# Basic Exercises Part 3.4 UISlider & UISegmentedControl

## UISlider

* The UISlider class is used to select a value from a possible range of values.
* By default a slider has a value between 0 and 1.
  + You can edit the minimum, maximum and current values in the Attributes Inspector.

### **1.1 Create a new project**

Create a basic Single View.

### **1.2 Add the Slider to the view**

Open the Library (also: Shift + Cmd + L) and search for: UISlider. 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.

### **1.3 Attach the elements as IBOutlets**

Hold down CTRL and click either the UILabel or UISlider 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 “slider” 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!*

### **1.4 Connecting the Slider value changed event**

In order to respond to event of the slider 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 “**sliderValueChanged**”, and add the following code:

label.text = “\(slider.value)”

In the code above, we’re simply getting the value of the slider and adding it to a string. Then we’re assigning that string as the text of the label. Run the app now and drag the slider thumb around. You’ll see that the label text changes to reflect the value of your slider.

### **1.5 Apple documentation for Text Fields**

Visit the page:

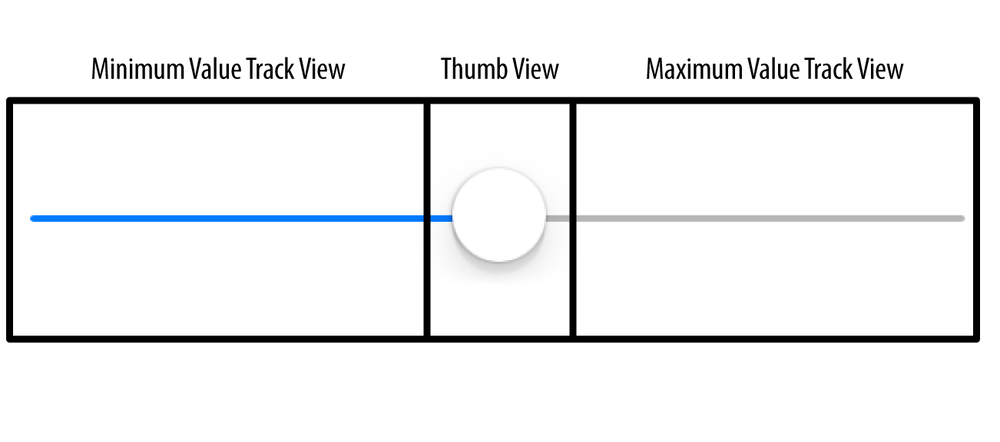
<https://developer.apple.com/documentation/uikit/uislider>

### **1.6 Customizing the UISlider**

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

Solution. Either modify the tint colors of the different parts of the slider or provide your own images for the parts (not recommended).

Apple giving us methods to modify the tint colors of various parts of the UI component. This is a slider broken it down into its different UI components.



For each of these components in UISlider, a method and property exist that allow you to change the appearance of the slider. The easiest of these properties to use are the ones that modify the tint color of these components:

* + minimumTrackTintColor. This property sets the tint color of the minimum value track view.
  + thumbTintColor. This property, as its name shows, sets the tint color of the thumb view.
  + maximumTrackTintColor. This property sets the tint color of the maximum value track view.

The following sample code instantiates a UISlider and places it at the center of the view of the view controller. It also sets the tint color of the minimum value tracking view of the slider to red, the tint color of the thumb view of the slider to black, and the tint color of the maximum value tracking view of the slider to green:

**-** (**void**)viewDidLoad{

[super viewDidLoad];

*/\* Create the slider \*/*

self.slider = [[UISlider alloc] initWithFrame:CGRectMake(0.0f, 0.0f, 118.0f, 23.0f)];

self.slider.value = 0.5;

self.slider.minimumValue = 0.0f;

self.slider.maximumValue = 1.0f;

self.slider.center = self.view.center;

[self.view addSubview:self.slider];

*/\* Set the tint color of the minimum value \*/*

self.slider.minimumTrackTintColor = [UIColor redColor];

*/\* Set the tint color of the thumb \*/*

self.slider.maximumTrackTintColor = [UIColor greenColor];

*/\* Set the tint color of the maximum value \*/*

self.slider.thumbTintColor = [UIColor blackColor];

}

### **1.7 What do you think of variables? How’s the tip calculator?**

Create a slider that represents the tip you want to leave in the restaurant (the more elements you add, the more professional looks like).

Hints: #1 Add a custom tip percentage field and use Double values.

#2. Add a UISlider to your app to control the tip percentage.

#3. Add output labels to display the tip.

#4. The problem with the UISlider is that it returns floats that kind of mess up with the custom tip percentage. Solve the problem to present strings in the output label.

Questions

**1 What is the difference between the following two lines of code?**  
var x = 1 / 3  
var y = 1.0 / 3.0

x is implicit **Int**; x = 0  
y is implicit **Double**; y = 0.333333333333333

## UISegmentedControl

* The UISegmentedControl class is a set of two or more segments, each of which functions as a mutually exclusive button. Within the control, all segments are equal in width.

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

Create a basic Single View.

### **2.2 Add the Segments to the view**

Open the Library (also: Shift + Cmd + L) and search for: UISegmentedControl. 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:

@IBAction func indexChanged(\_ sender: Any) {

switch segmentedControl.selectedSegmentIndex

{  
 case 0:  
 textLabel.text = "First Segment Selected"  
 case 1:  
 textLabel.text = "Second Segment Selected"  
 default:  
 break  
 }  
}

### **2.3 Project: Is a square, a ball or a rectangle?**

Create an application named “SquareBallRectangle”, which displays a square, a ball or a rectangle, depending on the selection made with UISegmentedControl object.

Present all the shapes in the center of the screen.

### **2.4 Triangle**

Modify the program so that it draws a triangle in place of the rectangle (or other) use the following code snippet:

# Swift

**@IBOutlet** **weak** **var** triangleView: UIView!

**func** setUpTriangle(){

**let** heightWidth = triangleView.frame.size.width

**let** path = CGMutablePath()

path.move(to: CGPoint(x: 0, y: heightWidth))

path.addLine(to: CGPoint(x:heightWidth/2, y: heightWidth/2))

path.addLine(to: CGPoint(x:heightWidth, y:heightWidth))

path.addLine(to: CGPoint(x:0, y:heightWidth))

**let** shape = CAShapeLayer()

shape.path = path

shape.fillColor = UIColor.blue.cgColor

triangleView.layer.insertSublayer(shape, at: 0)

}

### **2.4 Pie slice**

Modify the program so that it provides the possibility to select a fourth shape, which you can name as “Pie Slice”. You must use the Interface Builder so that the UISegmentedControl will have four selectable segments. In addition you must modify the drawing method so that it will draw the “Pie Slice”.

After these modifications, the app should look like the following when “Pie Slice” is selected.

A screenshot of a cell phone

Description automatically generated

### **2.5 Again**

As a challenge, repeat the project but this time …. with Objective C.