Project Bob

Swift 3.0 & Xcode 8.2.1

revised: Feb 14, 2017

Project Bob: Overview

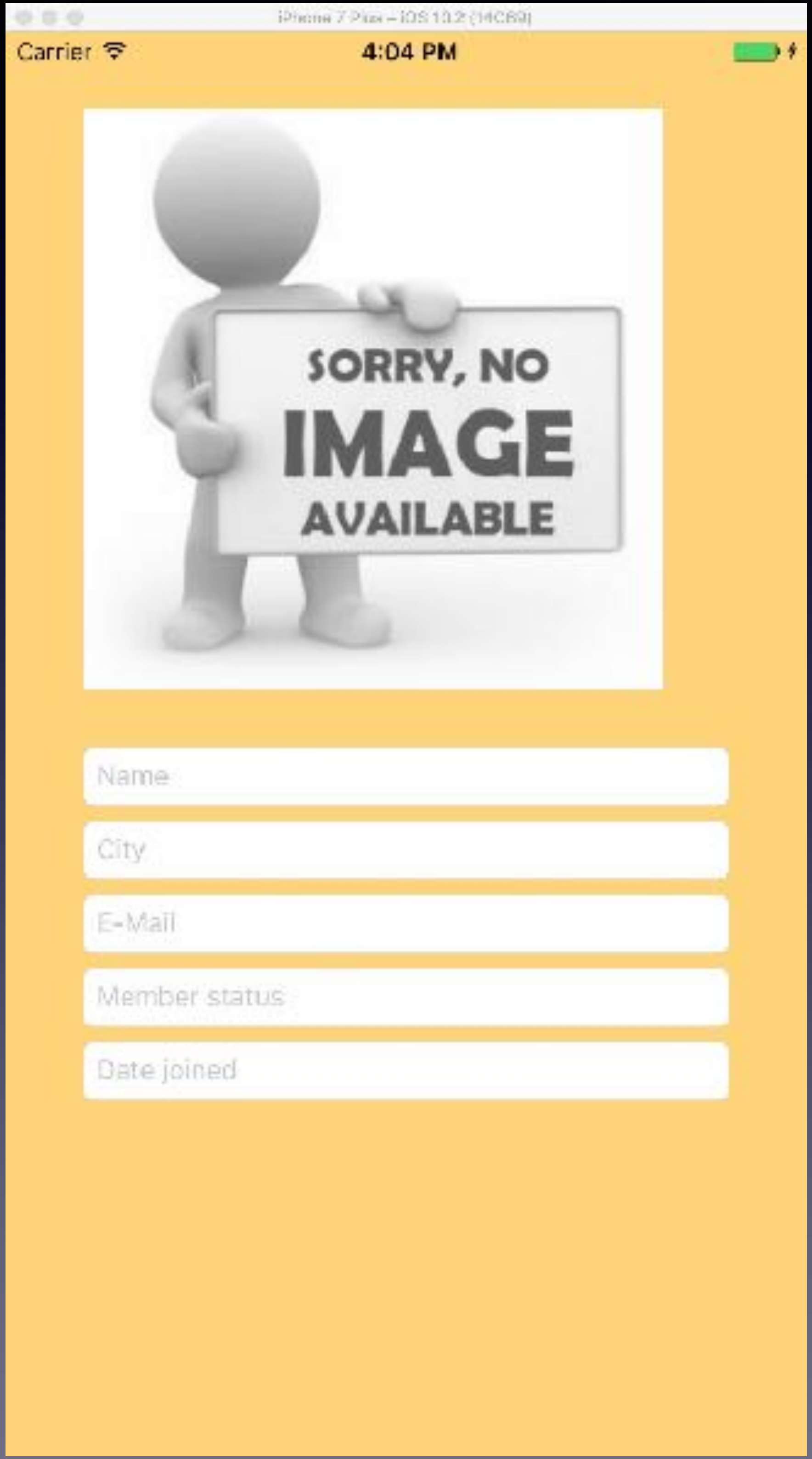
This is a three part project which covers a variety of SWIFT features in three phases

- **Bob 1** - Working with TextFields and Image Pickers to collect data for a person
- **Bob 2** - Adding person data to a Table View, editing it and managing the Table View
- **Bob 3** - Persisting the data using Core Data.

Data & Image Entry

Basic info entry to illustrate various techniques and issues that arise.

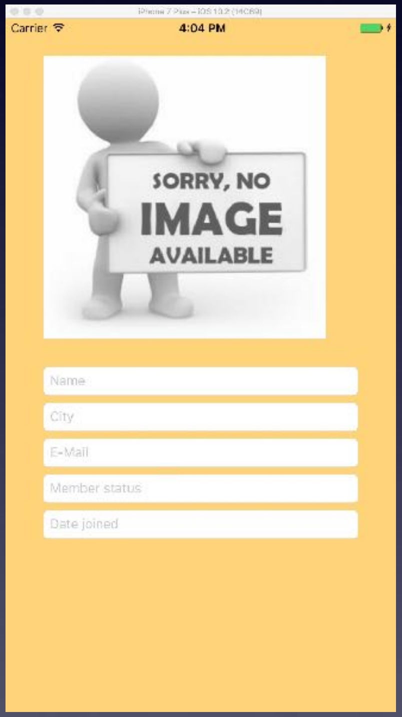
- Using the **Keyboard**; managing the keyboard.
- Using a **Picker View** to display choices; and put into a text box
- Using a **Date Picker** to set a date and put it into a text box
- Using **Image Picker Controller** to access camera/album images

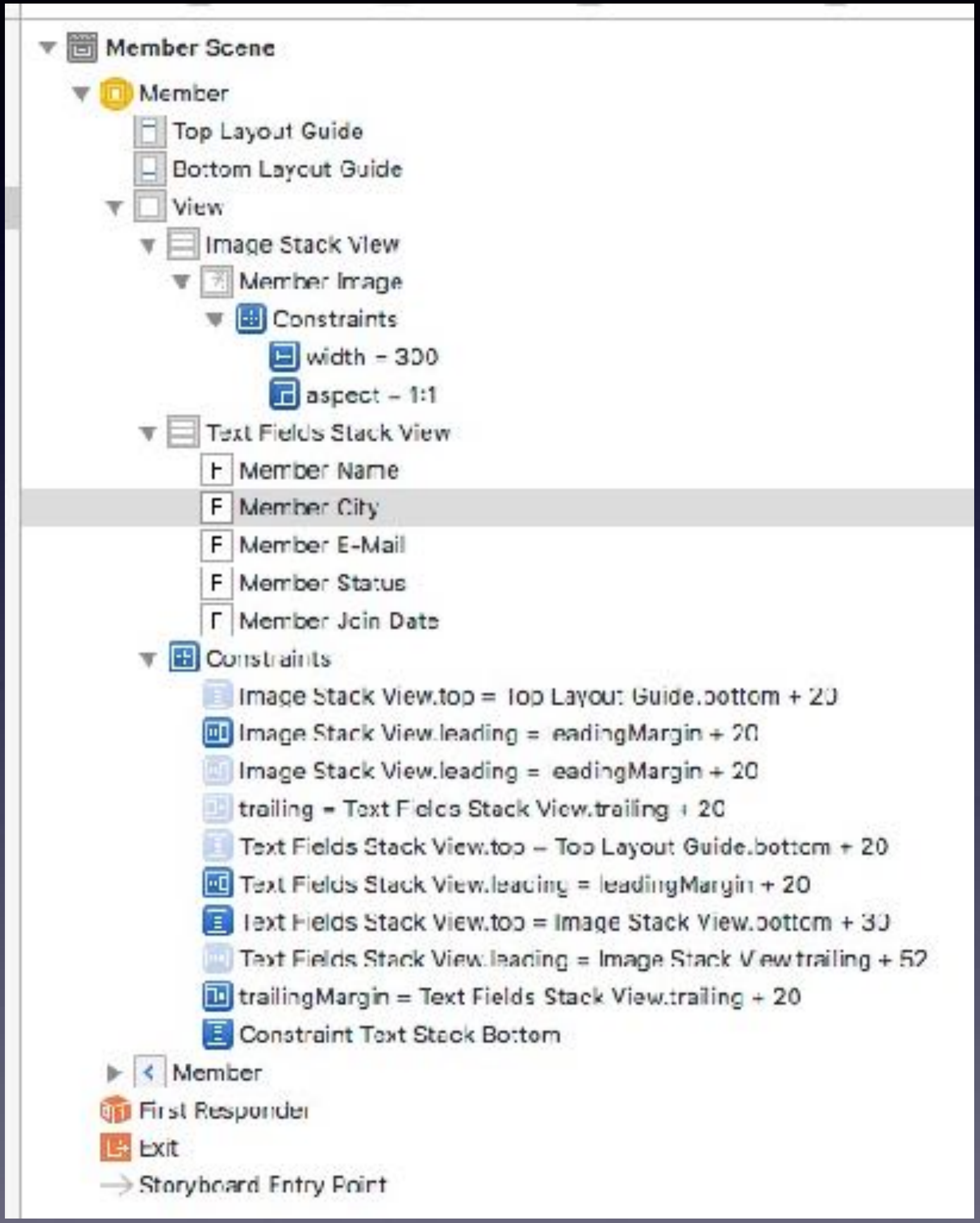


UI Design - starting point !

1. Add **Image View**
   This is where we will put a person's image.
2. Add 5 **Text Fields**
   This is where we enter additional info about the person.
3. Use **Stack View**

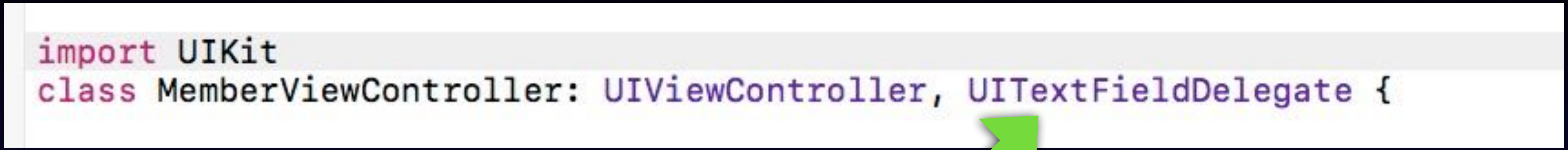
   To layout the UI
4. Add **Constraints**
5. Add **placeholder** names in TextFields
6. Add images to **Assets**.
   Set default image





```
▼ Member Scene
  ▼ Member
      Top Layout Guide
      Bottom Layout Guide
    ▼ View
      ▼ Image Stack View
        ▼ Member Image
          ▼ Constraints
              width = 300
              aspect = 1:1
      ▼ Text Fields Stack View
          Member Name
          Member City
          Member E-Mail
          Member Status
          Member Join Date
      ▼ Constraints
          Image Stack View.top = Top Layout Guide.bottom + 20
          Image Stack View.leading = leadingMargin + 20
          Image Stack View.leading = leadingMargin + 20
          trailing = Text Fields Stack View.trailing + 20
          Text Fields Stack View.top = Top Layout Guide.bottom + 20
          Text Fields Stack View.leading = leadingMargin + 20
          Text Fields Stack View.top = Image Stack View.bottom + 30
          Text Fields Stack View.leading = Image Stack View.trailing + 52
          trailingMargin = Text Fields Stack View.trailing + 20
          Constraint Text Stack Bottom
  ▶ Member
    First Responder
    Exit
    Storyboard Entry Point
```

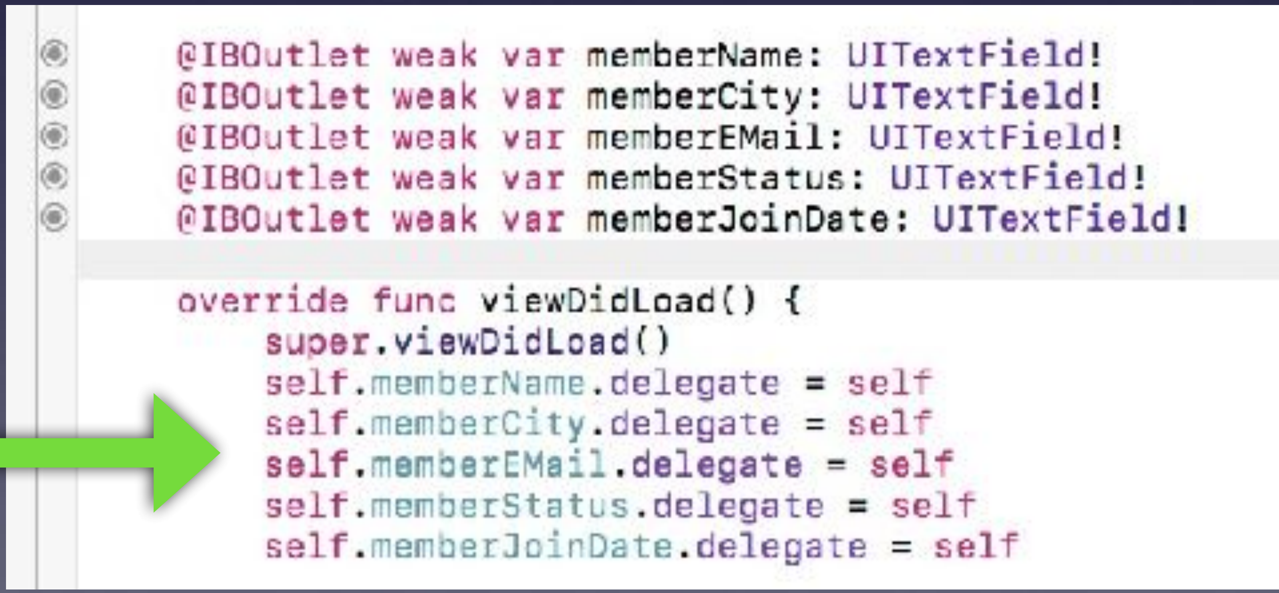
Text Field



```swift
import UIKit
class MemberViewController: UIViewController, UITextFieldDelegate {
```

- Add the Text Field protocols: **UITextFieldDelegate**
  The UITextFieldDelegate protocol defines methods that you use to manage the editing and validation of text in a UITextField object. All of the methods of this protocol are optional.

- Outlets for Text Fields to the **MemberViewController**
  Controller's link to elements of the UI.

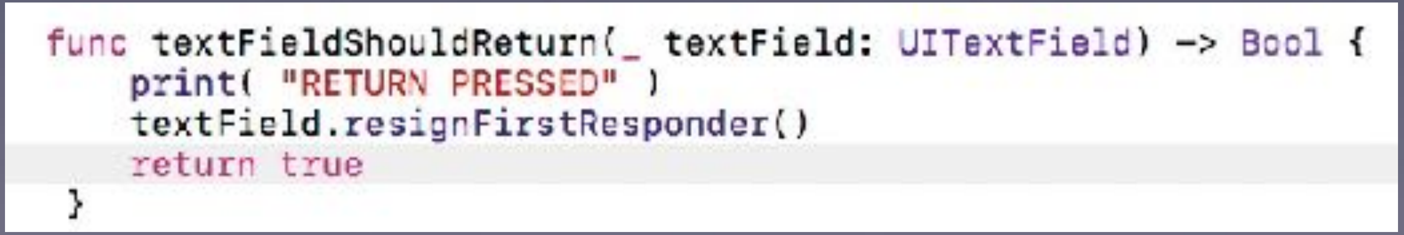


```swift
    @IBOutlet weak var memberName: UITextField!
    @IBOutlet weak var memberCity: UITextField!
    @IBOutlet weak var memberEMail: UITextField!
    @IBOutlet weak var memberStatus: UITextField!
    @IBOutlet weak var memberJoinDate: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        self.memberName.delegate = self
        self.memberCity.delegate = self
        self.memberEMail.delegate = self
        self.memberStatus.delegate = self
        self.memberJoinDate.delegate = self
```

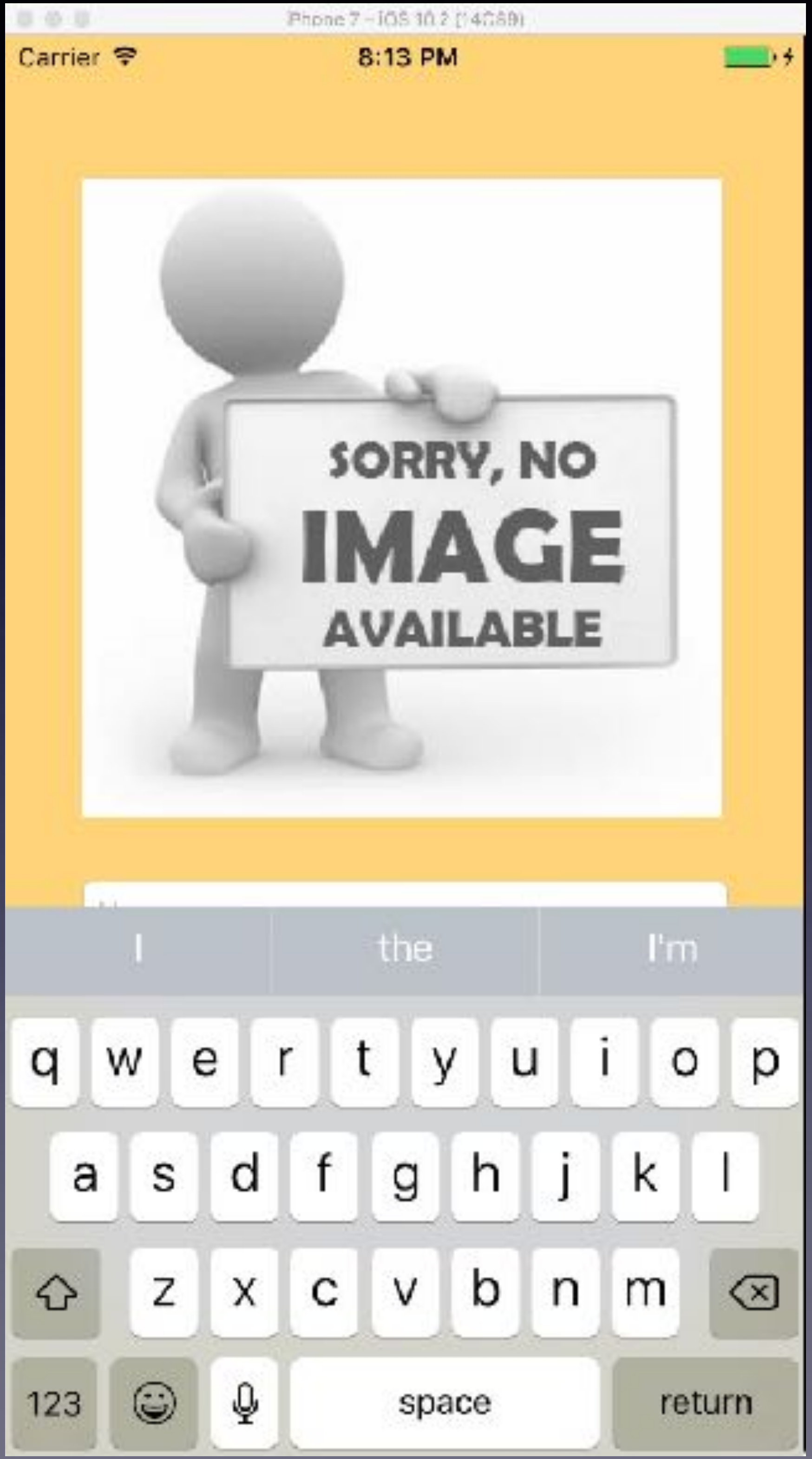
- Initialize delegates
  A **text field delegate** responds to editing-related messages from the text field. You can use the delegate to respond to the text entered by the user and to some special commands, such as when the return button is pressed.

Keyboard *appears - removed*

- **Keyboard** appears when Text Field gets focus, i.e., becomes **first responder**
- To remove the Keyboard we must **resignFirstResponder** status for that Text Field.
- Use **textFieldShouldReturn** to trigger the return key press event.



```
func textFieldShouldReturn(_ textField: UITextField) -> Bool {
    print( "RETURN PRESSED" )
    textField.resignFirstResponder()
    return true
}
```

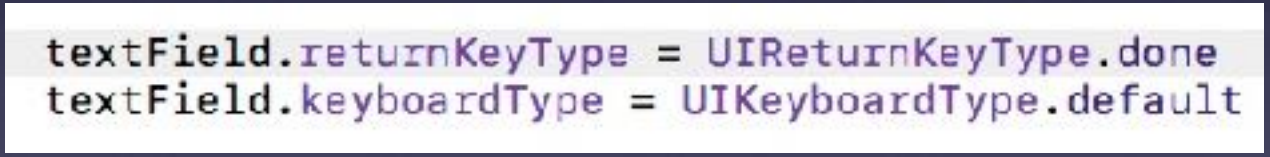


textFieldDidBeginEditing

Tells the delegate that editing began in the specified text field.

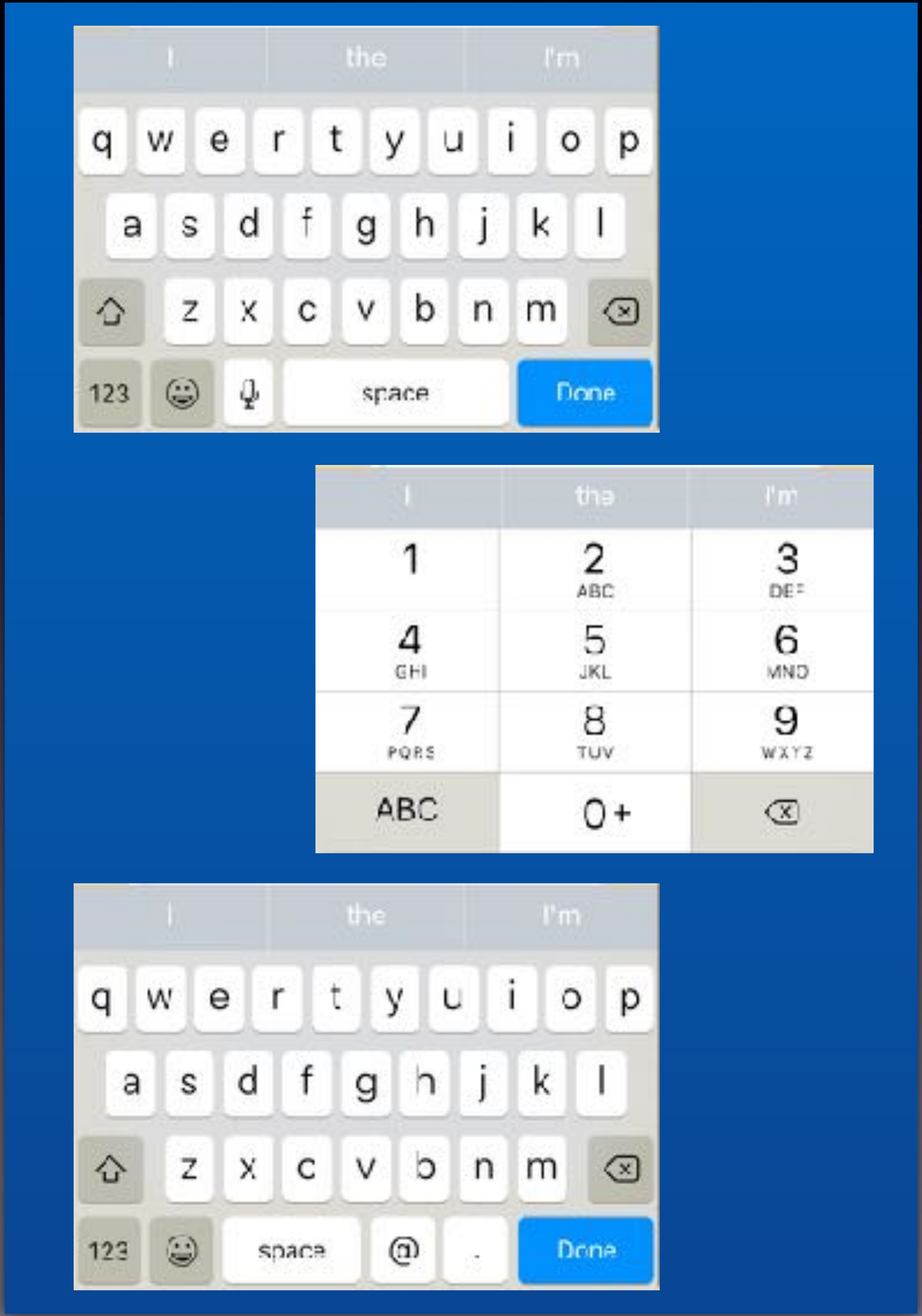
Actions to do here !

- Set text for Return Key
- Select type of Keyboard to display for each textField



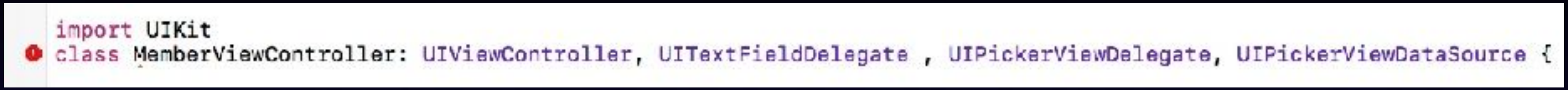
```
textField.returnKeyType = UIReturnKeyType.done
textField.keyboardType = UIKeyboardType.default
```

- Move textFields out of the way of Keyboard.
- Indicate that text is in “edit mode”



Picker View

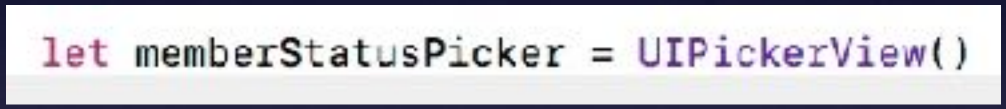
as Keyboard



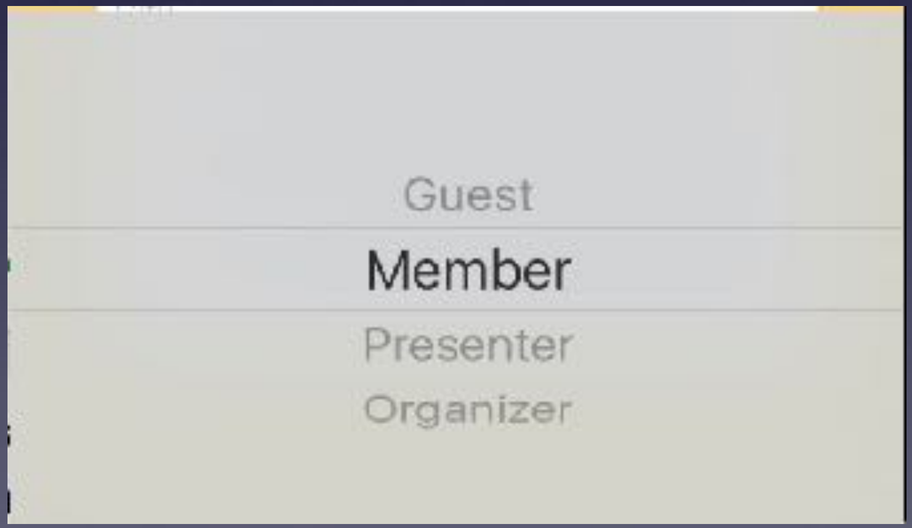
```
import UIKit
class MemberViewController: UIViewController, UITextFieldDelegate , UIPickerViewDelegate, UIPickerViewDataSource {
```

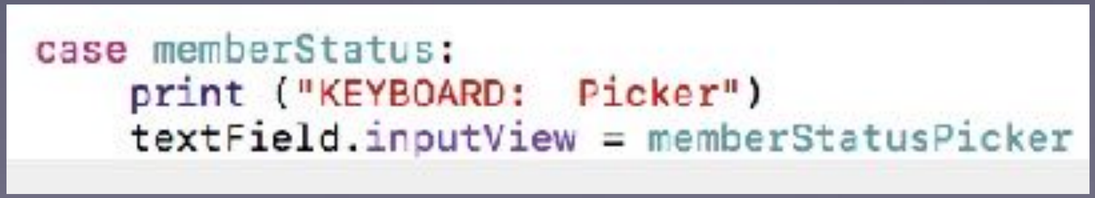
<u>**What needs to be done?**</u>

- Add protocols
  (delegate and source) to support the UIPickerView; error pending conforming to protocol.
- Add UIPickerView
- Initialize delegates
- Create array of picker options
- Set **textField.inputView** = picker view



```
let memberStatusPicker = UIPickerView()
```

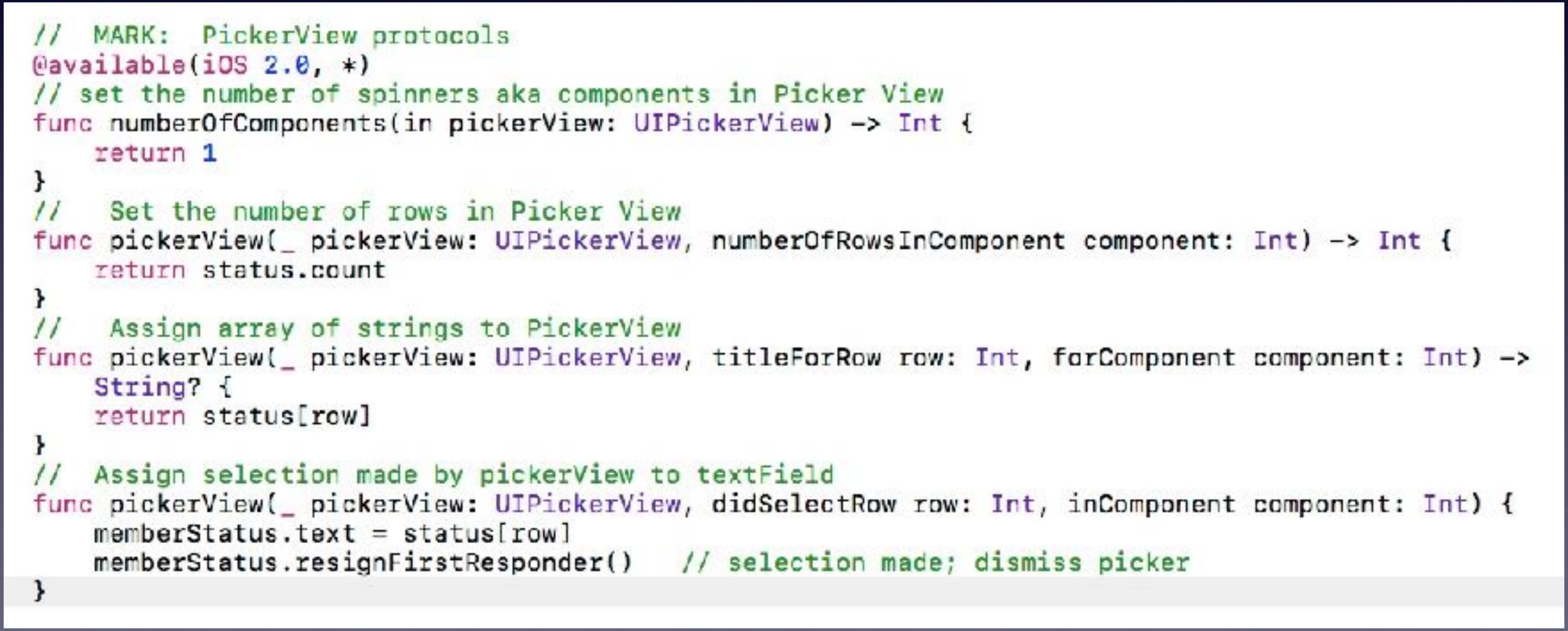




```
case memberStatus:
    print ("KEYBOARD:  Picker")
    textField.inputView = memberStatusPicker
```

PickerView Protocols

These protocols define the type and content of the Picker View display.

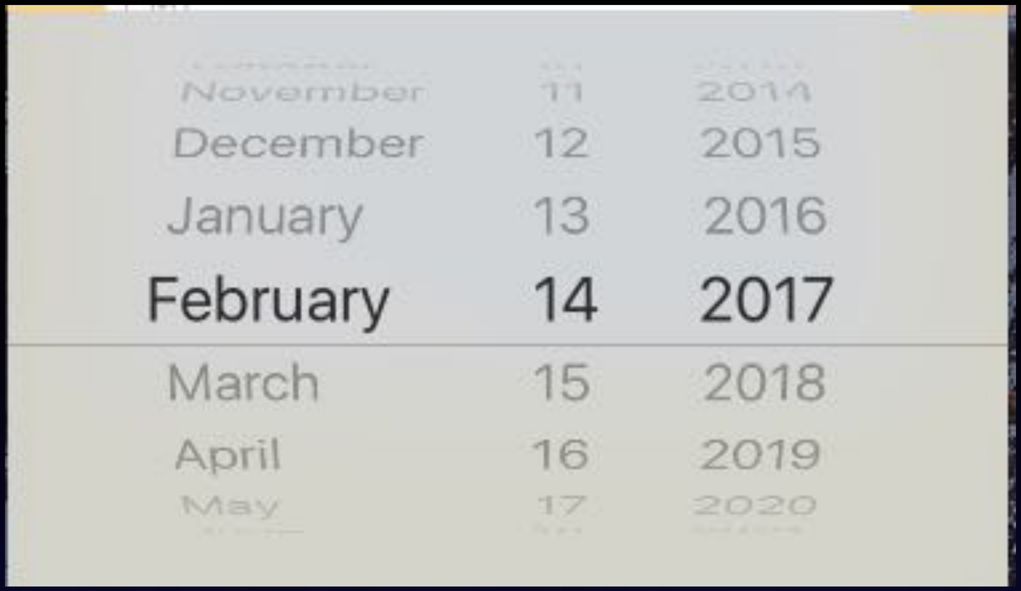


```swift
//  MARK:  PickerView protocols
@available(iOS 2.0, *)
// set the number of spinners aka components in Picker View
func numberOfComponents(in pickerView: UIPickerView) -> Int {
    return 1
}
//   Set the number of rows in Picker View
func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
    return status.count
}
//   Assign array of strings to PickerView
func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) ->
    String? {
    return status[row]
}
//  Assign selection made by pickerView to textField
func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int) {
    memberStatus.text = status[row]
    memberStatus.resignFirstResponder()   // selection made; dismiss picker
}
```

The ‘didSelectRow’ protocol is used to update the value selected and to dismiss the pickerView.

Date Picker View

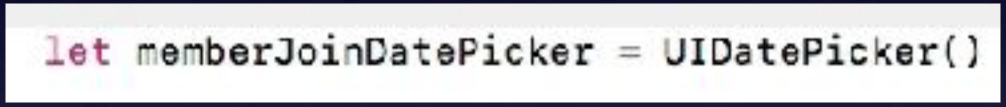
as Keyboard



<u>What needs to be done?</u>

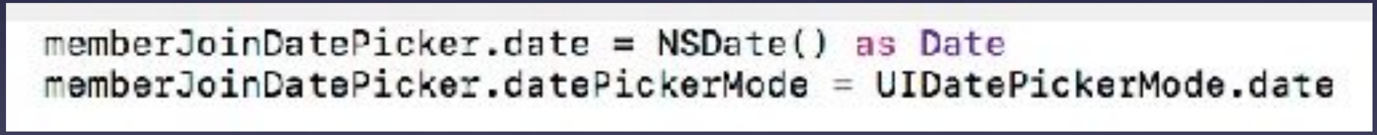
- Protocols for the Date Picker
  do not need to be added; are part of the UIDatePicker instance.

- Add UIDatePickerView



```
let memberJoinDatePicker = UIDatePicker()
```

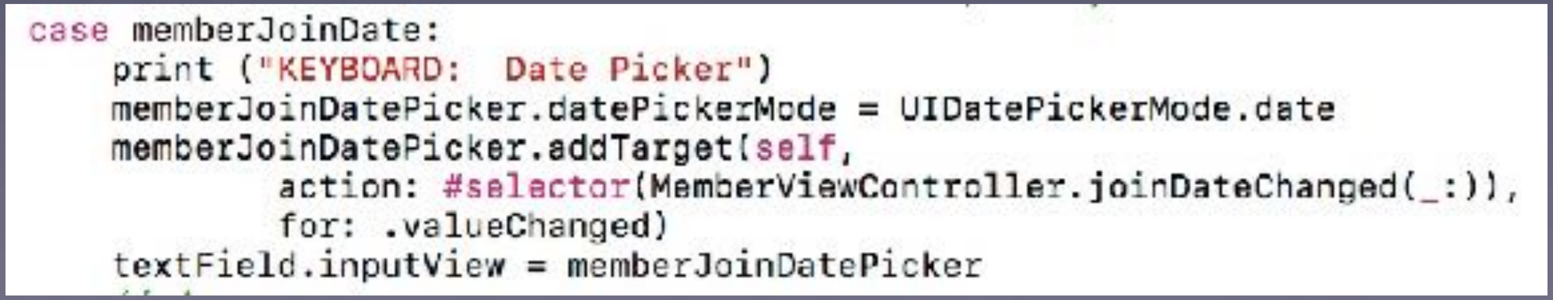
- Initialize dataPicker
  set date for picker; set display type.



```
memberJoinDatePicker.date = NSDate() as Date
memberJoinDatePicker.datePickerMode = UIDatePickerMode.date
```

- Add func **joinDateChanged**
  a function which is executed when the date is changed

- Set **textField.inputView**
  date picker



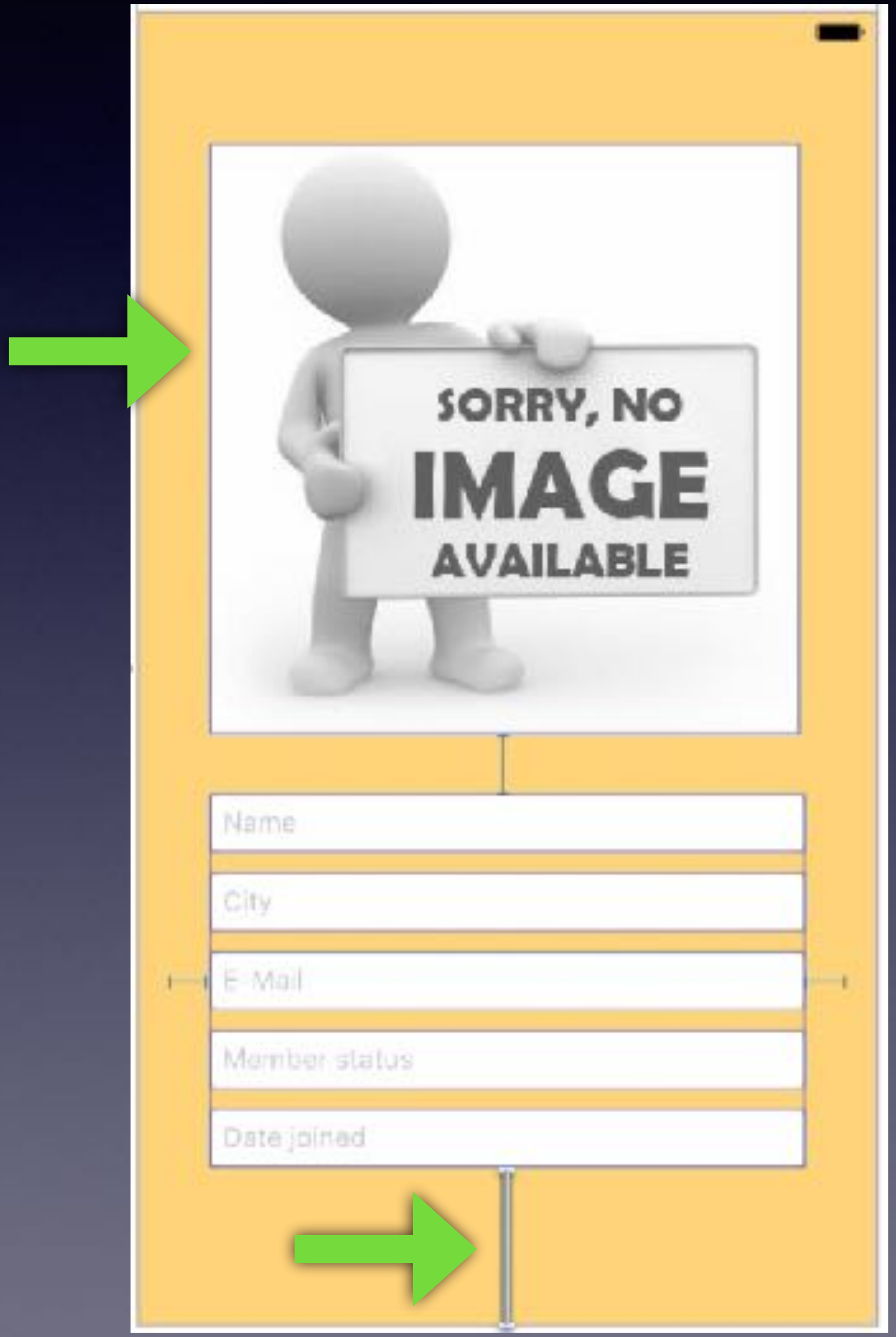
```
case memberJoinDate:
    print ("KEYBOARD:  Date Picker")
    memberJoinDatePicker.datePickerMode = UIDatePickerMode.date
    memberJoinDatePicker.addTarget(self,
            action: #selector(MemberViewController.joinDateChanged(_:)),
            for: .valueChanged)
    textField.inputView = memberJoinDatePicker
```

Conflict with Keyboard

Problem: Keyboard blocks some textFields.

Create Outlets
1) for the constraint of the Text Stack to Bottom Layer Guide.
2) for the image view.

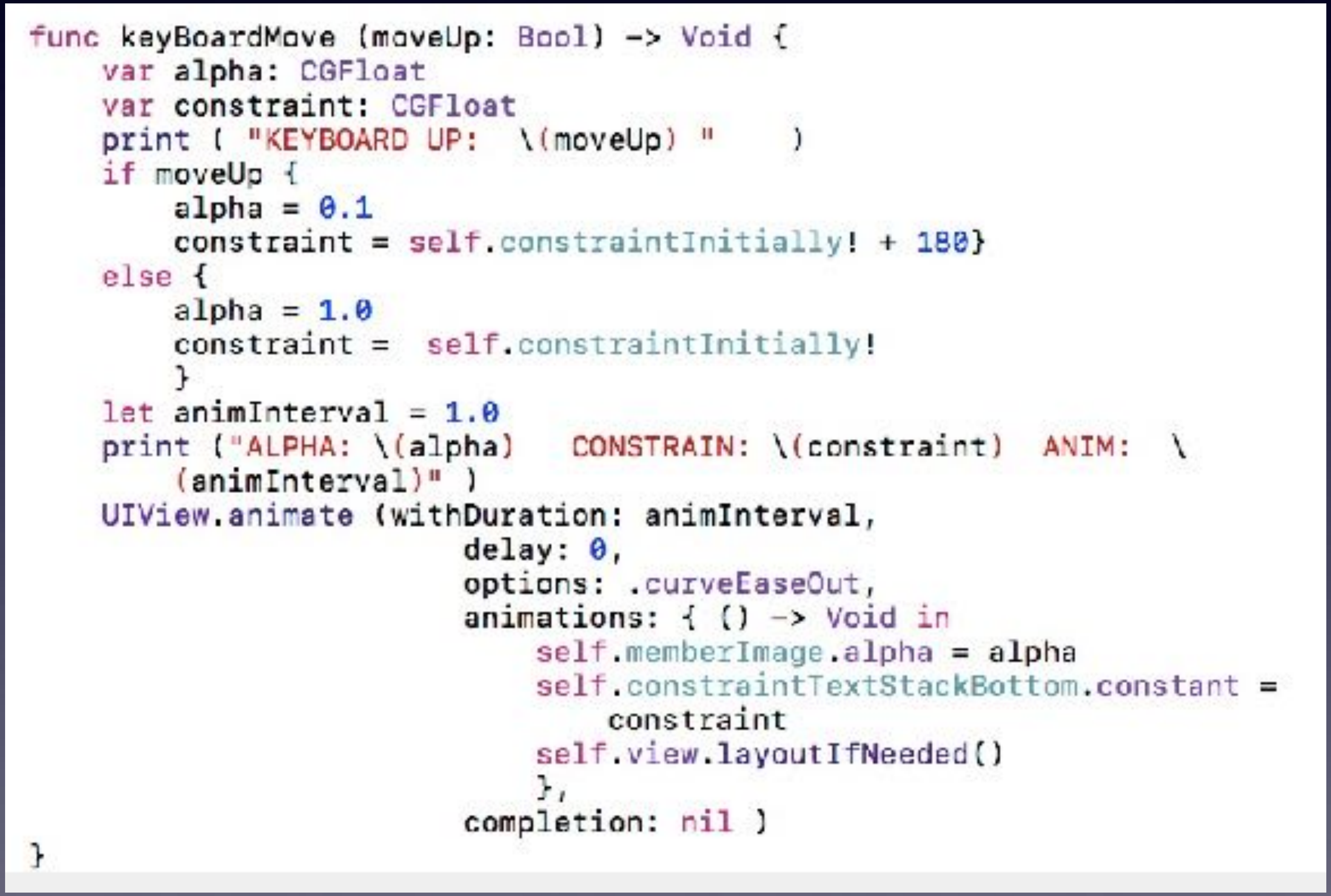
We will use these to fade the image and to move the text Fields up (out of the way of the Keyboard)



Animate for Keyboard

- To clear the way for the Keyboard we fade the image and move the text fields by 180

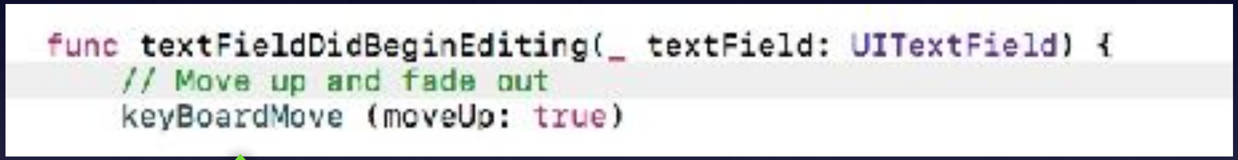
- When the Keyboard is dismissed we reverse the animation.



```
func keyBoardMove (moveUp: Bool) -> Void {
    var alpha: CGFloat
    var constraint: CGFloat
    print ( "KEYBOARD UP:  \(moveUp) "     )
    if moveUp {
        alpha = 0.1
        constraint = self.constraintInitially! + 180}
    else {
        alpha = 1.0
        constraint =  self.constraintInitially!
        }
    let animInterval = 1.0
    print ("ALPHA: \(alpha)   CONSTRAIN: \(constraint)  ANIM:  \
        (animInterval)" )
    UIView.animate (withDuration: animInterval,
                              delay: 0,
                              options: .curveEaseOut,
                              animations: { () -> Void in
                                  self.memberImage.alpha = alpha
                                  self.constraintTextStackBottom.constant =
                                      constraint
                                  self.view.layoutIfNeeded()
                                  },
                              completion: nil )
}
```

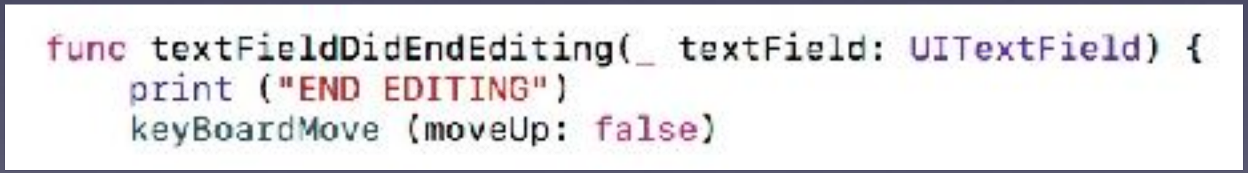
Some Problems

We initiate the move with…



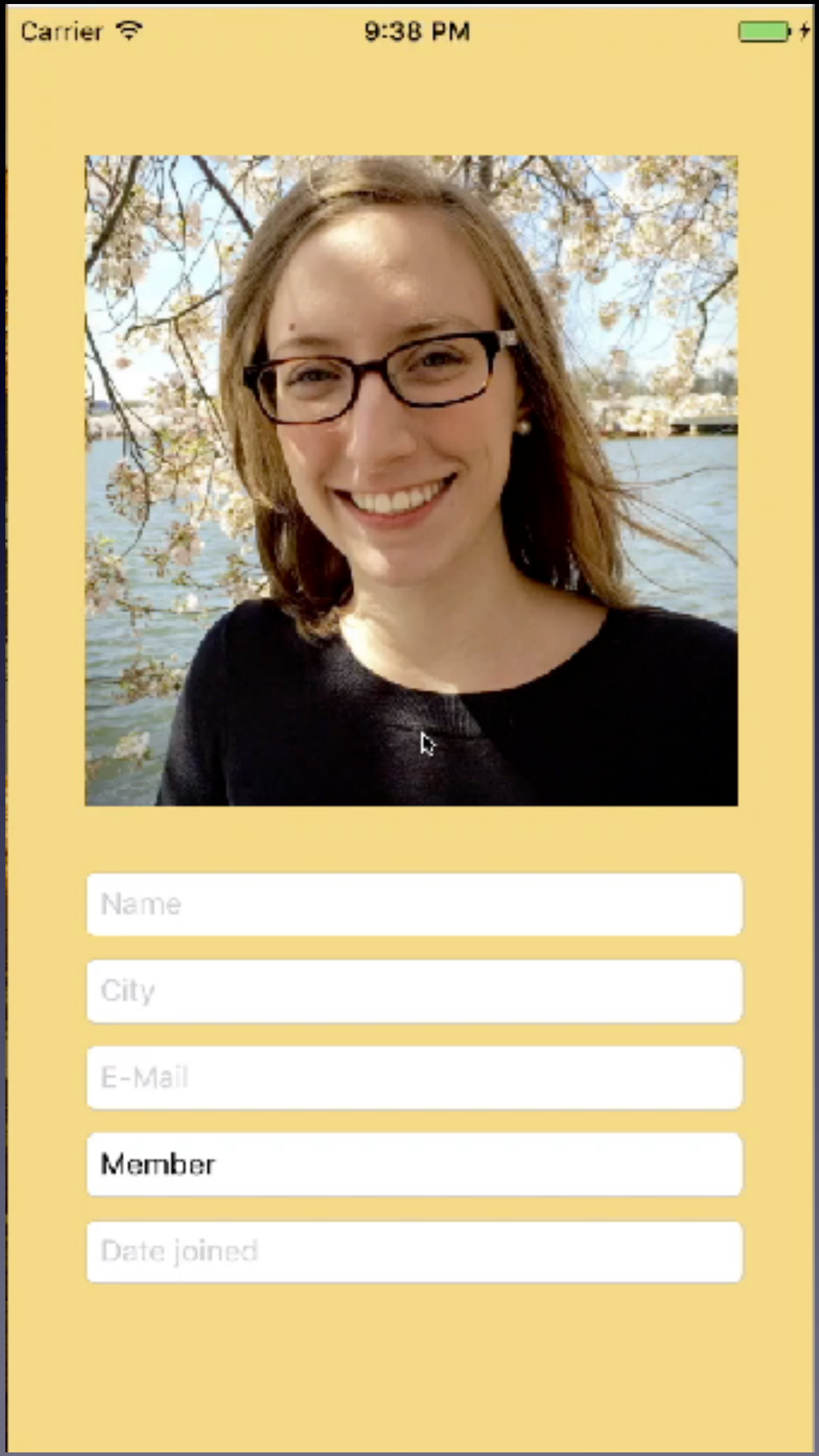
```
func textFieldDidBeginEditing(_ textField: UITextField) {
    // Move up and fade out
    keyBoardMove (moveUp: true)
```

We restore the screen with…



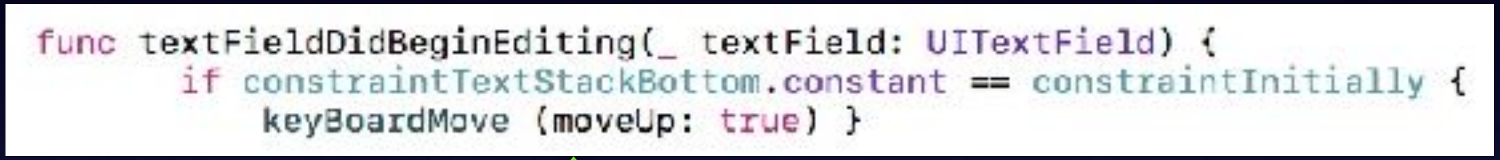
```
func textFieldDidEndEditing(_ textField: UITextField) {
    print ("END EDITING")
    keyBoardMove (moveUp: false)
```

PROBLEM: Fade in - out.



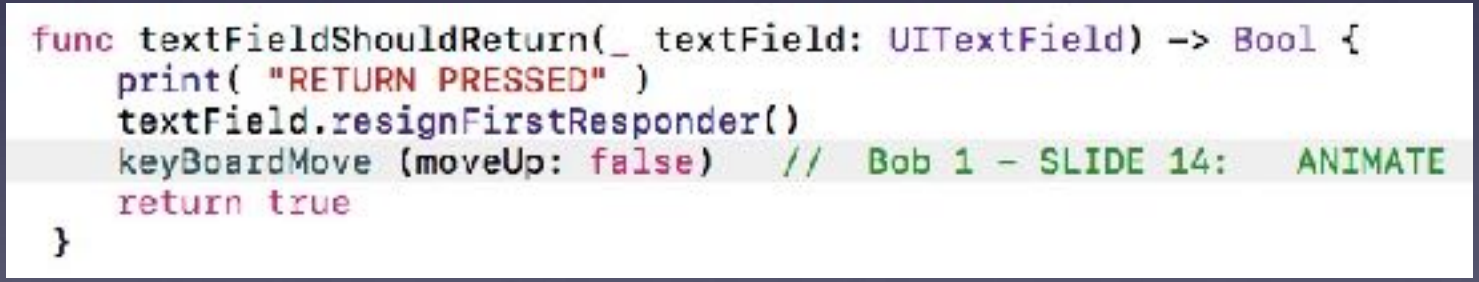
Try to Fix

Only move up if needed…



```
func textFieldDidBeginEditing(_ textField: UITextField) {
        if constraintTextStackBottom.constant == constraintInitially {
            keyBoardMove (moveUp: true) }
```

Put downward move into **textFieldShouldReturn**.



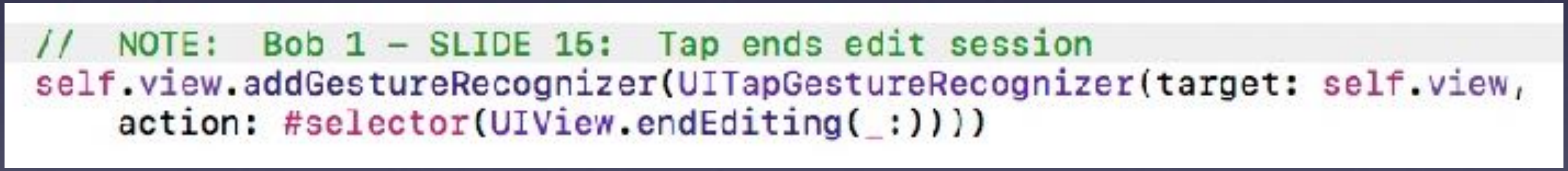
```
func textFieldShouldReturn(_ textField: UITextField) -> Bool {
    print( "RETURN PRESSED" )
    textField.resignFirstResponder()
    keyBoardMove (moveUp: false)   //  Bob 1 - SLIDE 14:   ANIMATE
    return true
 }
```

But that still leaves some gaps

Tap Gesture

- We want to resignFirstResponder status whenever we select anything other than a textField.

- We can accomplish this by adding a **addGestureRecognizer**



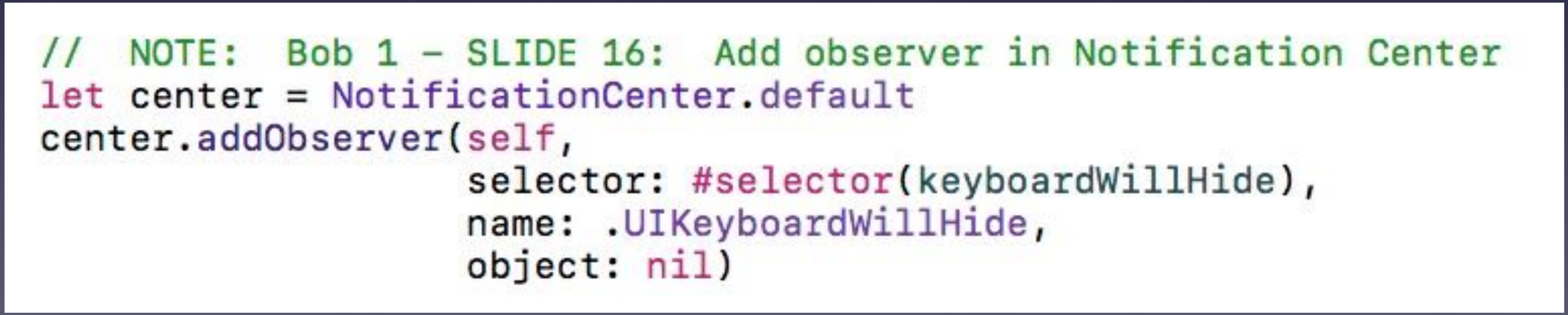
```
//  NOTE:  Bob 1 - SLIDE 15:  Tap ends edit session
self.view.addGestureRecognizer(UITapGestureRecognizer(target: self.view,
    action: #selector(UIView.endEditing(_:))))
```

- **.endEditing**
  Causes the view (or one of its embedded text fields) to resign the first responder status.

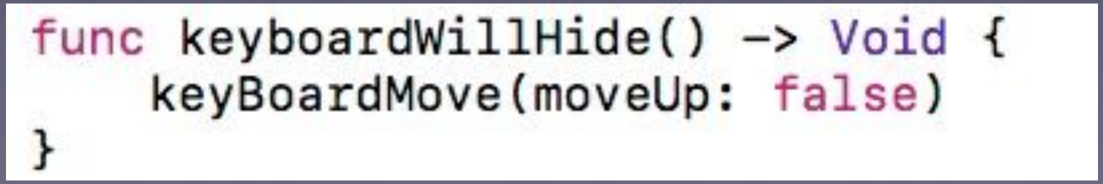
Notification

UIKeyboardWillHide - is a notification which is Posted immediately prior to the dismissal of the keyboard.

- By subscribing to this notification we can respond by restoring the position of the textFields and Image.



```
//  NOTE:  Bob 1 - SLIDE 16:  Add observer in Notification Center
let center = NotificationCenter.default
center.addObserver(self,
                   selector: #selector(keyboardWillHide),
                   name: .UIKeyboardWillHide,
                   object: nil)
```

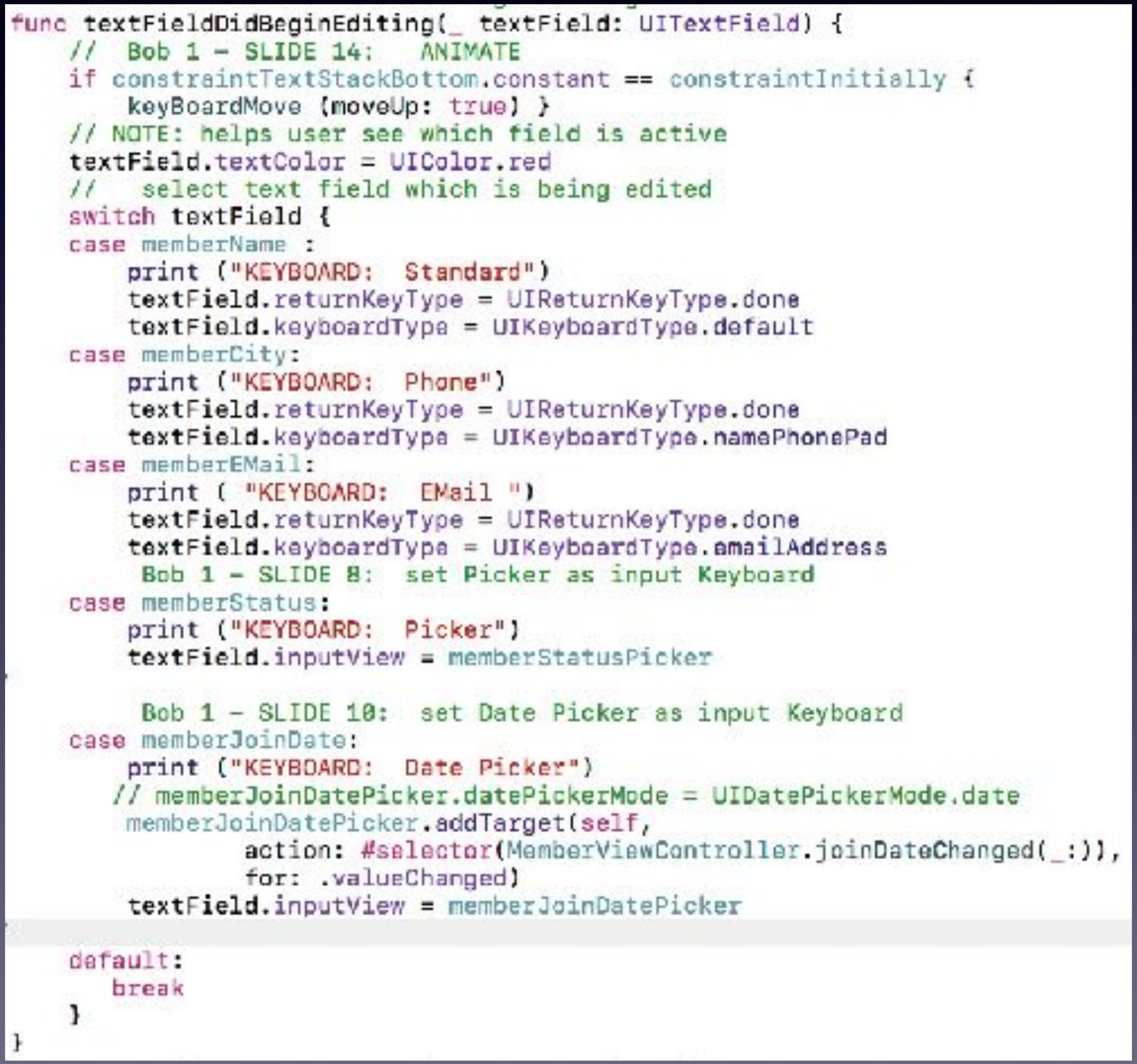


```
func keyboardWillHide() -> Void {
    keyBoardMove(moveUp: false)
}
```

textFieldDidBeginEditing

revisited

- A **switch** can select among the various types of keyboards to display.
- We initiate the move of the textFields here only if they were not moved yet.
- We change text color to red to indicate it is being edited.



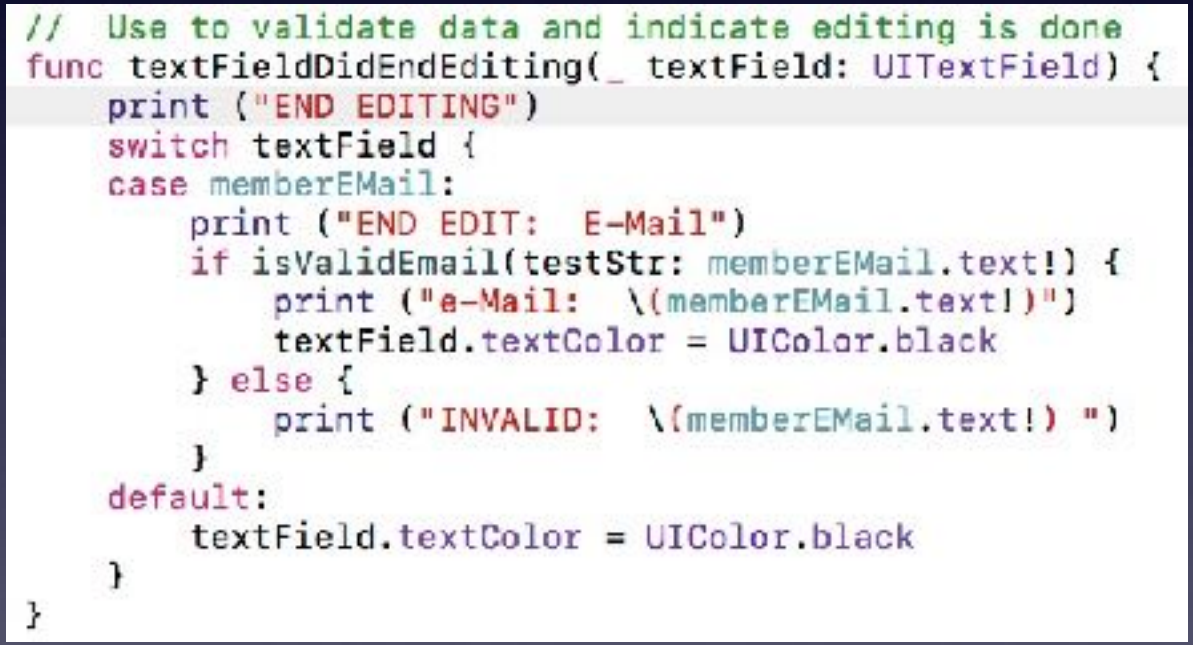
```
func textFieldDidBeginEditing(_ textField: UITextField) {
    //  Bob 1 - SLIDE 14:   ANIMATE
    if constraintTextStackBottom.constant == constraintInitially {
        keyBoardMove (moveUp: true) }
    // NOTE: helps user see which field is active
    textField.textColor = UIColor.red
    //   select text field which is being edited
    switch textField {
    case memberName :
        print ("KEYBOARD:  Standard")
        textField.returnKeyType = UIReturnKeyType.done
        textField.keyboardType = UIKeyboardType.default
    case memberCity:
        print ("KEYBOARD:  Phone")
        textField.returnKeyType = UIReturnKeyType.done
        textField.keyboardType = UIKeyboardType.namePhonePad
    case memberEMail:
        print ( "KEYBOARD:  EMail ")
        textField.returnKeyType = UIReturnKeyType.done
        textField.keyboardType = UIKeyboardType.emailAddress
         Bob 1 - SLIDE 8:  set Picker as input Keyboard
    case memberStatus:
        print ("KEYBOARD:  Picker")
        textField.inputView = memberStatusPicker

         Bob 1 - SLIDE 10:  set Date Picker as input Keyboard
    case memberJoinDate:
        print ("KEYBOARD:  Date Picker")
       // memberJoinDatePicker.datePickerMode = UIDatePickerMode.date
        memberJoinDatePicker.addTarget(self,
                action: #selector(MemberViewController.joinDateChanged(_:)),
                for: .valueChanged)
        textField.inputView = memberJoinDatePicker

    default:
       break
    }
}
```

textFieldDidEndEditing

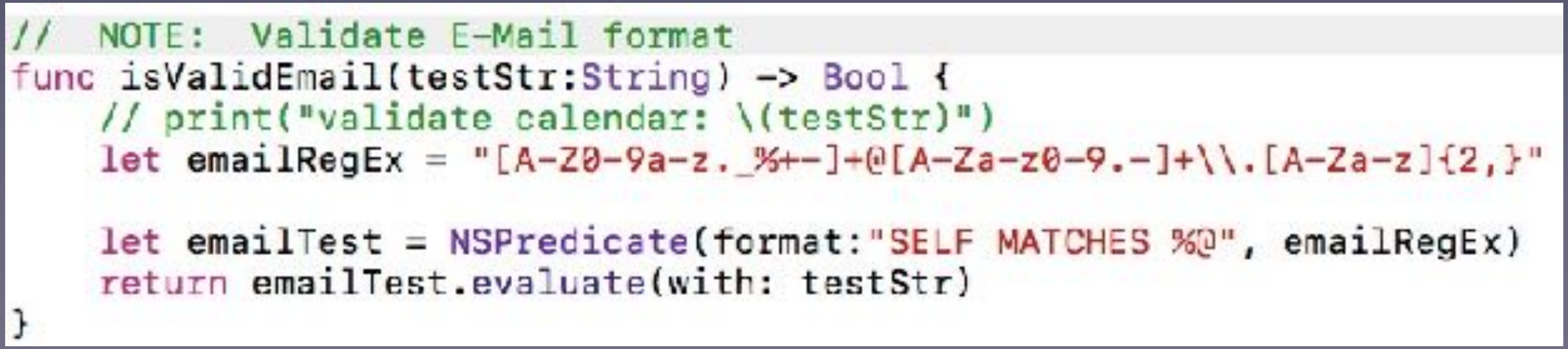
This is the place to clean up when you are done editing.

- Validate content of textField, i.e., make sure E-Mail has correct format.

- Indicate 'edit' session is completed, e.g., change text color from red to black.



```
//  Use to validate data and indicate editing is done
func textFieldDidEndEditing(_ textField: UITextField) {
    print ("END EDITING")
    switch textField {
    case memberEMail:
        print ("END EDIT:  E-Mail")
        if isValidEmail(testStr: memberEMail.text!) {
            print ("e-Mail:  \(memberEMail.text!)")
            textField.textColor = UIColor.black
        } else {
            print ("INVALID:  \(memberEMail.text!) ")
        }
    default:
        textField.textColor = UIColor.black
    }
}
```

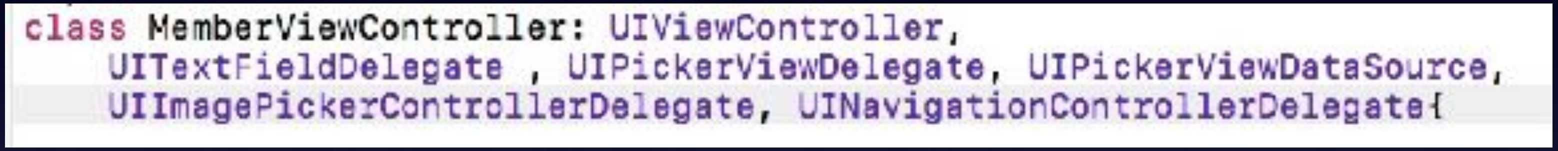


```
//  NOTE:  Validate E-Mail format
func isValidEmail(testStr:String) -> Bool {
    // print("validate calendar: \(testStr)")
    let emailRegEx = "[A-Z0-9a-z._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}"

    let emailTest = NSPredicate(format:"SELF MATCHES %@", emailRegEx)
    return emailTest.evaluate(with: testStr)
}
```

Add Image

from Album

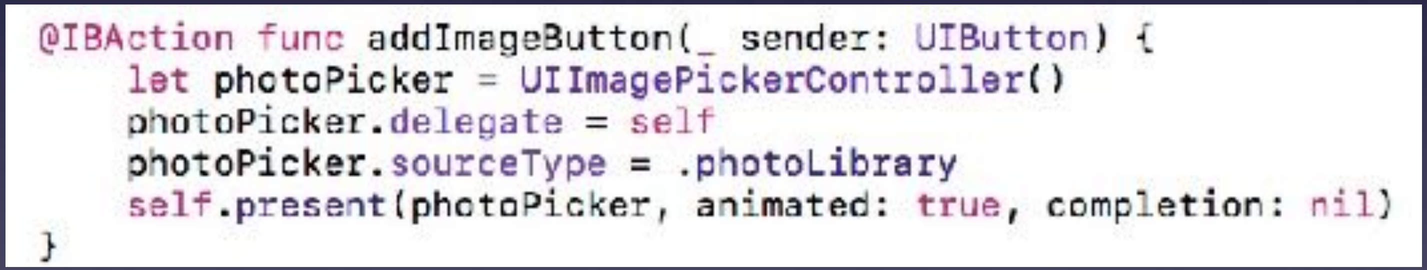
Actions to implement:



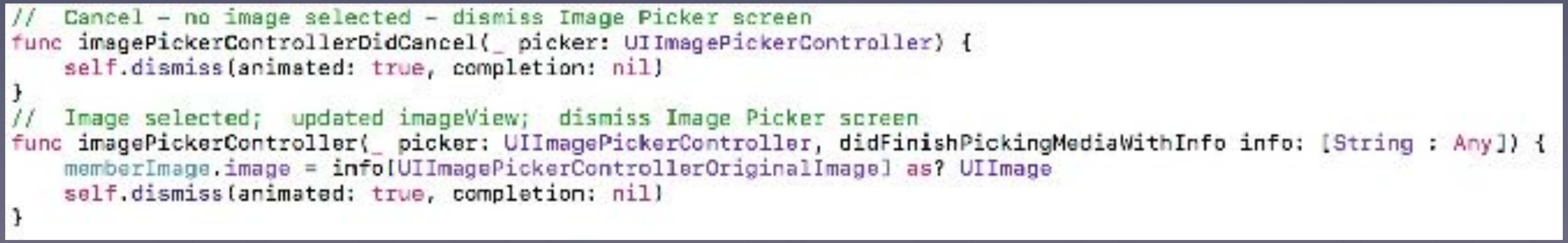
```
class MemberViewController: UIViewController,
    UITextFieldDelegate , UIPickerViewDelegate, UIPickerViewDataSource,
    UIImagePickerControllerDelegate, UINavigationControllerDelegate{
```

- Add Protocols:
  - **UIImagePickerControllerDelegate**
  - **UINavigationControllerDelegate**
- Add image Action
  addImageButton
  - Add instance of:
    UIImagePickerController
  - Set **sourceType = .photoLibrary**
  - **self.present**
    Presents a view controller modally.
- Add func
  to handle events of

  UIImagePickerController



```
@IBAction func addImageButton(_ sender: UIButton) {
    let photoPicker = UIImagePickerController()
    photoPicker.delegate = self
    photoPicker.sourceType = .photoLibrary
    self.present(photoPicker, animated: true, completion: nil)
}
```



```
//  Cancel - no image selected - dismiss Image Picker screen
func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
    self.dismiss(animated: true, completion: nil)
}
//  Image selected;  updated imageView;  dismiss Image Picker screen
func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [String : Any]) {
    memberImage.image = info[UIImagePickerControllerOriginalImage] as? UIImage
    self.dismiss(animated: true, completion: nil)
}
```

Quantum Films
s o f t w a r e

presenter Emil Safier

@EmilSafier
emil535@Gmail.com

Version 1.0, published to GitHub on 2/14/2017