# Basic Exercises Part 3.5 UISwitch, UIStepper, UIProgressView

## UISwitch

* The UISwitch class declares a property and a method to control its on/off state. Actually, is a switch the same as a slider? … but only with two values.
* As with UISlider, when the user manipulates the switch control (“flips” it) a valueChanged event is generated, which results in the control (if property configured) sending an action message.
* You can customize the appearance of the switch by changing the color used to tint the switch when it is on or off.
  + You can edit the minimum, maximum and current values in the Attributes Inspector.

### **1.1 Apple documentation**

For information about basic behaviors, visit:

https://developer.apple.com/library/archive/documentation/WindowsViews/Conceptual/ViewPG\_iPhoneOS/Introduction/Introduction.html#//apple\_ref/doc/uid/TP40009503

### **1.2 Create a new project**

Create a basic Single View.

### **1.3 Add the Switch to the view**

Open the Library (also: Shift + Cmd + L) and search for: UISwitch. Click and drag it onto your view. Next, do the same thing with a UILabel. Search for it and then drag it onto your view.

### **1.4 Attach the elements as IBOutlets**

Hold down CTRL and click either the UILabel or UISwitch element on the storyboard and drag over to the right hand side in-between the class ViewController and override func.

A blue line should follow you mouse cursor and when you release, a dialog will popup and ask you what name your IBOutlet property. Name it “switch” o “label” depending on which element you’re currently doing. Do it for both.

What we’re doing here is connecting those elements in the storyboard to properties of the ViewController class so that we can access these elements programmatically in the ViewController class.

*A common mistake: If you made a mistake in connecting the elements or naming your properties and you want to re-do it, you can delete the property in the .swift file but you also have to break the connection by going to your storyboard, right-clicking the element and clicking the “x” beside the outlet reference. If you don’t do this and you only delete the property from the .swift file, then the element will be connected to a property that no longer exists and your app will crash!*

### **1.5 Connecting the Switch value changed event**

In order to respond to event of the switch value changing, we have to hook up the event in the .swift file. Once again, we want to be in **Assistant Editor** view.

You’re going to hold down CTRL and click and drag the slider element so that a blue line follows your mouse. Drag it to an empty space in between the **override func** and the **closing curly brace** of the class method. In the dialog that pops up, you can name the IBAction method “**switchSateDidChanged**”, and add the following code:

**override** **func** viewDidLoad() {

**super**.viewDidLoad()

**let** switchOnOff = UISwitch(frame:CGRect(x: 150, y: 150, width: 0, height: 0))

switchOnOff.addTarget(**self**, action: **#selector**(ViewController.switchStateDidChange(**\_**:)), for: .valueChanged)

switchOnOff.setOn(**true**, animated: **false**)

**self**.view.addSubview(switchOnOff)

}

**@objc** **func** switchStateDidChange(**\_** sender:UISwitch){

**if** (sender.isOn == **true**){

print("UISwitch state is now ON")

}

**else**{

print("UISwitch state is now Off")

}

}

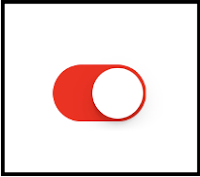
In the code above, we’re simply getting the state of the switch and add printing the state.

Assigning the state as the text of the label. Run the app now and drag the switch around. You’ll see that the label text changes to reflect the state of the switch.

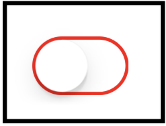
### **1.6 Customizing the UISwitch**

Problem. You are using the default appearance of the UISwitch UI component, and now you want to customize this look and feel.

* onTintColor. Modify the tint color of switch when it is turn on.



* tintColor. Set tint color of switch when is turn off.



* thumbTintColor. Set tint color of the thumb.

A picture containing drawing

Description automatically generated

# Swift

* onImage. Set image of the switch when it is on.

switchOnOff.onImage = UIImage(named: "on-switch")

* offImage. Set image of the switch when it is off.

switchOnOff.offImage = UIImage(named: "off-switch")

Note: You need to change the image name.

# Objective-C

//set off-image

mySwitch.offImage = [UIImage imageNamed:@"off\_image"];

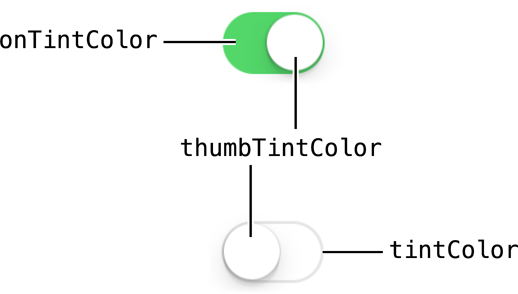
[mySwitch setOffImage:[UIImage imageNamed:@"off\_image"]];

//set on-image

mySwitch.onImage = [UIImage imageNamed:@"on\_image"];

[mySwitch setOnImage:[UIImage imageNamed:@"on\_image"]];

Recap:



### **1.7 Turn off the lights!**

Maybe Nelly Furtado has a lot of lights on her house or her house basement needs some lights. Let’s help her a little by creating a funny app to turn off the light (and follow me down down down!!).

Hints: #1 Draw a light in the top of the screen.

#2. Add a UISwitch to your app to control the light. On -> yellow color. Off -> clear color.

#3. Add output labels to display the light state.

#4. Get some inspiration. <https://www.youtube.com/watch?v=kOL7aeIDruA>

### **1.8 Remember to turn the lights off.**

Save the state of the switch. Next time the app comes from background, present the previous state for the switch/label/light.

### **1.9 Forgot something?**

Why we always leave Objective C at the end?

## UIStepper

* The UISetepper class is a control to increment or decrement a value.
* If you set stepper behavior to “autorepeat” (which is the default), pressing and holding one of its buttons increments or decrements the stepper’s value repeatedly. The rate of change depends on how long the use continues pressing the control.
* The maximum value must be greater than or equal to the minimum value. If you set up a maximum or minimum value that would break this invariant, both values are set to the new value. For example, if the minimum value is 200 and you set a maximum value of 100, then both the minimum and maximum become 200.

### **2.1 Create a new project**

Create a basic Single View.

### **2.2 Add the Stepper to the view**

Open the Library (also: Shift + Cmd + L) and search for: UIStepper. Click and drag it onto your view. Next, do the same thing with a UILabel. Search for it and then drag it onto your view (ie create the outlets). Again, Ctrl and drag from the Segmented Control to the ViewController.swift class and create an action. This Action or method is called when the segment index is changed by the user.

Select the “Resolve Auto Layout Issues” button on the bottom-right of the IB (interface builder) and choose “Reset to Suggested Constraints”.

Implement this code:

# Swift

**var** label : UILabel!

**var** stepper : UIStepper!

**override** **func** loadView() {

// UI

**let** view = UIView()

view.backgroundColor = .white

label = UILabel()

stepper = UIStepper()

stepper.tintColor = .red

stepper.addTarget(**self**, action: **#selector**(updateView), for: .valueChanged)

stepper.value = 15

stepper.minimumValue = -100

stepper.maximumValue = 100

stepper.stepValue = 5

view.addSubview(label)

view.addSubview(stepper)

// Layout

label.translatesAutoresizingMaskIntoConstraints = **false**

stepper.translatesAutoresizingMaskIntoConstraints = **false**

NSLayoutConstraint.activate([

label.topAnchor.constraint(equalTo: view.topAnchor, constant: 20),

label.leadingAnchor.constraint(equalTo: view.leadingAnchor, constant: 20),

stepper.leadingAnchor.constraint(equalTo: label.trailingAnchor, constant: 20),

stepper.centerYAnchor.constraint(equalTo: label.centerYAnchor),

])

**self**.view = view

updateView()

}

**@objc** **func** updateView() {

label.text = "Counter: \(Int(stepper.value))"

}

### **2.3 Foo notes**

The UIStepper is a great way to increment or decrement a small numeric value. For large numbers, or numbers where large adjustments are needed, UIStepper is not ideal.

### **2.4 One challenge**

Try on Objective C. Same project than 2.2.

## UIProgressView

* The UIProgressView class is a view that depicts the progress of a task over time.
* Provides properties for managing the style of the progress bar and for getting and setting values that are pinned to the progress of a task.

### **3.1 Create a new project**

Create a basic Single View.

### **3.2 Add the Progress view to the view**

Open the Library (also: Shift + Cmd + L) and search for: UIProgressView. Click and drag it onto your view. Next, do the same thing with a UILabel and UIButton. Search for it and then drag it onto your view (ie create the outlets).

Use a UIProgressView and the progress property with a float to display a progress bar. This animated bar can tell the user that progress is being made.

Label. We add a Label to display a textual form of our progress (this is not always needed).

Button. The button is used (here) to increment the progress by 10% each tap.

### **3.3 Add some action**

We use primary action is triggered and create an “actionTriggered” function. This is where our logic is placed.

Inside the code. We use an “if” statement to only allow the function to handle 10 button taps (not including the initial state).

Float. Progress is a float. We use the Float class to ensure we get a good Float progress number. It must go from 0 to 1.

Increment. On each button tap, we increment the value of the “current” Int by 1. So the program’s state changes on each tap.

Issues? To correctly use UIProgressView we usually need to assign its progress in a viewDidLoad method. This initializes the state of the control.

### **3.4 Coding**

We use primary action is triggered and create an “actionTriggered” function. This is where our logic is placed.

@IBOutlet weak var simpleProgress: UIProgressView!

@IBOutlet weak var simpleLabel: UILabel!

var current: Int = 0

override func viewDidLoad() {

super.viewDidLoad()

}

@IBAction func actionTriggered(sender: AnyObject) {

// Get current values.

let i = current

let max = 10

// If we still have progress to make.

if i <= max {

// Compute ratio of 0 to 1 for progress.

let ratio = Float(i) / Float(max)

// Set progress.

simpleProgress.**progress** = Float(ratio)

// Write message.

simpleLabel.text = "Processing \(i) of \(max)..."

current++

}

}

}

Can we use a slider instead of a button?.

### **3.4 How to control a progress view’s animation**

The UIProgressView class has a built-in setProgress() method that adjusts its progress with animation, but by default you don’t have control over the timing of that animation. Fortunately, if you want to time the animation to match other aspects of your user interface you can wrap it inside an animation block of you own.

For example, given a UIProgressView stored in a progressView property, this will animate the progress view to completion over four seconds:

UIView.animate(withDuration: 4.0) {

**self**.progressView.setProgress(1.0, animated: true)

}