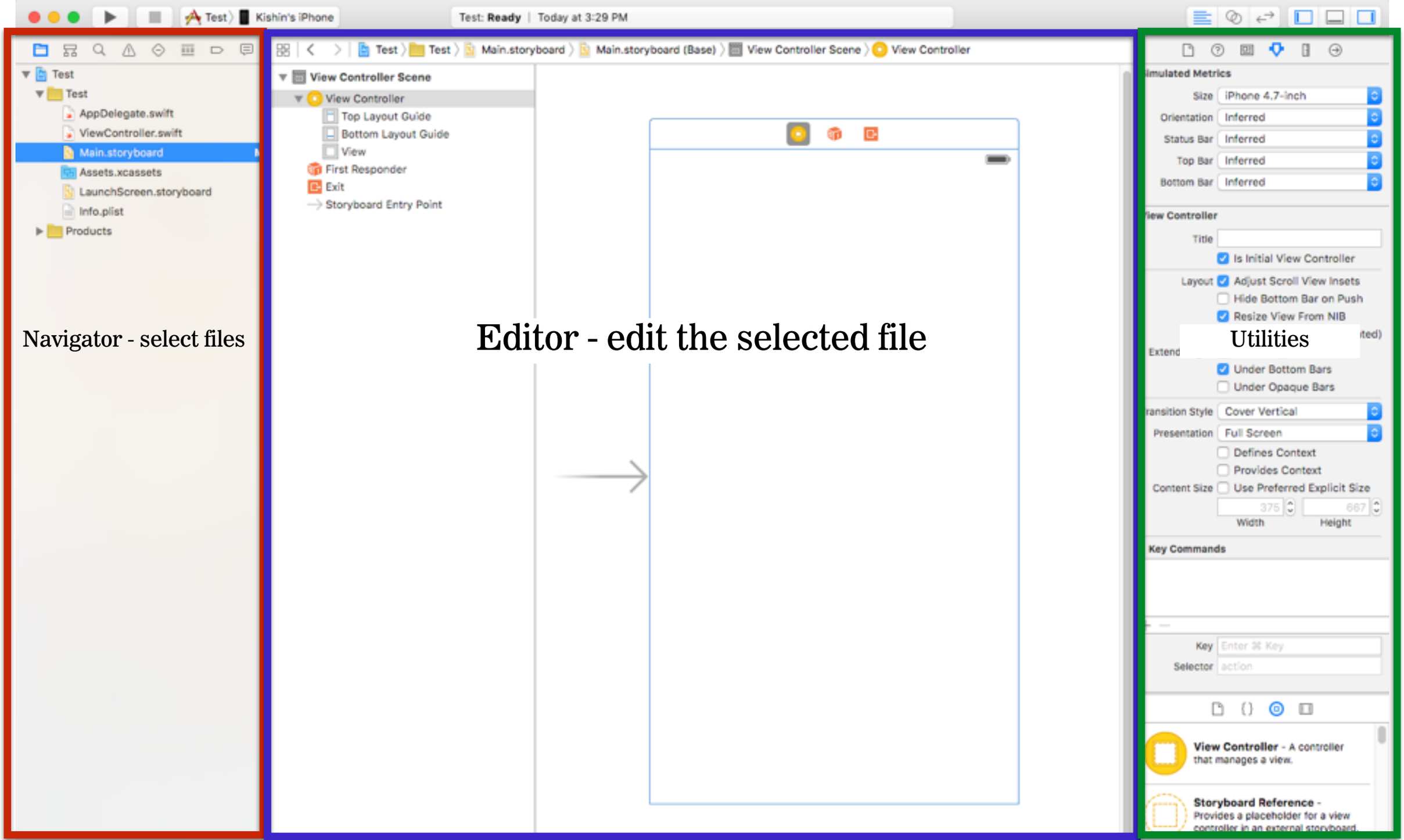
EMAIL HW BY 10/14 @ 6:29 PM

kishin.manglani@gmail.com

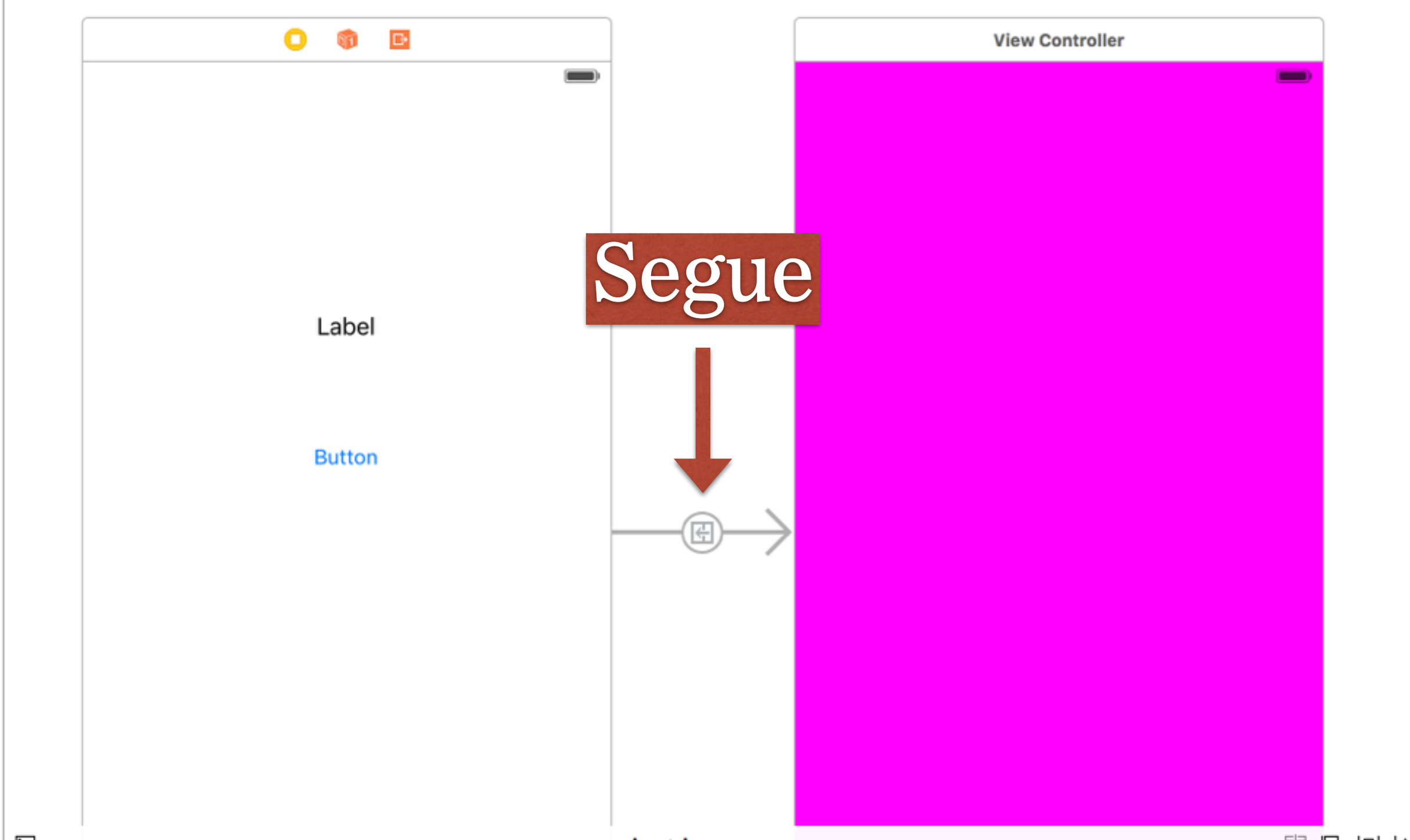
EMOTIONAL CYCLE OF CHANGE

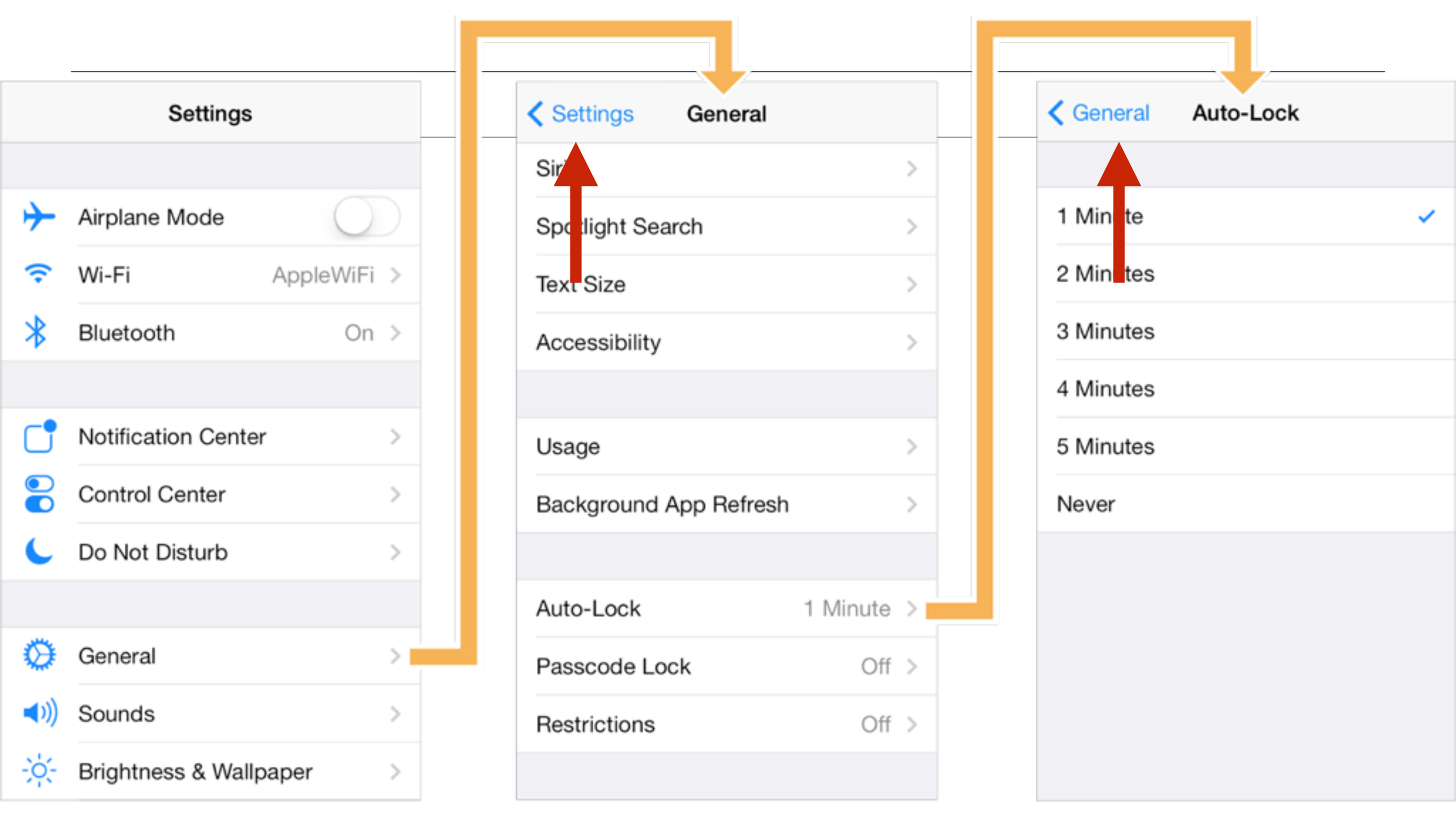


RECAP


Test
Kishin's iPhone
Test: Ready | Today at 3:29 PM
Test
Test
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
First Responder
Exit
Storyboard Entry Point
Navigator - select files
Editor - edit the selected file
Utilities
Simulated Metrics
Size
iPhone 4.7-inch
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Inferred
View Controller
Title
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Full Screen
Defines Context
Provides Context
Content Size
Use Preferred Explicit Size
375
667
Width
Height
Key Commands
Key
Enter ⌘ Key
Selector
action
View Controller - A controller that manages a view.
Storyboard Reference - Provides a placeholder for a view controller in an external storyboard.

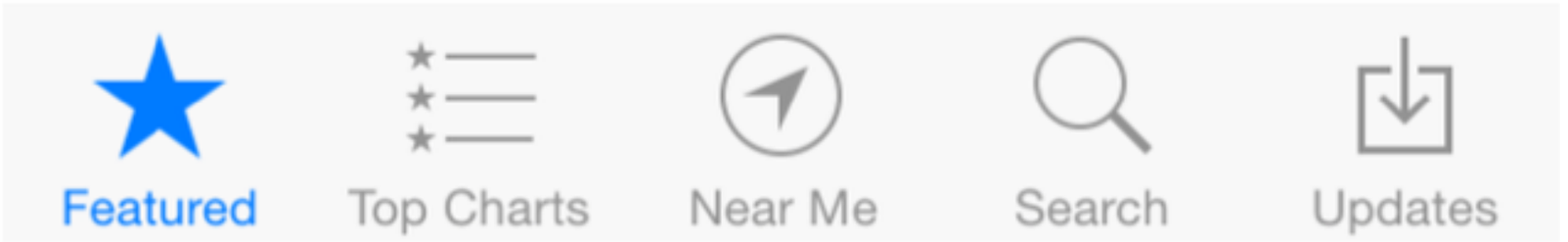
UIVIEW VS UIVIEWCONTROLLER

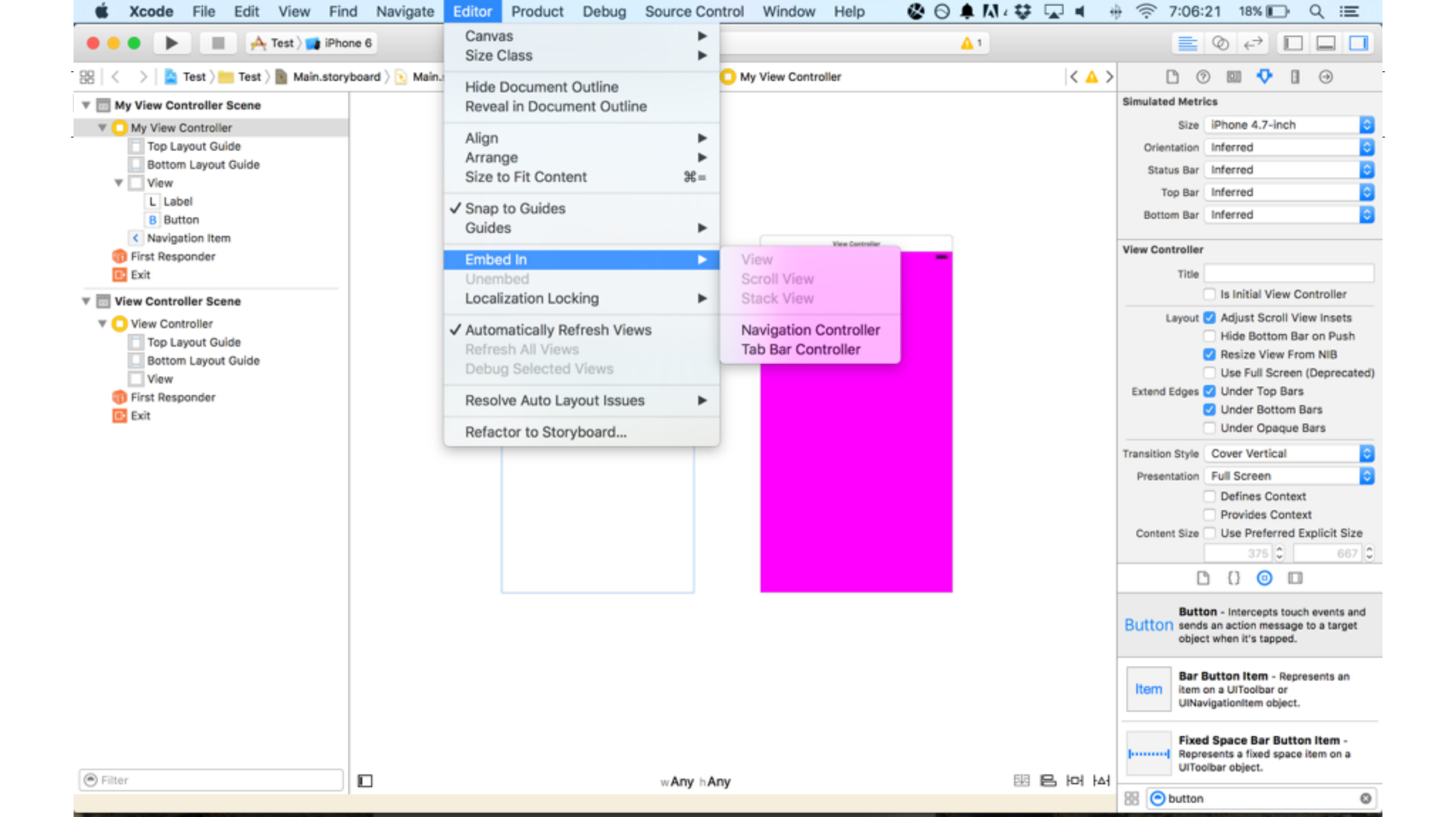

View Controller
Label
Button
Segue

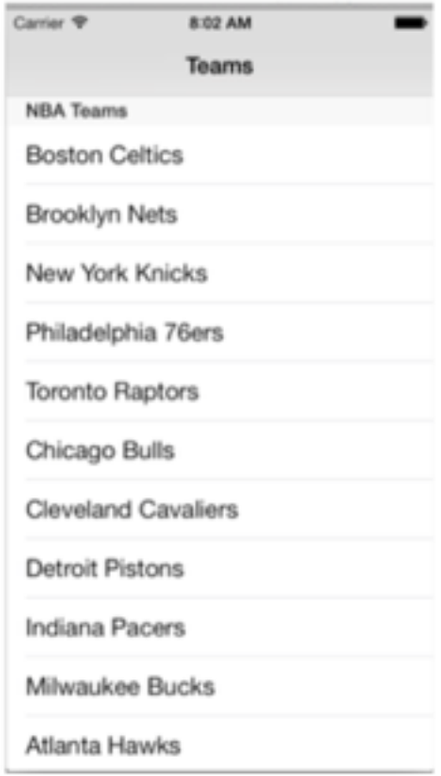

Settings
Airplane Mode
Wi-Fi
AppleWiFi
Bluetooth
On
Notification Center
Control Center
Do Not Disturb
General
Sounds
Brightness & Wallpaper
Settings
General
Siri
Spotlight Search
Text Size
Accessibility
Usage
Background App Refresh
Auto-Lock
1 Minute
Passcode Lock
Off
Restrictions
Off
General
Auto-Lock
1 Minute
2 Minutes
3 Minutes
4 Minutes
5 Minutes
Never

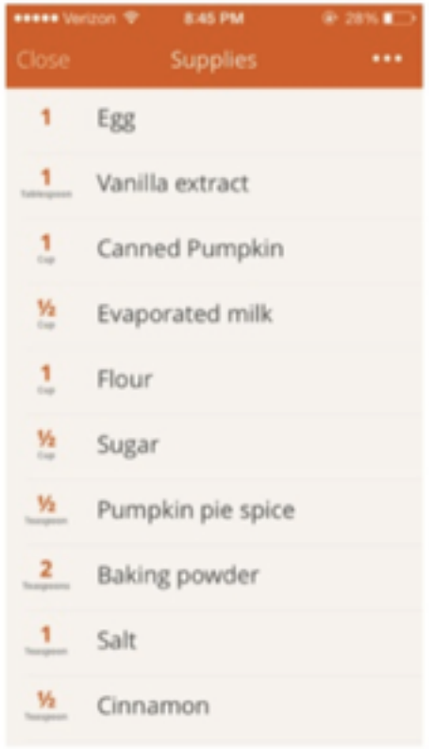
UITABBARCONTROLLER

- UITabBarController
- Provides tabs at the bottom of the screen
- Similar to what you see in the Facebook, Twitter, or Instagram apps
- Each tab represents a view controller
- UITabBar



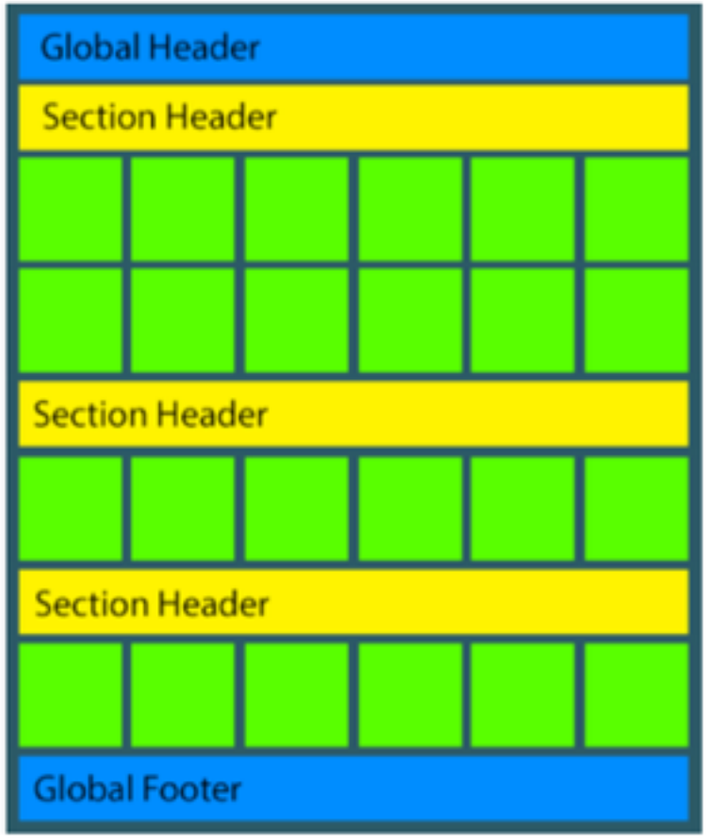

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Canvas
Size Class
Hide Document Outline
Reveal in Document Outline
Align
Arrange
Size to Fit Content
Snap to Guides
Guides
Embed In
Unembed
Localization Locking
Automatically Refresh Views
Refresh All Views
Debug Selected Views
Resolve Auto Layout Issues
Refactor to Storyboard...
View
Scroll View
Stack View
Navigation Controller
Tab Bar Controller
My View Controller Scene
My View Controller
Top Layout Guide
Bottom Layout Guide
View
Label
Button
Navigation Item
First Responder
Exit
View Controller Scene
View Controller
Simulated Metrics
Size
iPhone 4.7-inch
Orientation
Status Bar
Top Bar
Bottom Bar
Inferred
View Controller
Title
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Full Screen
Defines Context
Provides Context
Content Size
Use Preferred Explicit Size
Button
Button - Intercepts touch events and sends an action message to a target object when it's tapped.
Item
Bar Button Item - Represents an item on a UIToolbar or UINavigationItem object.
Fixed Space Bar Button Item - Represents a fixed space item on a UIToolbar object.
wAny hAny
button


Carrier
8:02 AM
Teams
NBA Teams
Boston Celtics
Brooklyn Nets
New York Knicks
Philadelphia 76ers
Toronto Raptors
Chicago Bulls
Cleveland Cavaliers
Detroit Pistons
Indiana Pacers
Milwaukee Bucks
Atlanta Hawks


Verizon
8:45 PM
Close
Supplies
1 Egg
1 Vanilla extract
1 Canned Pumpkin
½ Evaporated milk
1 Flour
½ Sugar
½ Pumpkin pie spice
2 Baking powder
1 Salt
½ Cinnamon


Verizon
12:09 PM
88%
iPhone
Collection View Default Gestures
Use Your Loaf · 21h
UIPopover arrow not repositioned correctly on rotation
Use Your Loaf · 2d
iOS 7 apps: more than just a face-lift?
Tapity · 2d
Working with SKTransition
Mobiletuts+ · 4d
Android SDK: Supporting Alternative Input Devices
Mobiletuts+ · 5d
Build an Airplane Game with


Verizon
10:07 PM
99%
Starchitects
by Airbnb
Share
Alsace Architectural modern hou...
Mulhouse, FR
$62
Stay in the "8 House" in Copenh...
Copenhagen, DK


Global Header
Section Header
Section Header
Section Header
Global Footer

INTRO TO SWIFT

PROGRAMMING LANGUAGES

- What is a programming language?
- It's really a way to define variables and constants (data) and perform logical manipulations of that data
- **Data & Logic** REMEMBER THIS
- Each programming language has Keywords, which have specified functionality
  - These keywords are reserved, and you cannot use them in your code

PROGRAMMING LANGUAGES

- Swift: 2014
- Constantly being updated
- Swift vs Objective-C

OBJECTIVES

- Learn the basics of Swift
- Understand what classes and objects are
- Bridge the gap between UIViews and UIViewControllers and code
- Blow your mind

OUTLINE

- Variables
- Constants
- Variable Types
- Print
- Mathematical and Comparison Operators
- If/Else Statements
- Logical Operators
- Functions
- Classes and Objects
- IBOutlet and IBAction

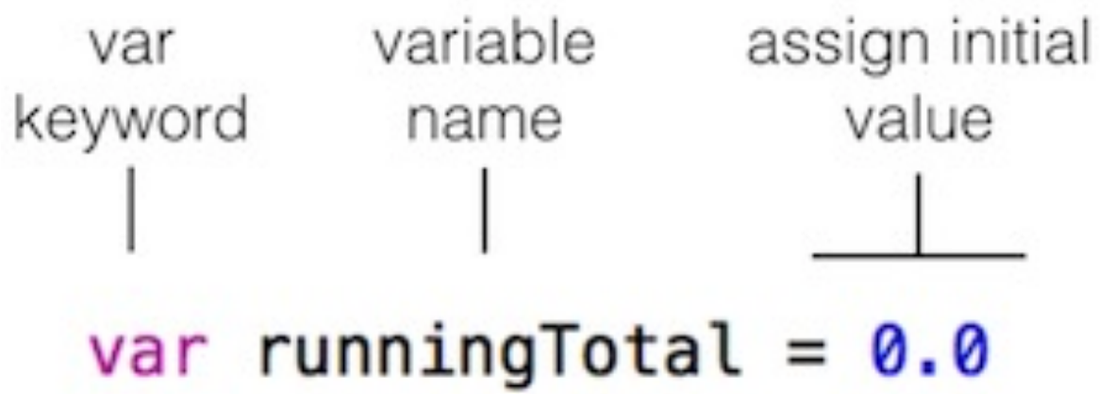
SWIFT!

INTRO TO SWIFT

VARIABLES & CONSTANTS

VARIABLES

- A named data container
- It can take on a value and then be changed at a later time
- Best practice**



VARIABLES

- If we declare a variable and set it's value at the same time, Swift will infer the type
- If we do NOT assign a value, we must specify the type

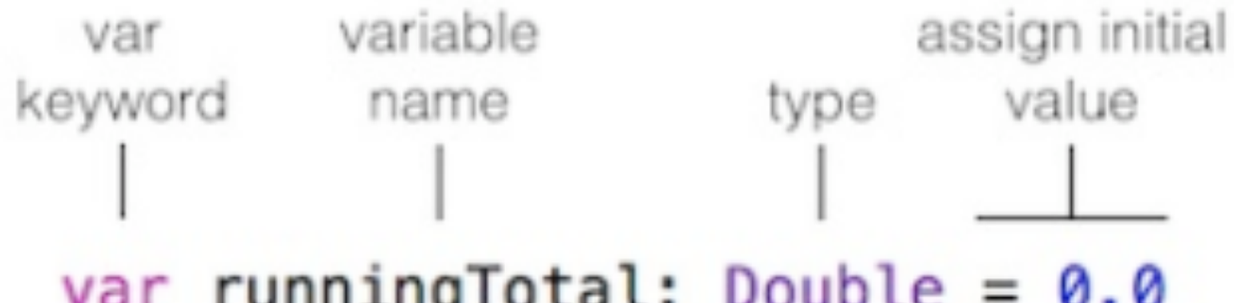
```
// variable
var name: String = "John Doe"
var temperature: Double = 36.5
var year: Int = 2014
var visible: Bool = true

year = 2015
```

VARIABLE TYPES

- Now what is the variable type?
- It's the kind of data this container is capable of holding
- Just as we (people) have special containers for holding different things, so does a computer
- For example, we have a different container to hold a bag of oranges and a gallon of milk. Each of these containers are specially designed to properly hold what it contains.
- Swift does something similar when we specify the TYPE of the variable

VARIABLE TYPES

- Variables an have different types of data
  - Int – whole numbers, or integers
  - Double – decimal numbers
  - Float – decimal numbers
  - Bool – a value that can be true, or false
  - String – a “string” of letters or words
  - These are the basic data types



NAMING VARIABLES

- What did we say earlier about naming variables?

CONSTANTS

- Sometimes we want to add data that doesn't change
- For example, the title of a page may always be "Settings"
- We can set that title in the Storyboard or we can set it in the view controller
- Instead of using var, we use let
- Once assigned, a constant cannot be changed

```
// constants
let name: String = "John Doe"
let temperature: Double = 36.5
let year: Int = 2014
let visible: Bool = true

name = "Steve Jobs"    // error
temperature = 36.8     // error
year = 2015            // error
visible = false        // error
```

STRING INTERPOLATION

- What does interpolation mean in English?
- “alter (a book or text) by insertion of new material”
- That’s exactly what it means in Swift, we take a String and insert some new material (in this case a variable, constant, or an “expression”)
- Backslash, above return/enter key

```
var numberOfApples = 5
var myString = "Sally has \(numberOfApples) apples"
```

"Sally has 5 apples"

COMMENTS

- Comments are not interpreted by Swift
- Good for adding human readable text, maybe a sentence explaining what something does
- Use // to denote a comment

```
//hey this is a comment
//so is this
var name = "Kishin"
```

INTRO TO SWIFT

OPERATORS & IF/ELSE

MATHEMATICAL OPERATORS

- You can perform mathematical operations on Swift variables
- You can also add strings!

```
var excitementLevelExplicit: Int = 100      100
var excitementLevel = 100                   100

excitementLevel = excitementLevel * 2       200
excitementLevel = excitementLevel + 20      220
excitementLevel = excitementLevel / 2       110
excitementLevel = excitementLevel - 15      95
```

COMPARISON OPERATORS

- So a Bool represents either a true or false value
- When would we use a Bool? Maybe we want to represent a preference a user has set, or we could even create a Bool from doing a comparison
- > (Greater Than)
- < (Less Than)
- == (Equal To)
- != (Not Equal To)
- >= (Greater Than or Equal To)
- <= (Less Than or Equal To)

COMPARISON OPERATORS

- So here's an example
- We declare two variables and check if they are equal to each other and then if experiencePoints is greater than opponentsExperiencePoints
- We can see on the right they return true or false

```
var experiencePoints = 100                          100
var opponentsExperiencePoints = 90                  90

experiencePoints == opponentsExperiencePoints       false
experiencePoints > opponentsExperiencePoints        true
```

IF/ELSE STATEMENTS

- If/else statements allow code to be executed based on some conditions
- The conditions should be a Bool; this is where the Bool really comes in handy
- For example, if I am hungry, then I will eat, else I will not eat
- If it is raining, take umbrella, else take sunglasses

```
var isRaining = true

if isRaining {
    print("it's raining")
}
else {
    print("it's not raining!")
}
```

```
if a < b {
    print("executed if")
}
else {
    print("executed else")
}
```

IF/ELSE STATEMENTS

```
var temperature = 90

if temperature > 212 {
    print("it's boiling")
}
else if temperature > 100 {
    print("it's sweltering")
}
else {
    print("it's not that bad")
}
```

- We can also add multiple conditions by adding an “else if”
- You can add as many of these as you want

LOGICAL OPERATORS

- What if we want to check multiple conditions in an if statement?
- We can use logical operators
- && (and)
- || (or)

```
if isHot && isRaining {
    print("it's hot, and it's raining")
}
else if isHot || isRaining {
    print("it's hot or it's raining")
}
else {
    print("it's not hot, and it's not raining")
}
```

PLAYGROUNDS

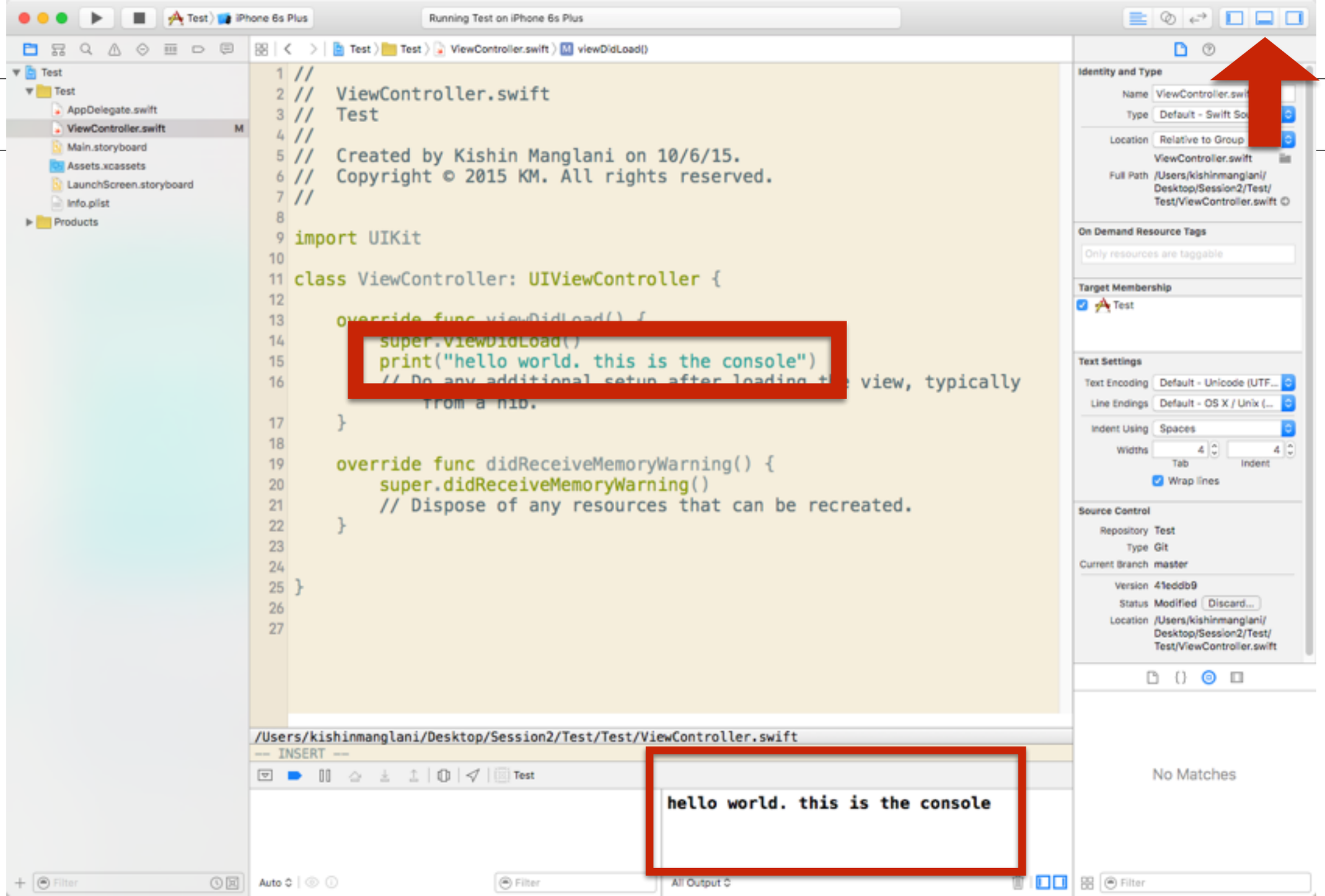
INTRO TO SWIFT

FUNCTIONS

PRINT

- print is a function
- print in Swift prints text to the debugging console
- This is not user facing. Can also be seen when a device is plugged into Xcode
- Remember to use double quotes “, not single quote ‘
- We can print specific text or even a variable
- **Best practice**: Remove all print statements before submitting your app to the app store

```
print("hello world")
print(name)
```


Test
iPhone 6s Plus
Running Test on iPhone 6s Plus
Test
Test
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
ViewController.swift
viewDidLoad()
//
// ViewController.swift
// Test
//
// Created by Kishin Manglani on 10/6/15.
// Copyright © 2015 KM. All rights reserved.
//
import UIKit
class ViewController: UIViewController {
override func viewDidLoad() {
super.viewDidLoad()
print("hello world. this is the console")
// Do any additional setup after loading the view, typically
from a nib.
}
override func didReceiveMemoryWarning() {
super.didReceiveMemoryWarning()
// Dispose of any resources that can be recreated.
}
}
Identity and Type
Name ViewController.swift
Type Default - Swift So
Location Relative to Group
ViewController.swift
Full Path /Users/kishinmanglani/Desktop/Session2/Test/Test/ViewController.swift
On Demand Resource Tags
Only resources are taggable
Target Membership
Test
Text Settings
Text Encoding Default - Unicode (UTF...
Line Endings Default - OS X / Unix (...
Indent Using Spaces
Widths 4 4
Tab Indent
Wrap lines
Source Control
Repository Test
Type Git
Current Branch master
Version 41eddb9
Status Modified Discard...
Location /Users/kishinmanglani/Desktop/Session2/Test/Test/ViewController.swift
/Users/kishinmanglani/Desktop/Session2/Test/Test/ViewController.swift
-- INSERT --
Test
hello world. this is the console
Auto
All Output
No Matches

FUNCTIONS

- Functions are named chunks of code
- How do we create our own functions?
- Functions can have inputs (arguments) and an output (return value) both of these are optional
- Use the `func` keyword to create a function

FUNCTIONS

```
func firstFunction() {
    print("This is our first function")
}
```

FUNCTIONS

```
func firstFunction() {
    print("This is our first function")
}

firstFunction()
```

FUNCTIONS

```
func printTax(amount: Double) {
    print(amount * 1.08)
}

printTax(2.00)
```

FUNCTIONS

```
func addTax() -> Double {
    print("Here we are")
    return 1.02
}

addTax()
```

FUNCTIONS

```
func addTax(amount: Double) -> Double {
    //If the amount is more than 100 don't add tax
    if amount > 100 {
        return amount
    }
    return amount * 1.08
}

addTax(2.00)
```

FUNCTIONS

```
func addTax(amount: Double) -> Double {
    return amount * 1.08
}

var totalAmount = addTax(10)
```

FUNCTIONS

- Why use functions instead of copying and pasting?
- DRY: Don’t Repeat Yourself
  - Think “lazy”
- Limit bugs and less typing when there are changes

INTRO TO SWIFT

OBJECTS AND CLASSES

OBJECTS

- An object holds or contains functions/methods and variables together
- Why would we want this? For convenience
- This allows us to repeat code that's already used
- For example, a string is an object. It is an object that Apple created and wrote for us. It too has variables/properties and functions/methods

```
var username: String = "Kishin"
username.uppercaseString
username.hasPrefix("Kish")
username.hasPrefix("John")
```

OBJECTS

- So basically, an object is just a set of variables and functions to deal with them
- But what exactly is an object? How does it get those variables and functions?
- There is a recipe or blueprint for an object
- That recipe/blueprint is called a class

CLASSES

- Classes allow us to create objects
- Classes allow us to create our own objects or data types
  - Int, Bool, String
- A class is a blueprint for an object
- Class is to Object as Blueprint is to Building
- Classes are abstract, objects and instances are concrete
- Classes contain functions and properties or variables

OBJECTS

- For example, let's create a Person class
  - We could create an object that had:
    - A name
    - An age
  - A function/method of Person could be eat. All Person's eat
- So Person is the class
- An object could be Kishin, Kishin is concrete

BRAINSTORM

CLASSES

```
class Hat {
    var color: String
    var size: Int

    init(newColor: String, newSize: Int) {
        self.color = newColor
        self.size = newSize
    }
}
```

CLASSES

```
var kishinsMetsHat = Hat(newColor: "Blue", newSize: 7)
```

CLASSES

```
var kishinsMetsHat = Hat(newColor: "Blue", newSize: 7)
print(kishinsMetsHat.color)
```

```
Hat
"Blue\n"
```

```
class Hat {
    var color: String
    var size: Int

    init(newColor: String, newSize: Int) {
        self.color = newColor
        self.size = newSize
    }

    func dyeBlack() {
        self.color = "Black"
    }

    func shrink() {
        if self.size > 1 {
            self.size = self.size - 1
        }
    }
}
```

CLASSES

```
func dyeBlack() {
    self.color = "Black"
}
```

```
var kishinsMetsHat = Hat(newColor: "Blue", newSize: 7)
print(kishinsMetsHat.color)
kishinsMetsHat.dyeBlack()
print(kishinsMetsHat.color)
```

```
Hat
"Blue\n"
Hat
"Black\n"
```

CLASSES

```
class Dog {
    var name: String
    var breed: String
    var age: Int

    init(name: String, breed: String, age: Int) {
        self.name = name
        self.breed = breed
        self.age = age
    }
}
```

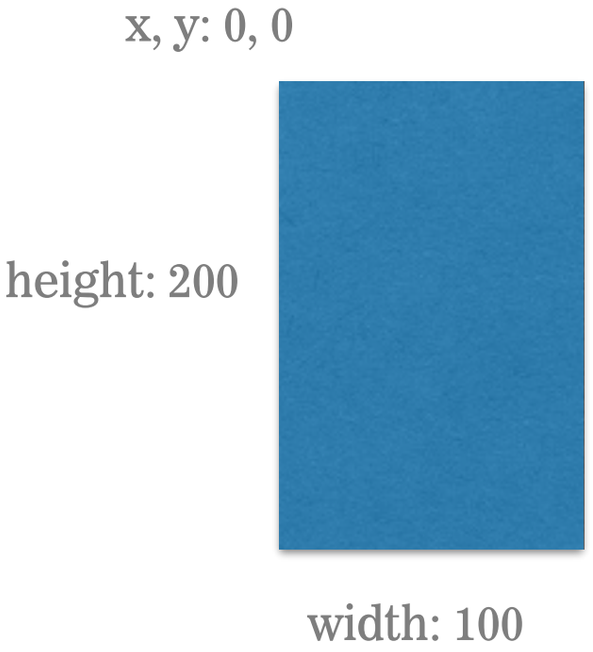
LET'S TRY IT

CLASSES

- This is the logic of our apps
- Each UIView we dragged and dropped in interface builder created an object from a class behind the scenes

REMEMBER UIKIT?

- Remember I said Apple wrote a bunch of code for us
- And that UIView was part of UIKit
- Apple just created a UIView class
- When you drag out a UIView (UILabel, etc), Storyboard sets it's properties and displays it, that's it
- It may do the equivalent:




```
var label = UILabel()
label.x = 20
label.y = 100
label.height = 200
label.width = 200
label.text = "Hello world"
```

REMEMBER UIKIT?

- Apple wrote all that code, so we can make beautiful apps
- They don’t want us to rewrite any of it
- They use a lot of special functions that only Apple can use, to make sure the phone is more secure

REMEMBER UIKIT?

- Apple even wrote code for UIViewController
- The UIViewController knows how to display it's view, so we can see "screens"
- It does a lot of special things and has that "view" property/variable
- Somewhere, something like this might exist

```
class UIViewController {
    var view: UIView
    //a whole lot more...
}
```

REMEMBER UIKIT?

- But remember, the UIViewController is the logic of our app, and we want to add custom logic to manipulate our data
- Remember, data and logic is what an app is all about
- If Apple already wrote the UIViewController class, do we need to rewrite it all again if we want to use the special functionality?

REMEMBER UIKIT?

- We can reuse all of Apple's code
- So what do I mean reuse? We can inherit from the UIViewController class
- What does inherit mean? We get all the "parent class's" variables and functions
- We add this colon and the other class we want to inherit from

```
class MyViewController: UIViewController {



}
```

REMEMBER UIKIT?

- We now magically have all those variables and methods
- To use the parent class method, we use the super keyword

```
class UIViewController {
    var view: UIView

    func viewDidLoad() {
        //do stuff
    }
}
```

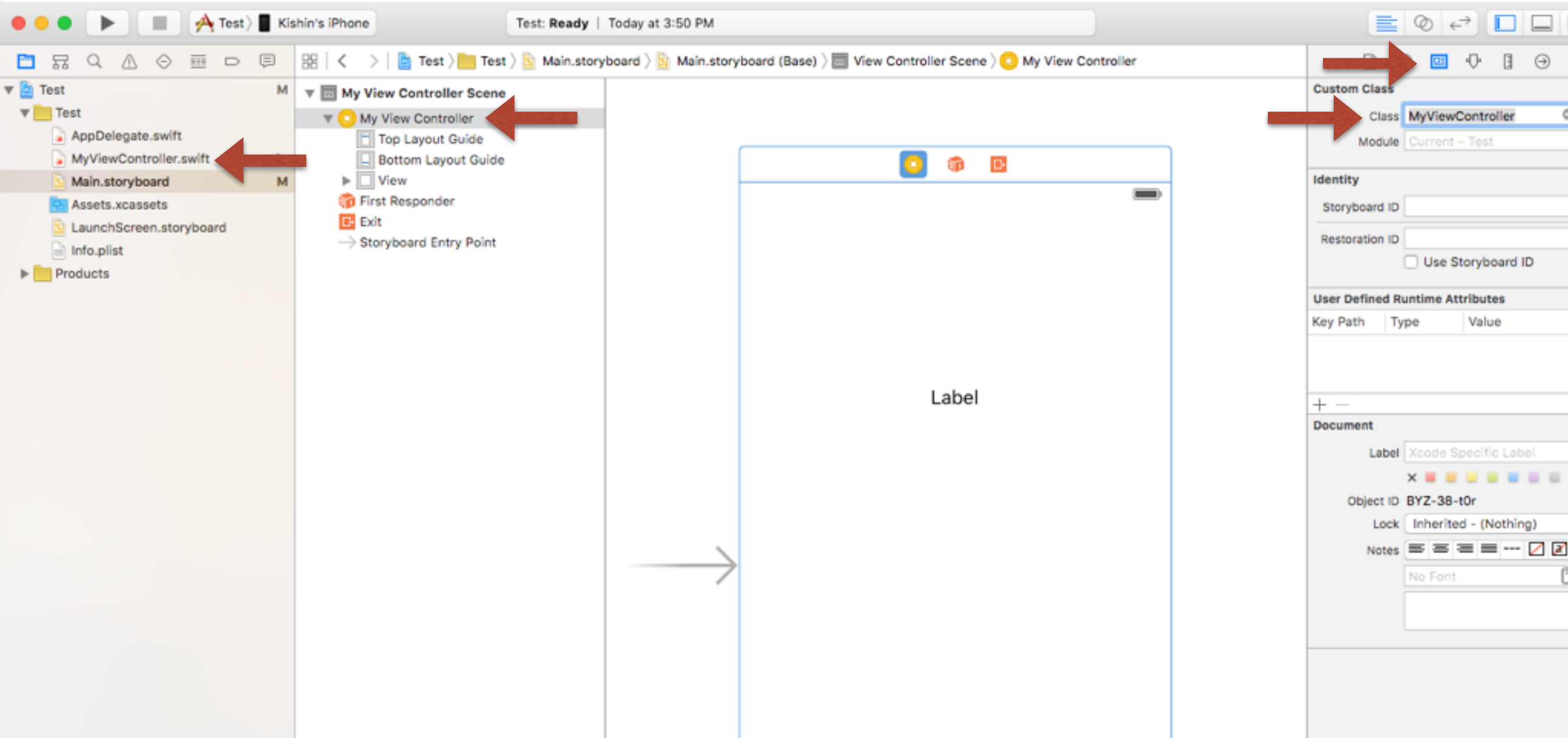
```
class MyViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        print("hello world. this is the console")
    }

}
```

REMEMBER UIKIT?

- Now the MyViewController class has all of the variables and all of the functions that UIViewController has
- MyViewController (child class) is a subclass of UIViewController (parent)
- UIViewController (parent) is the superclass of MyViewController (child)
- Since it is a UIViewController, Apple and Storyboard know how to draw it and know it has a view, etc.

```
class MyViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        print("hello world. this is the console")
    }

}
```

This is no coincidence

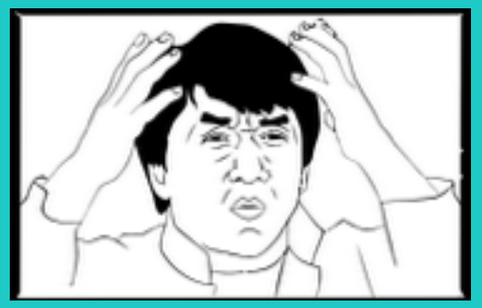


REMEMBER UIKIT?

- Now the MyViewController class has all of the variables and all of the functions that UIViewController has
- MyViewController (child class) is a subclass of UIViewController (parent)
- UIViewController (parent) is the superclass of MyViewController (child)
- Since it is a UIViewController, Apple and Storyboard know how to draw it and know it has a view, etc.

```
class MyViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        print("hello world. this is the console")
    }

}
```

INTRO TO SWIFT



IBOUTLET AND IBACTION

IBOUTLET

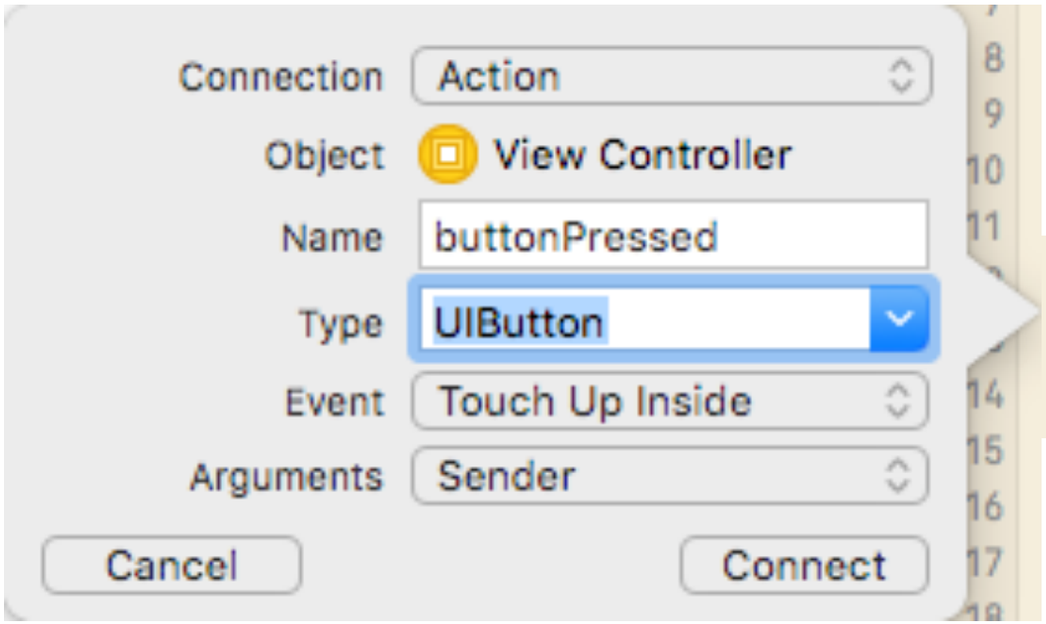
- IBOutlet or Interface Builder Outlet
- Create by control + dragging from our storyboard to our view controller file in the assistant editor
- What is this doing? This is creating a variable in our code that we can reference later
- So we are adding this variable to our subclass of UIViewControllers

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

IBACTION

- IBAction or Interface Builder Action
- Create by control + dragging from our storyboard to our view controller file in the assistant editor
- This is creating a function in our code that Interface Builder automatically invokes for us
- We are adding a function to our UIViewController subclass

IBACTION




```
@IBAction func buttonPressed(sender: UIButton) {

}
```

OBJECTIVES

- Learn the basics of Swift
- Understand what classes and objects are
- Bridge the gap between UIViews and UIViewControllers and code
- Blow your mind