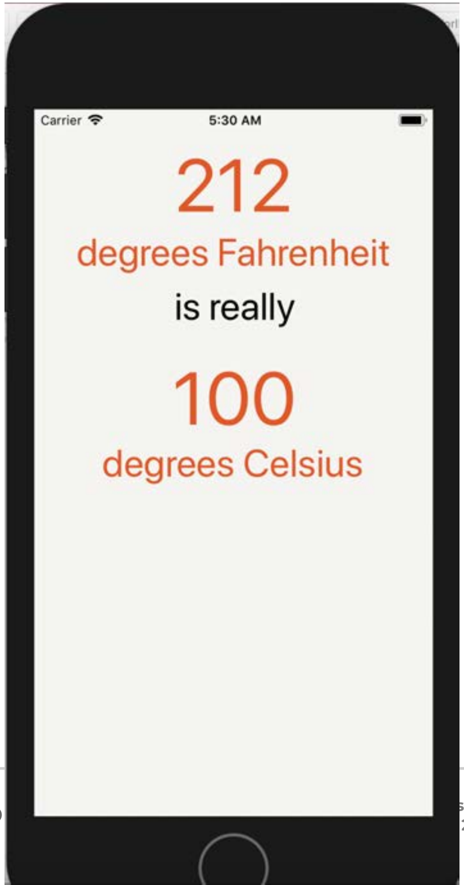
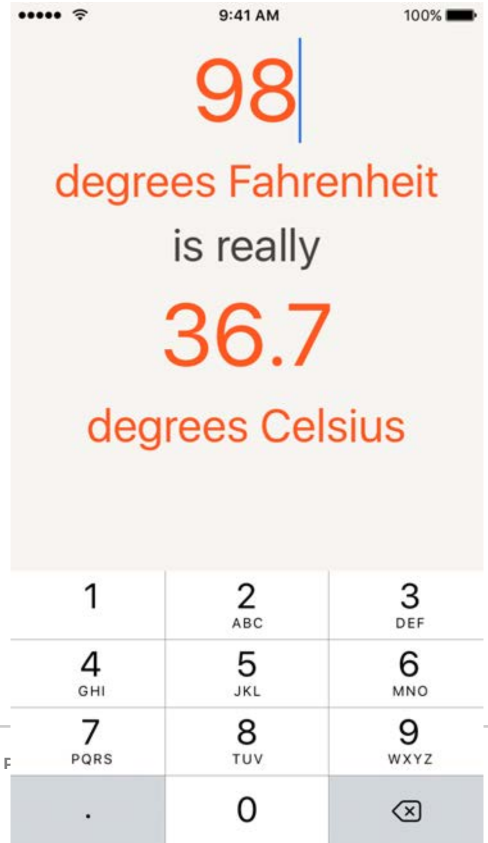
Text Input

Keyboard

Text Input

UITextField allows the user to type in a text in temperature text field





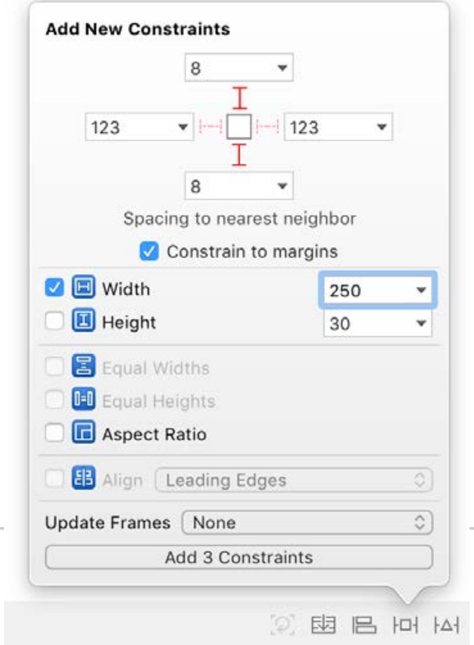
Text Input

Open Main.storyboard.
Add a TextField to the interface to replace the top label "212:"

Open the object library and drag a Text Field to the top of the canvas
- set up the constraints for this text field.

In the Align menu align the view Horizontally in Container with a constant of 0 and then Add 1 Constraint.

- Open the Add New Constraints menu.
- Give the text field a top edge constraint of 8 points,
- a bottom edge constraint of 8 points,
- and a width of 250. Add these three constraints.


Add New Constraints

8

123

123

8

Spacing to nearest neighbor

✓ Constrain to margins

✓ Width 250

Height 30

Equal Widths

Equal Heights

Aspect Ratio

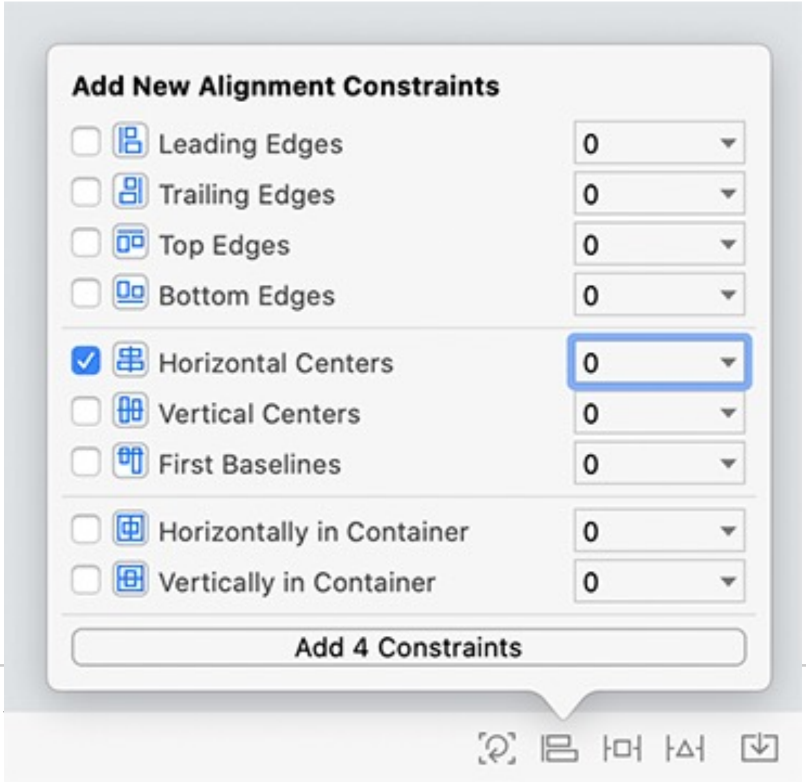
Align Leading Edges

Update Frames None

Add 3 Constraints

Text Input

Select the text field and the 4 labels below it.
Open the Align menu, select Horizontal Centers with a constant of 0,
Update Frames for All Frames in Container, and finally Add 4 Constraint



Add New Alignment Constraints

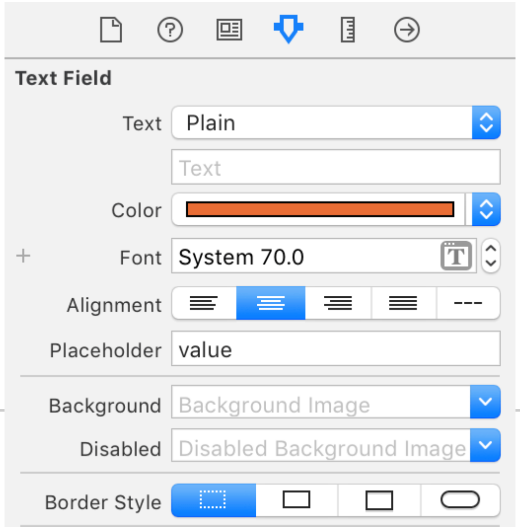
☐	Leading Edges	0
☐	Trailing Edges	0
☐	Top Edges	0
☐	Bottom Edges	0
✓	Horizontal Centers	0
☐	Vertical Centers	0
☐	First Baselines	0
☐	Horizontally in Container	0
☐	Vertically in Container	0

Add 4 Constraints

Text Input

customize some of the text field properties. Open the attributes inspector:

• Set the text color (from the Color menu) to burnt orange (hex color E15829).
• Set the font size to System 70.
• Set the Alignment to centered.
• Set the placeholder text to be value. This is the text that will be displayed when the user has not entered any text.
• Set the Border Style to be none, which is the first element of the segmented control with the dotted
lines.
- Uncheck Adjust to Fit under Min Font Size.



Text Field

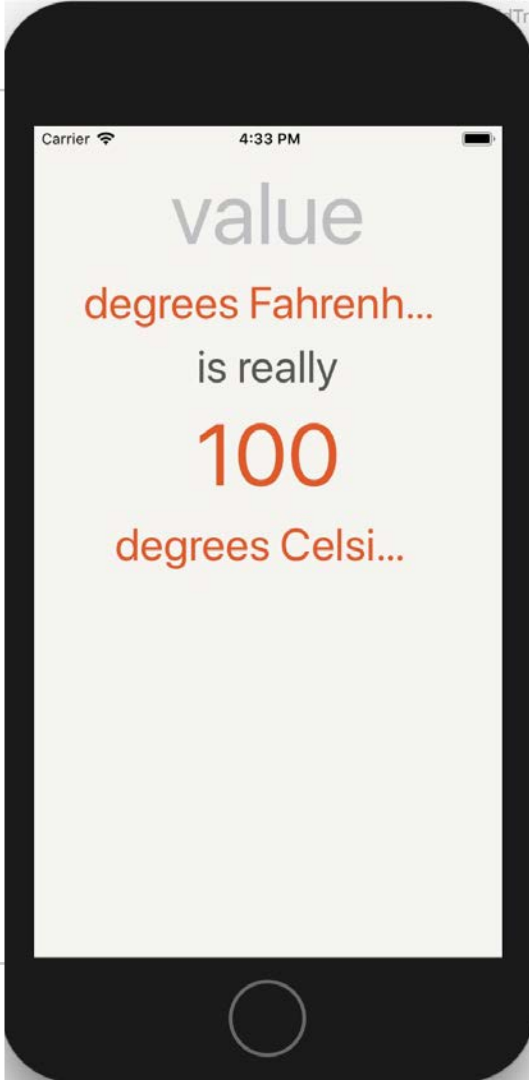
Text	Plain
	Text
Color	🟧
Font	System 70.0
Alignment	centered
Placeholder	value
Background	Background Image
Disabled	Disabled Background Image
Border Style	none

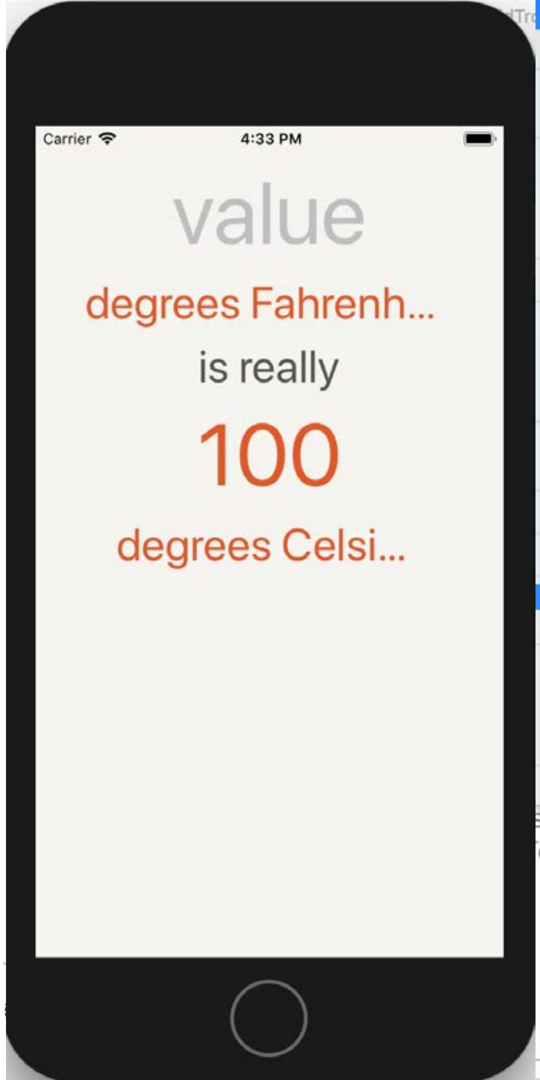
Text Input

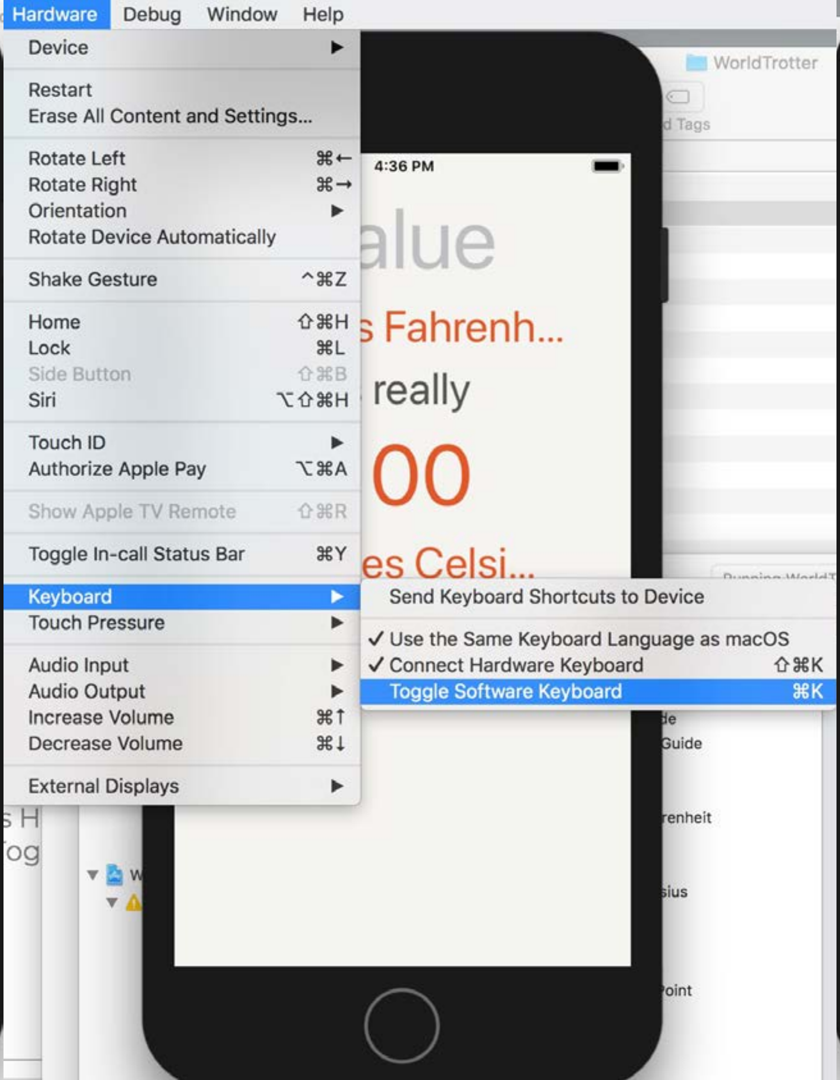
The text field and labels will be repositioned to match their constraints.

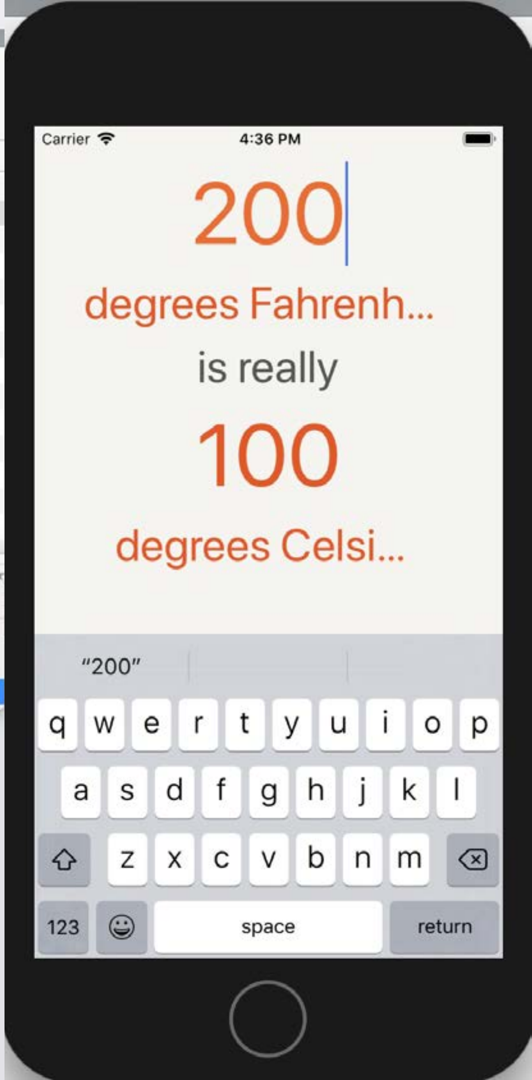
Build and run the app.
Tap on the text field and enter some text.

To show the keyboard,
click the simulator’s Hardware menu >
select Keyboard → Toggle Software Keyboard.




Carrier
4:33 PM
value
degrees Fahrenh...
is really
100
degrees Celsi...

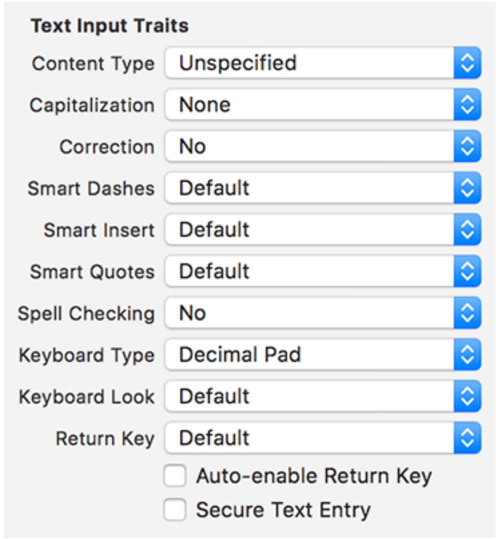

Hardware
Debug
Window
Help
Device
Restart
Erase All Content and Settings...
Rotate Left ⌘←
Rotate Right ⌘→
Orientation
Rotate Device Automatically
Shake Gesture ^⌘Z
Home ⇧⌘H
Lock ⌘L
Side Button ⇧⌘B
Siri ⌥⇧⌘H
Touch ID
Authorize Apple Pay ⌥⌘A
Show Apple TV Remote ⇧⌘R
Toggle In-call Status Bar ⌘Y
Keyboard
Touch Pressure
Audio Input
Audio Output
Increase Volume ⌘↑
Decrease Volume ⌘↓
External Displays
Send Keyboard Shortcuts to Device
✓ Use the Same Keyboard Language as macOS
✓ Connect Hardware Keyboard ⇧⌘K
Toggle Software Keyboard ⌘K
WorldTrotter
4:36 PM


Carrier
4:36 PM
200
degrees Fahrenh...
is really
100
degrees Celsi...
"200"
123
space
return

Keyboard attributes

When a text field is tapped, the keyboard automatically slides up onto the screen.
The keyboard's appearance is defined by a set of the UITextField's properties called the UITextInputTraits.

Keyboard Type - Decimal Pad.
Change both Correction and Spell Checking to No



Text Input Traits

Content Type	Unspecified
Capitalization	None
Correction	No
Smart Dashes	Default
Smart Insert	Default
Smart Quotes	Default
Spell Checking	No
Keyboard Type	Decimal Pad
Keyboard Look	Default
Return Key	Default

☐ Auto-enable Return Key
☐ Secure Text Entry

Build and run the application. Tapping on the text field will now reveal the decimal pad.

Responding to text field changes

The next step is to update the Celsius label when text is typed into the Fahrenheit text field.
You will need to use code to do the conversion between Fahrenheit and Celsius.

This code will go into the view controller subclass associated with this interface.

“Open ConversionViewController.swift and define this outlet and action. ”

Responding to text field changes

In ConversionViewController.swift :

```
class ConversionViewController: UIViewController {

    @IBOutlet var celsiusLabel: UILabel!

    override func viewDidLoad() {
super.viewDidLoad()

        print("ConversionViewController loaded its view.")
    }

    @IBAction func fahrenheitFieldEditingChanged(_ textField:
UITextField) {
        celsiusLabel.text = textField.text
    }
}
```

Responding to text field changes

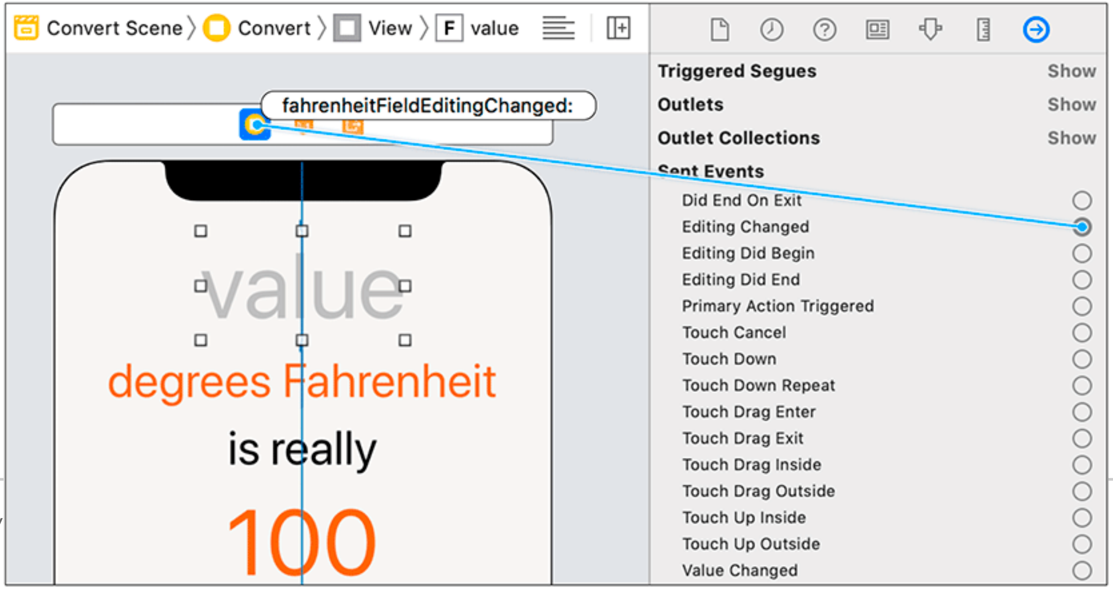
Open Main.storyboard “to make these connections.

Control-drag from the conversion view controller (Convert) to the Celsius label (the one that currently says 100) and connect it to the celsiusLabel.

select the text field on the canvas and open its connections inspector in the inspector area (the right-most tab, or Command-Option-7). The connections inspector allows you to make connections and see what connections have already been made.

Responding to text field changes

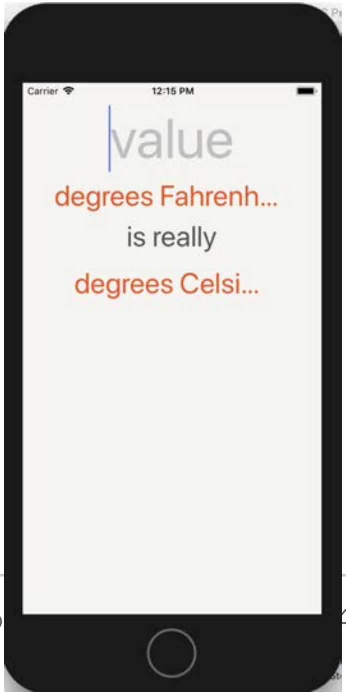
In the connections inspector, locate the Sent Events section and the Editing Changed event. Click and drag from the circle to the right of Editing Changed to the conversion view controller and click the fahrenheitFieldEditingChanged: action in the pop-up menu.

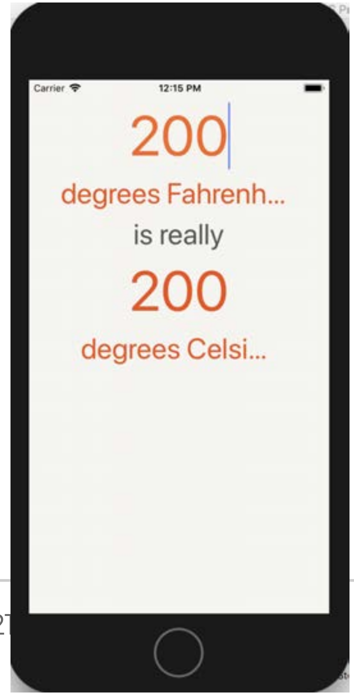


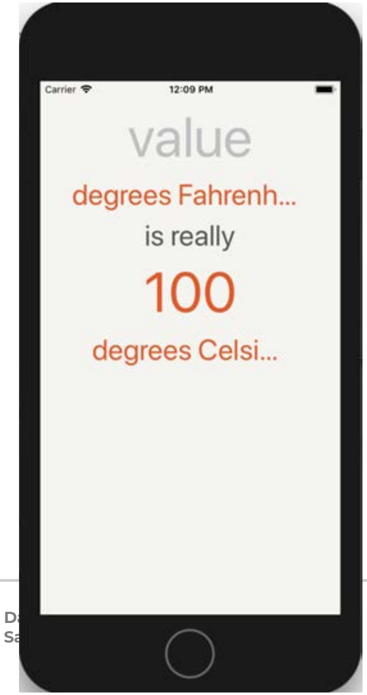
Responding to text field changes

Build and run the application.
Tap the text field and type some numbers. The Celsius label will mimic the text that is typed in.

If it does not work or you get an error- connect the action and event again.







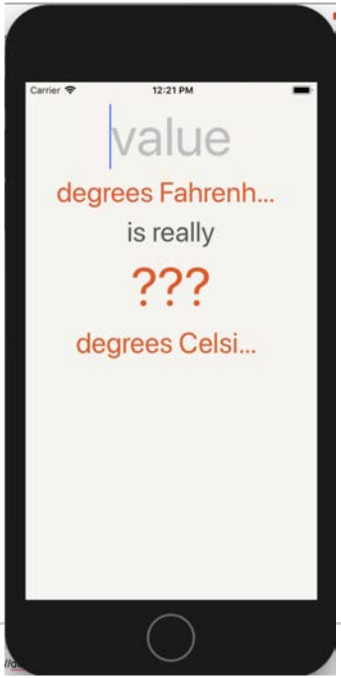
Responding to text field changes

delete the text in the text field and notice how the label seems to go away.
A label with no text has an intrinsic content width and height of 0, so the labels below it move up. This can be fixed.

In ConversionViewController.swift, update fahrenheitFieldEditingChanged(_:)

```
@IBAction func fahrenheitFieldEditingChanged(_ textField: UITextField) {
~~celsiusLabel.text = textField.text~~
if let text = textField.text, !text.isEmpty {
celsiusLabel.text = text
} else {
celsiusLabel.text = "???"
}
}
```

If the text field has text and that text is not empty,
it will be set on the celsiusLabel, otherwise the celsiusLabel will be given the string“???”.
Build and run the application to confirm.



ConversionViewController.swift code

```
import UIKitclass ConversionViewController: UIViewController {

@IBOutlet var celsiusLabel: UILabel!
@IBAction func fahrenheitFieldEditingChanged(_ textField: UITextField) {

if let text = textField.text, !text.isEmpty {

celsiusLabel.text = text
} else {
celsiusLabel.text = "???"
}
}}
```

Dismissing the keyboard

the user taps on the background view to trigger the dismissal

When the text field is tapped, the method becomeFirstResponder() is called on it.

To dismiss the keyboard, you call the method resignFirstResponder() on the text field. This method that is triggered when the background view is tapped.

Open ConversionViewController.swift and declare an outlet

```
@IBOutlet var celsiusLabel: UILabel!
@IBOutlet var textField: UITextField!
```

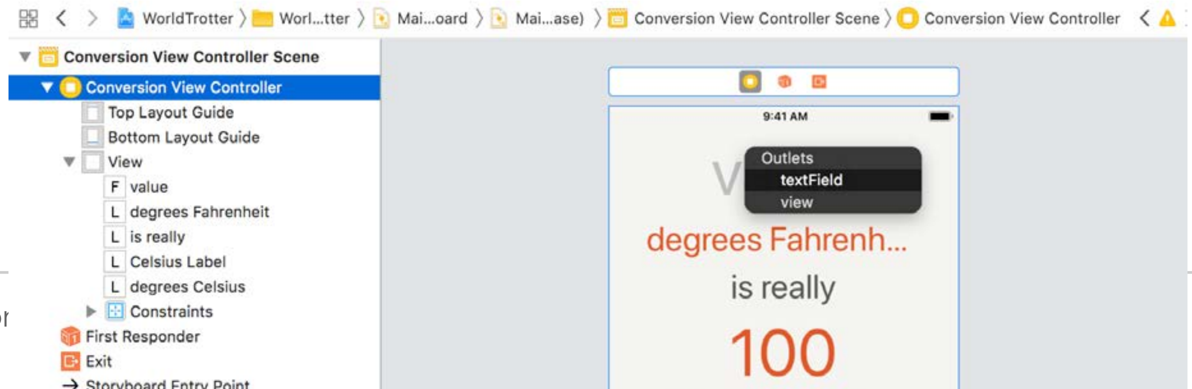
Dismissing the keyboard

implement an action method that will dismiss the keyboard when called.

```
@IBAction func dismissKeyboard(_ sender:
UITapGestureRecognizer) {
textField.resignFirstResponder()
}
```

1. The textField outlet needs to be connected in the storyboard file.

   open Main.storyboard and select the Conversion View Controller.
   Control-drag from the Conversion View Controller to the text field on the canvas and connect it to the textField outlet.



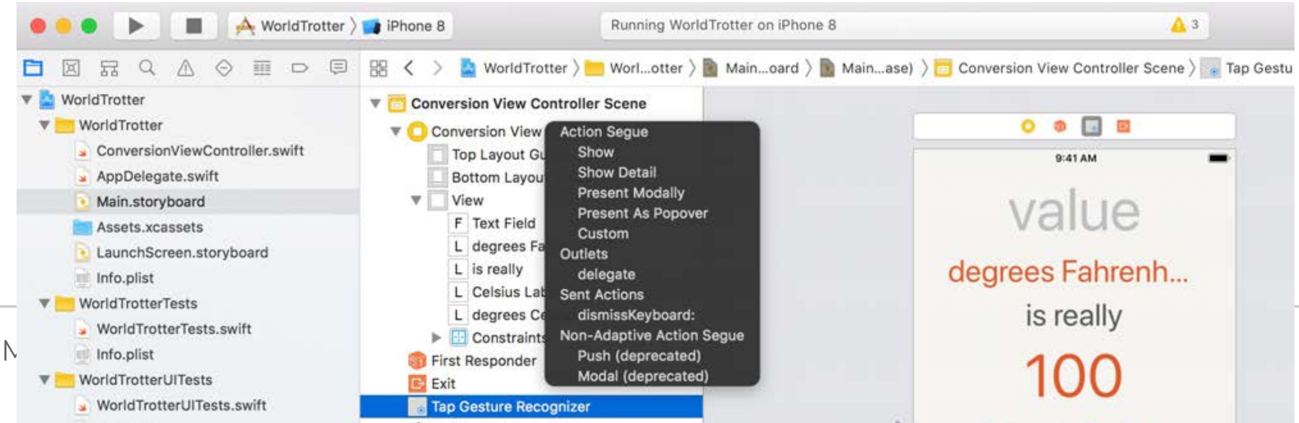
Dismissing the keyboard

2. We need a way of triggering the dismissKeyboard(_:) method

   a gesture recognizer subclass of UIGestureRecognizer to detect when the user taps the background view.
   (recognizer that detect taps, swipes, long presses, etc.)

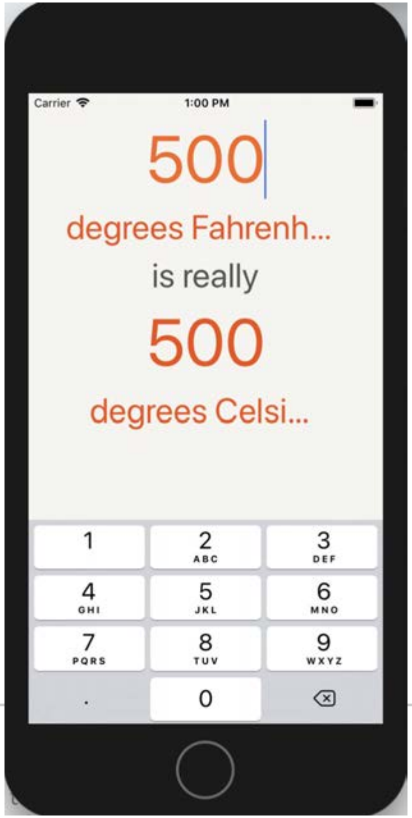
   In Main.storyboard, find Tap Gesture Recognizer in the object library. Drag this object onto the background view for the Conversion View Controller.

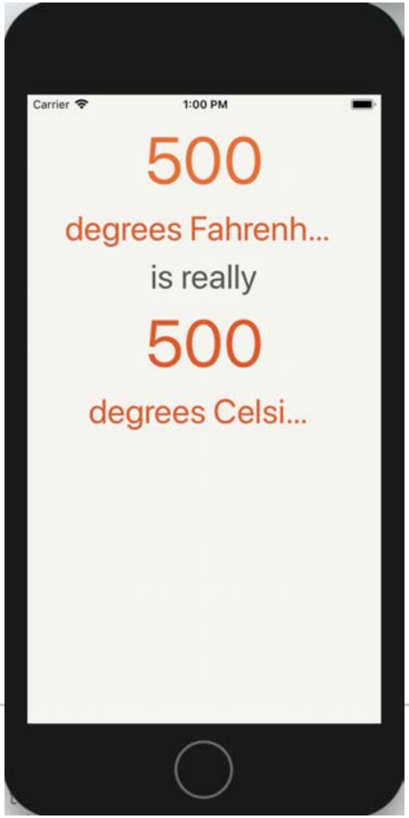
   Control-drag from the gesture recognizer in the scene dock to the Conversion View Controller and connect it to the dismissKeyboard: method



Dismissing the keyboard

Build and run to test the keyboard dismissal.

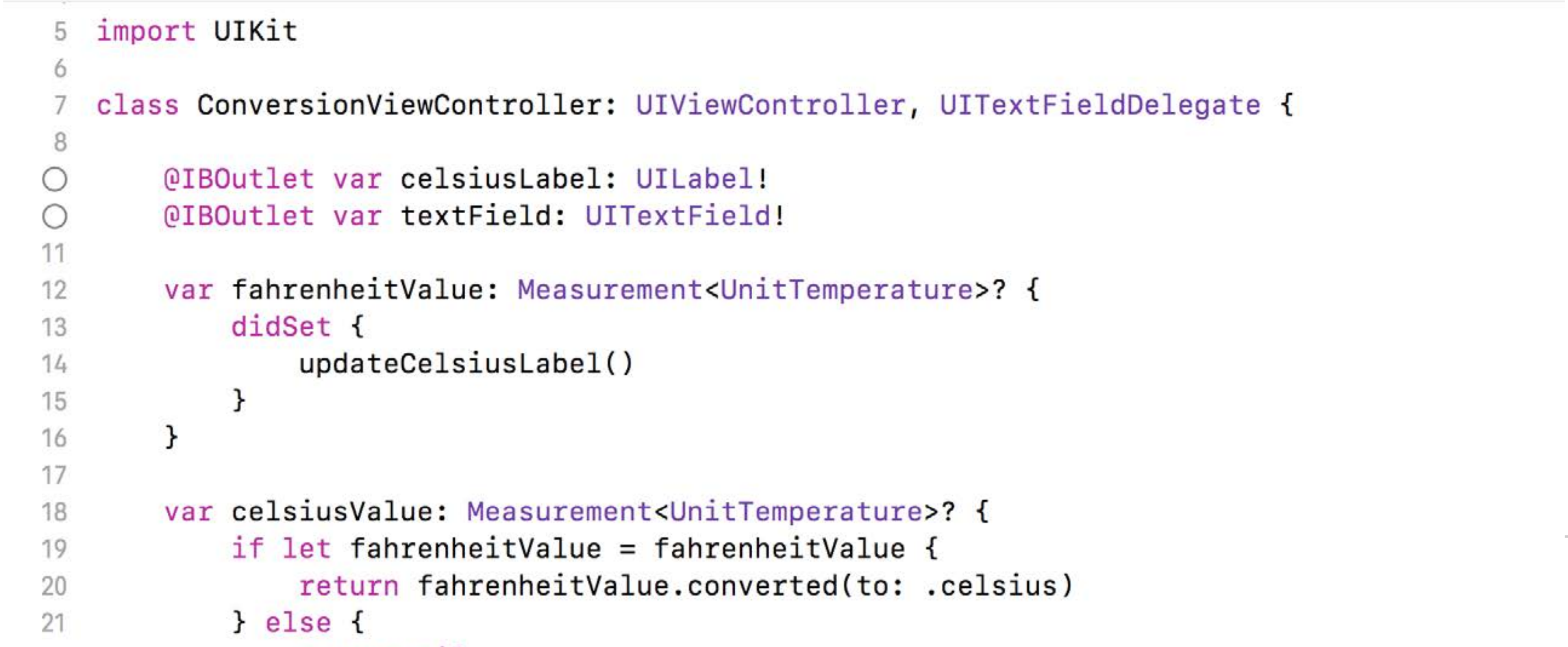




Implementing the Temperature Conversion code

Copy code from TemperatureConversionCode (folder)
Replace all the code in the ConversionViewcontroller.swift

Test and run the app. The temperature should be automatically converted from Fahrenheit to Celsius as you type in the values.

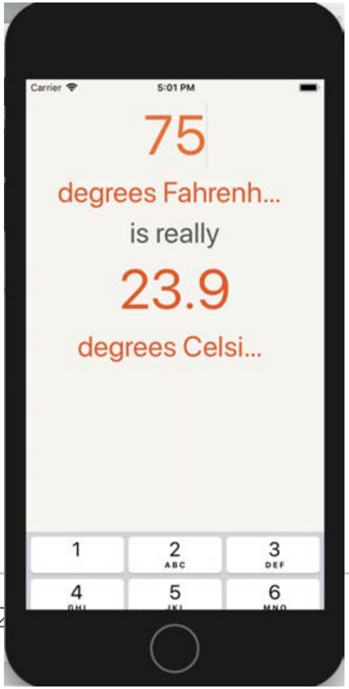


```swift
import UIKit

class ConversionViewController: UIViewController, UITextFieldDelegate {

    @IBOutlet var celsiusLabel: UILabel!
    @IBOutlet var textField: UITextField!

    var fahrenheitValue: Measurement<UnitTemperature>? {
        didSet {
            updateCelsiusLabel()
        }
    }

    var celsiusValue: Measurement<UnitTemperature>? {
        if let fahrenheitValue = fahrenheitValue {
            return fahrenheitValue.converted(to: .celsius)
        } else {
```

Implementing the Temperature Conversion code

Copy code from TemperatureConversionCode (folder)
Replace all the code in the ConversionViewcontroller.swift

Test and run the app. The temperature should be automatically converted from Fahrenheit to Celsius as you type in the values.


Carrier
5:01 PM
75
degrees Fahrenh...
is really
23.9
degrees Celsi...
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO