GENERAL ASSEMBLY

MOBILE DEVELOPMENT

SWIFT & GITHUB

Kishin Manglani

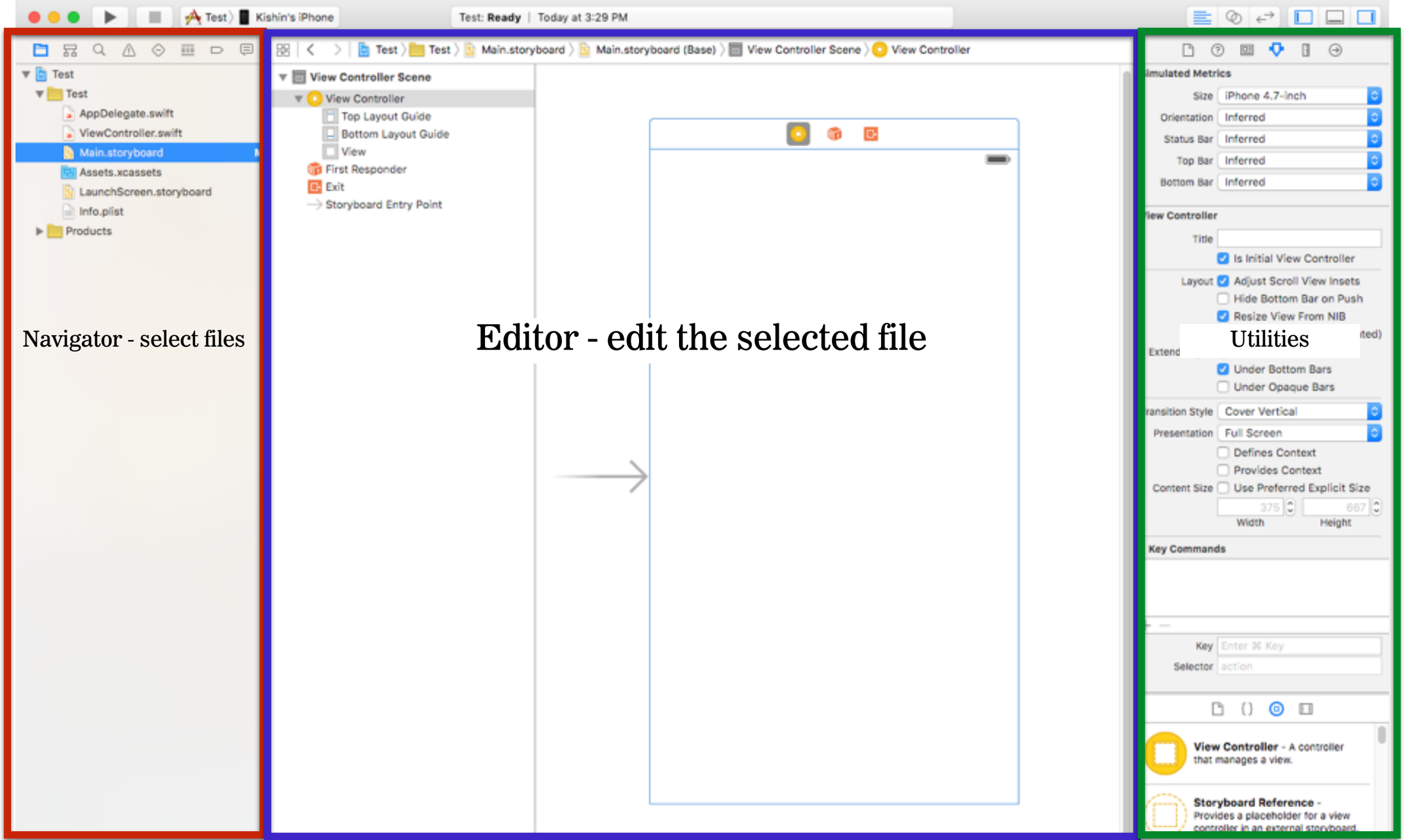
AGENDA

- Recap
- Objects and Classes
- Connecting Storyboard to Code
- Arrays
- For Loops
- Github

RECAP

RECAP 1

- Familiarized ourselves with Xcode and the different buttons
- UIView vs UIViewController
- Multiple View Controllers: what are the three topics we covered with multiple view controllers?

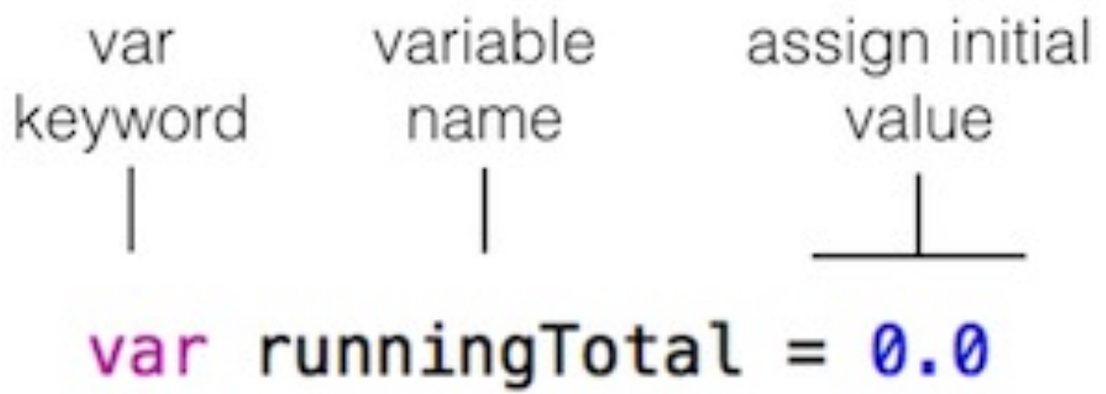

Test
Kishin's iPhone
Test: Ready | Today at 3:29 PM
Test
Test
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
View Controller Scene
View Controller
Top Layout 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.

RECAP

- What is a programming language?
- What keywords do we use to declare: variables, constants, functions, classes?
- What are the variable types we learned about?
- Mathematical, Comparison, Logical Operators
- If/Else Statements
- Functions

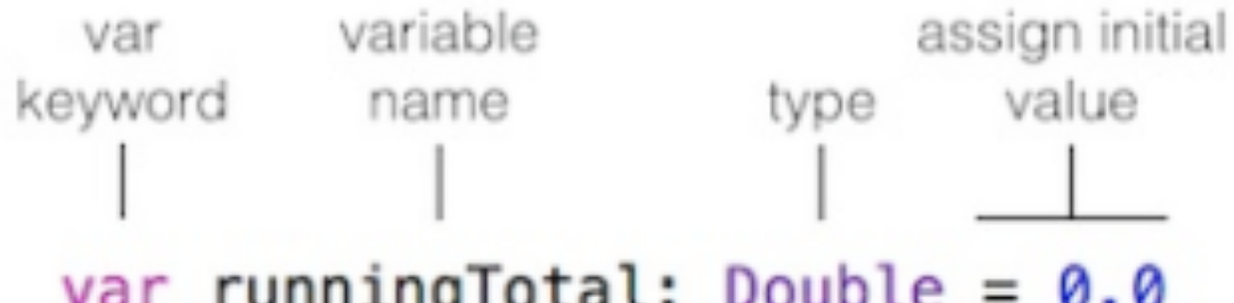
VARIABLES

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



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



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"

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")
}
```

FUNCTIONS

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

var totalAmount = addTax(10)
```

OBJECTIVES

- Develop a stronger understanding of classes
- Connect Storyboard to Code
- Learn how to use Arrays
- Learn how to use For Loops
- Learn how to use Github to submit homework

INTRO TO SWIFT

OBJECTS AND CLASSES

CLASSES

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

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

INITIALIZATION

- Initialization is the process of preparing an instance of a class for use
- Setting an initial value for each stored property on that instance and performing any other setup or initialization that is required before the new instance is ready for use
- Essentially, this is the process of creating an object out of a class

CLASSES

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

INITIALIZERS

- Initializers are called to create a new instance of a particular type. In its simplest form, an initializer is like an instance method with no parameters, written using the init keyword

```
init() {
    // perform some initialization here
}
```

INITIALIZERS

- After calling an initializer, ALL VARIABLES MUST HAVE A VALUE
- In other words, you must set all the variables to something
- An alternative is to set a default value, that way the initializer does not need to set the value, since it has a default value

CLASSES

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

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

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

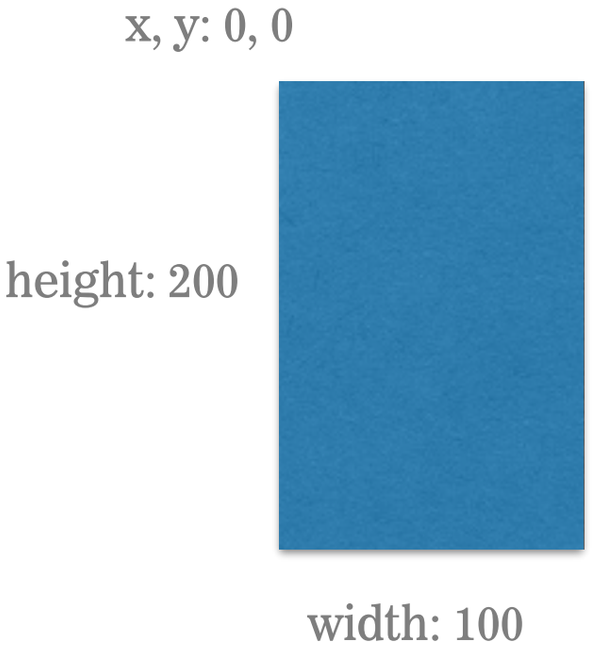
CLASSES

- Classes have functions/methods and variables/properties
- We know how to create variables and we know how to create functions, and now we've just put them inside of a class

CONNECTING STORYBOARD TO CODE

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?

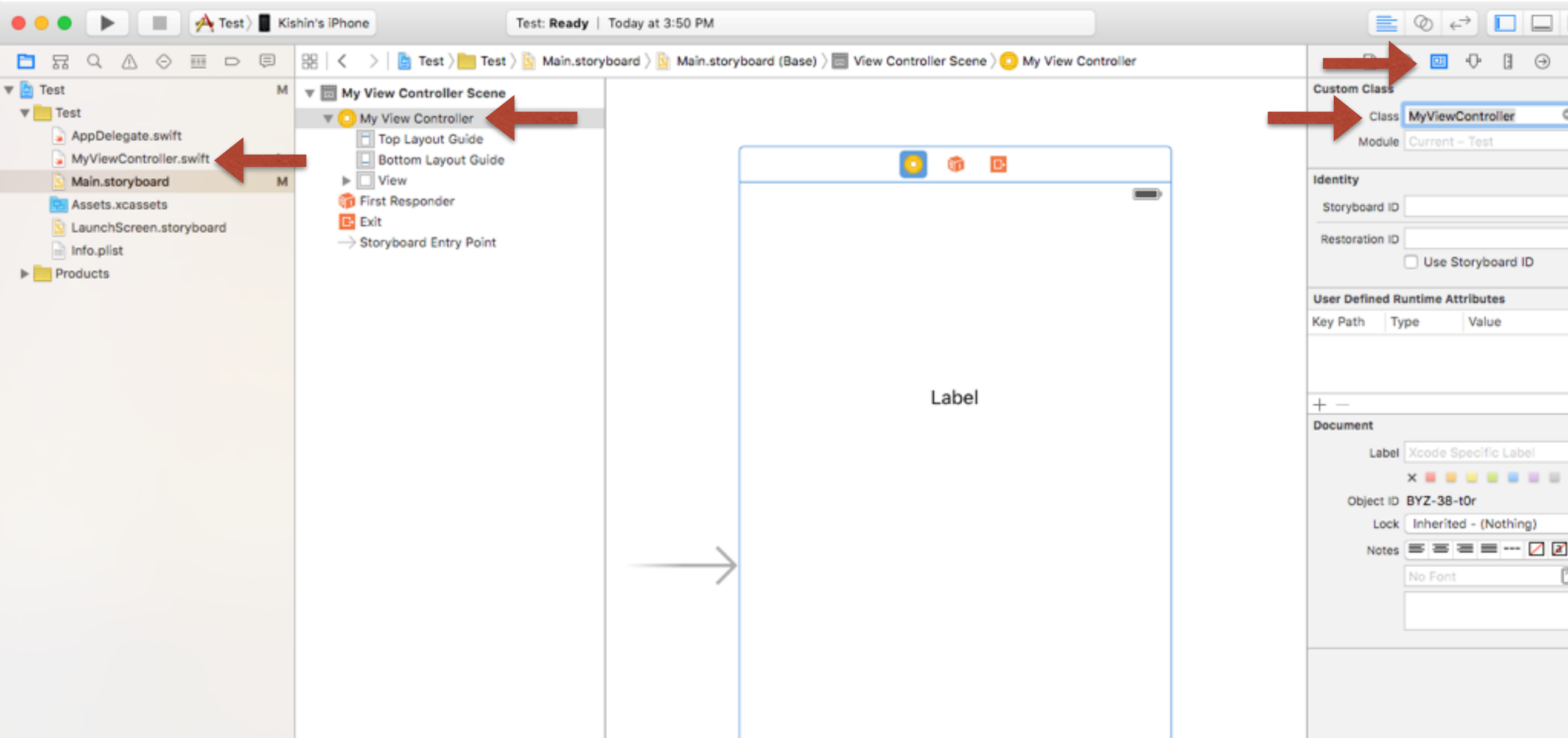
- Now the MyViewController class has all of the variables and all of the functions that UIViewController has
- MyViewController (child class) is a subclass of UIViewController (parent)
- UIViewController (parent) is the superclass of MyViewController (child)
- Since it is a UIViewController, Apple and 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")
    }

}
```

IBOUTLET AND IBACTION

IBOUTLET

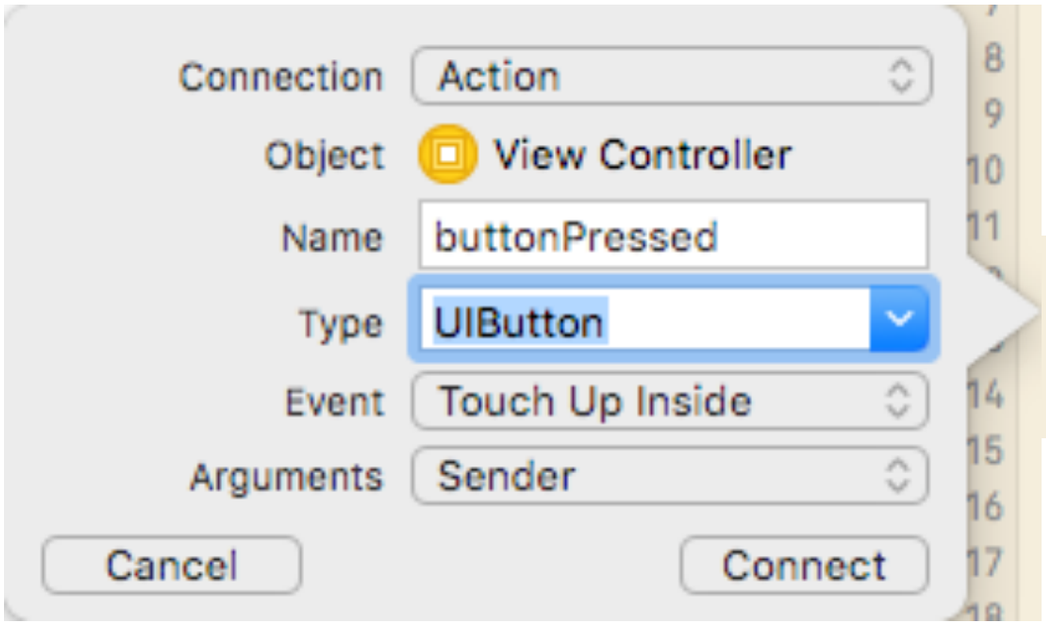
- IBOutlet or Interface Builder Outlet
- Create by control + dragging from our storyboard to our view controller file in the assistant editor
- What is this doing? This is creating a variable in our code that we can reference later
- So we are adding this variable to our subclass of UIViewControllers

```
@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) {

}
```

LET'S TRY IT

DEMO

- Add a UIButton to a View Controller
- Add an IBAction to the View Controller
- In the function/action, change the view background color:

```
self.view.backgroundColor = UIColor(red: 244, green: 244, blue: 244, alpha: 0)
```

ARRAYS

ARRAY

- Let's think about properties in our classes again
- In our city example, how would we store multiple buildings in the city class?

```
var shoppingList = ["Eggs", "Milk"]
```

```
var shoppingList: [String] = ["Eggs", "Milk"]
```

ARRAY

- An array is a collection data type
- An array stores values of the same type in an ordered list
- The same value can appear in an array multiple times at different positions

```
var shoppingList = ["Eggs", "Milk", "Eggs"]
```

ARRAY

- How do we access elements in an array?
- Arrays are zero-based, in other words the first element is really the zeroth element
- Subscript and index

```
var shoppingList = ["Eggs", "Milk"]
shoppingList[0]
shoppingList[1]
```

```
["Eggs", "Milk"]
"Eggs"
"Milk"
```

ARRAY

- Arrays have some useful properties
  - count: returns the number (Int) of objects in the array
  - isEmpty: returns a Bool checking if the array is empty (0 objects)
- Append method: adds an object to the array
  - shoppingList.append(“carrots”)

```
var shoppingList = ["Eggs", "Milk"]
shoppingList.append("Carrots")
shoppingList[2]
shoppingList.isEmpty
shoppingList.count
```

```
["Eggs", "Milk"]
["Eggs", "Milk", "Carrots"]
"Carrots"
false
3
```

FOR LOOPS

FOR LOOPS

- So arrays allow us to store an undetermined amount of data
- How can we process or perform some logic on an undetermined amount of data?
- For loops
- Begin and end with curly braces
- Run code between curly braces 0 or more times

FOR LOOPS

```
for var i = 0; i < 3; i++
{
    print(i)
}
```

FOR LOOPS

```
for i in 0...2 {
    print(i)
}
```

FOR LOOPS

```
var shoppingList = ["Eggs", "Milk", "Cheese"]
for item in shoppingList {
    print(item)
}
```

```
["Eggs", "Milk", "Cheese"]

(3 times)
```

GITHUB DEMO