

# **Unit 4—Lesson 4:**

## **Scroll Views**

# Scroll views



# UIScrollView

- For displaying more content than can fit on the screen
- Users scroll within the content by making swiping gestures
- Content can optionally be zoomed with a pinch gesture
- UIScrollView needs to know the size of the content

# Scroll views

First Name

First Name

Last Name

Last Name

Address Line 1

Address Line 1

Address Line 2

Address Line 2

City

City

State

State

Zip Code

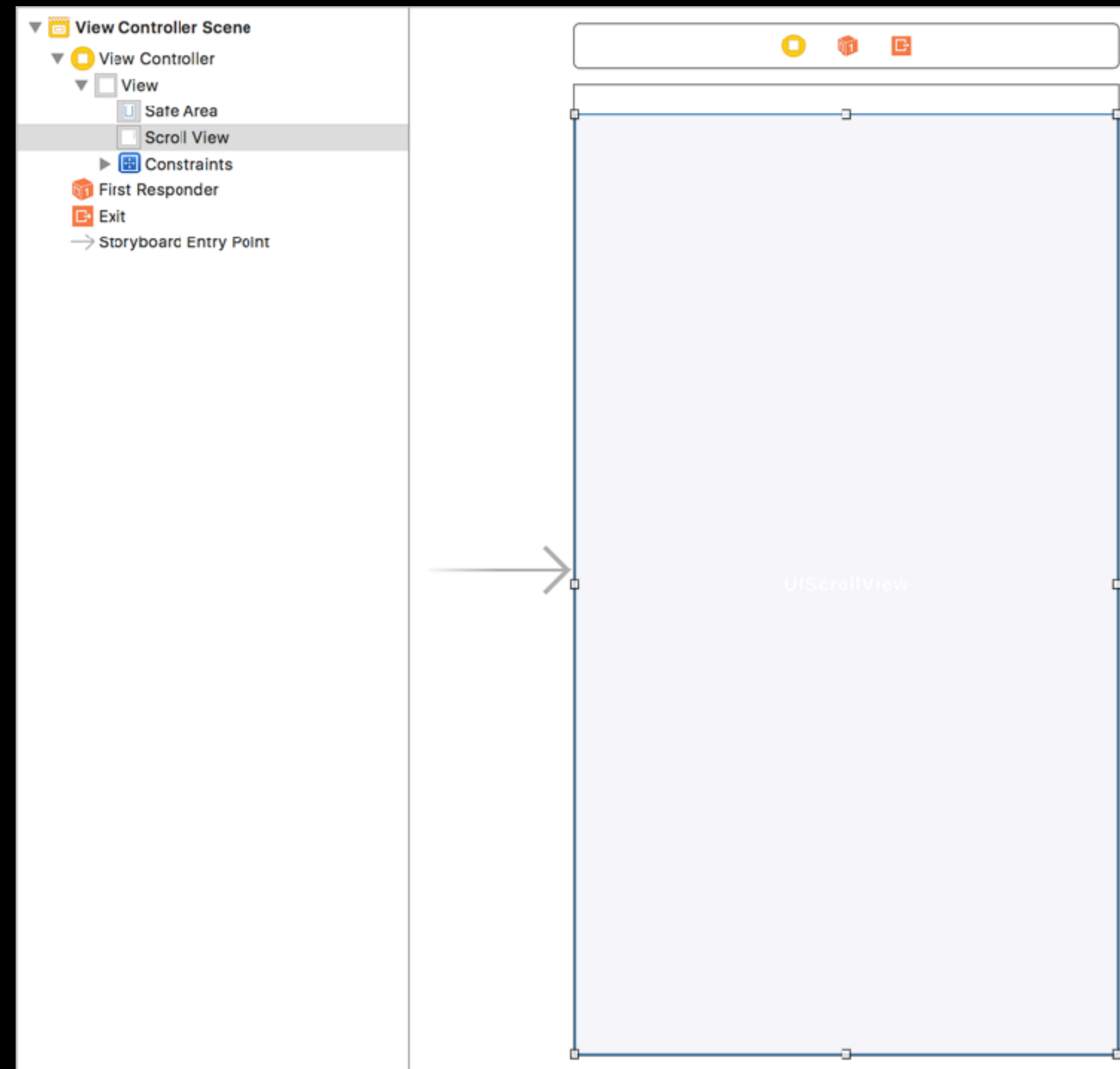
Zip Code

Phone Number

Phone Number

# Scroll views in Interface Builder

## Define scroll view frame



# Scroll views in Interface Builder

## Add constraints

**Add New Constraints**

0

0 0

0

Spacing to nearest neighbor

☐ Constrain to margins

---

☐ Width 375

☐ Height 667

---

☐ Equal Widths

☐ Equal Heights

☐ Aspect Ratio

---

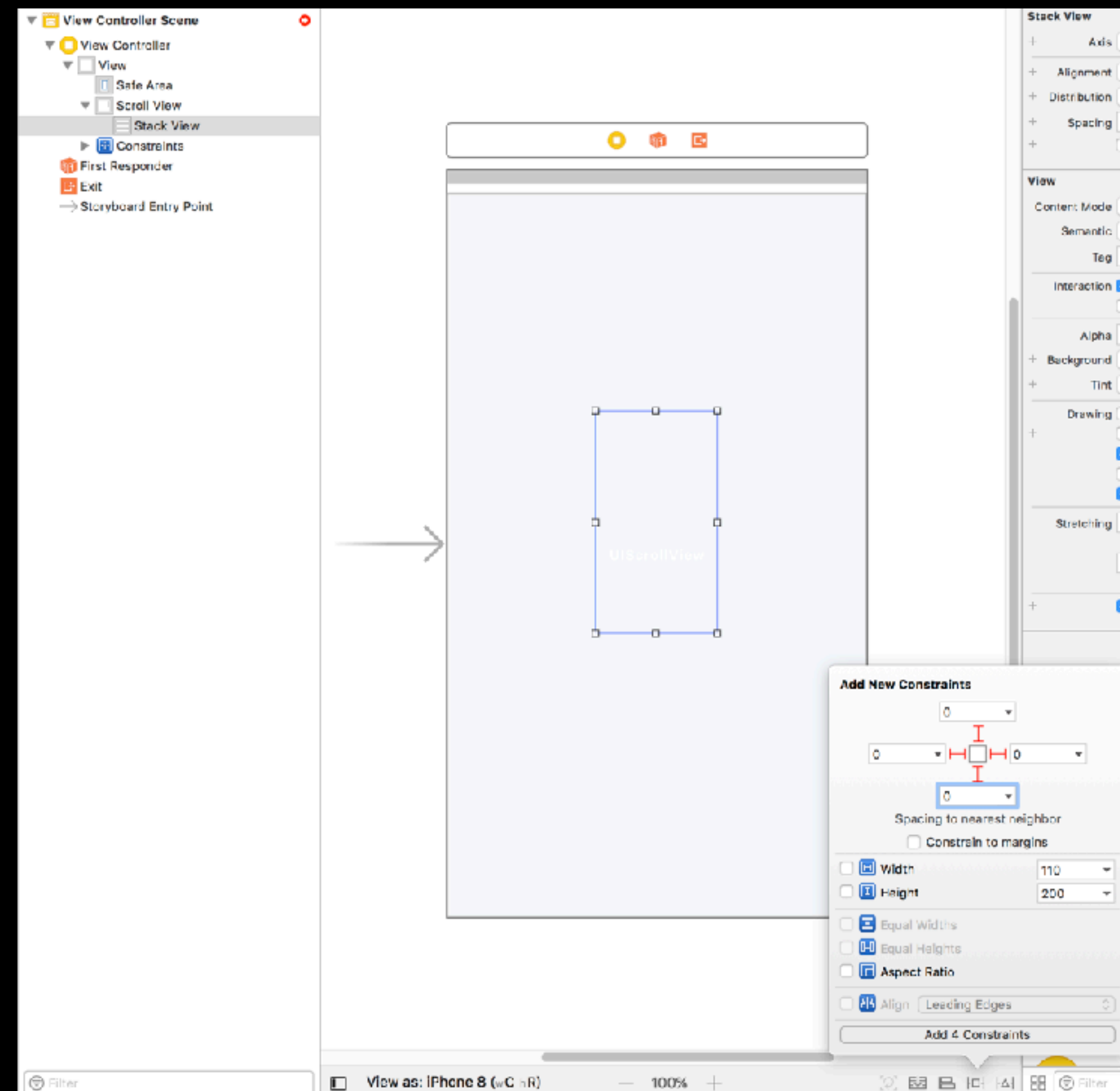
☐ Align Leading Edges

---

Add 4 Constraints

# Scroll views in Interface Builder

## Define content view using stack view



# Scroll views in Interface Builder

## Programmatic constraints

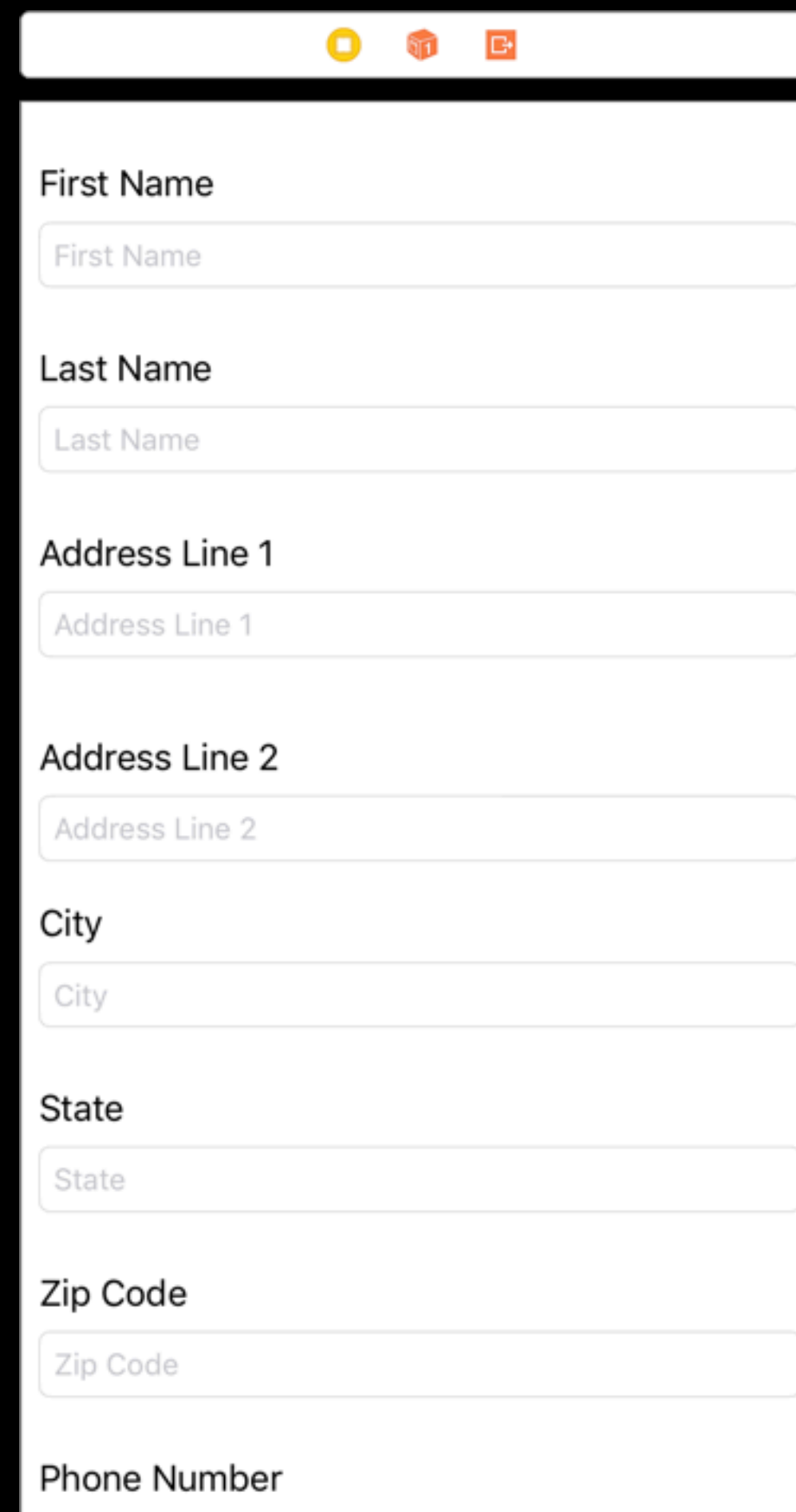
```
imageView.centerXAnchor.constraints(equalTo: scrollView.contentLayoutGuide.centerXAnchor)
```

```
imageView.centerYAnchor.constraints(equalTo: scrollView.contentLayoutGuide.centerYAnchor)
```



# Scroll views in Interface Builder

## Create the form and add fields



The image shows a scroll view containing a form with the following fields:

- First Name
- Last Name
- Address Line 1
- Address Line 2
- City
- State
- Zip Code
- Phone Number

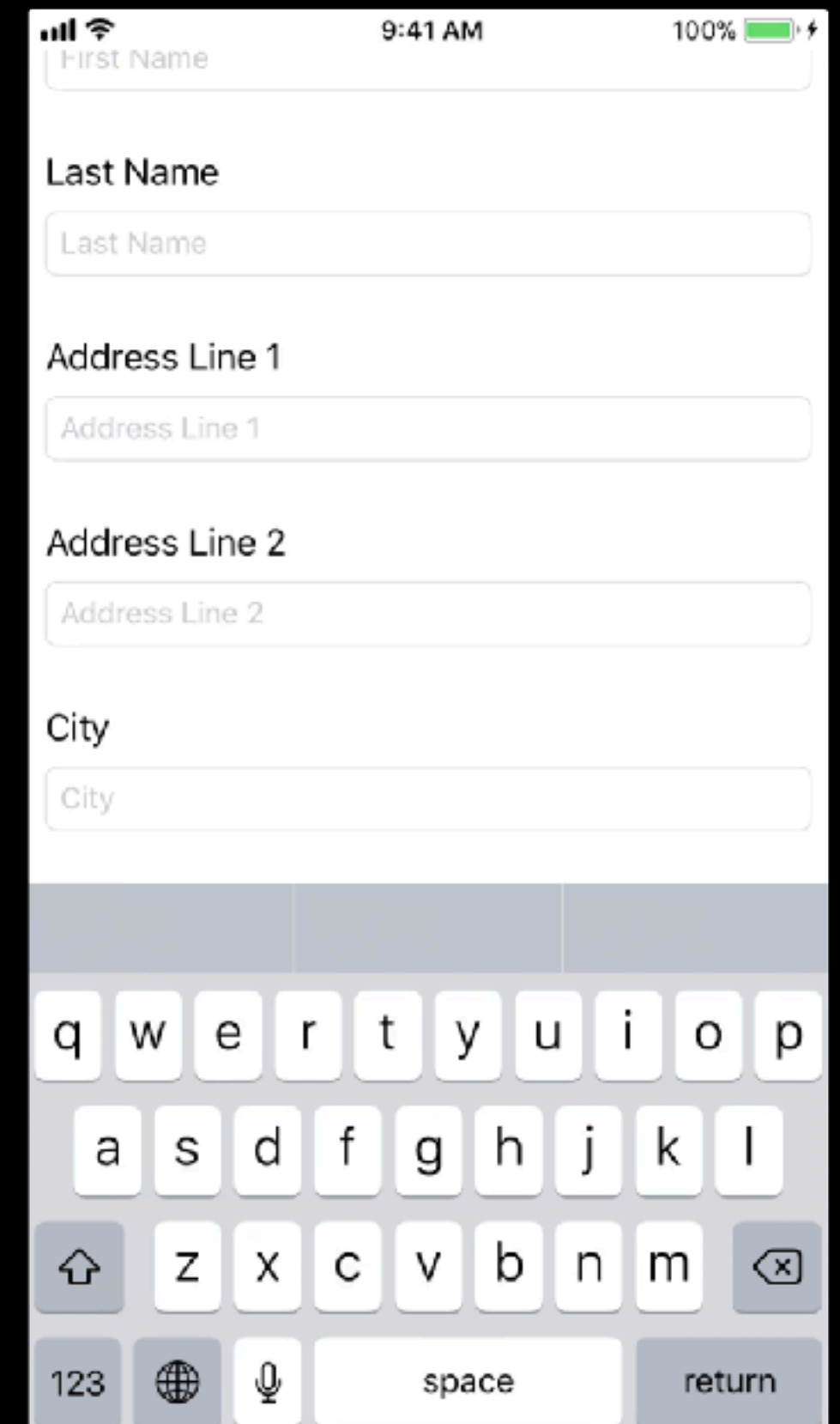
Each field is represented by a text label above a text input field. The input fields are currently empty and have a light gray placeholder text color.

# Keyboard issues

Sent a notification when the keyboard has been shown or will be hidden

Register for keyboard notifications

```
func registerForKeyboardNotifications() {  
    NotificationCenter.default.addObserver(self, selector:  
#selector(keyboardWasShown(_:)), name: .UIKeyboardDidShow, object: nil)  
  
    NotificationCenter.default.addObserver(self, selector:  
#selector(keyboardWillBeHidden(_:)), name: .UIKeyboardWillHide, object: nil)  
}
```



```
func keyboardWasShown(_ notificiation: NSNotification) {  
    guard let info = notificiation.userInfo,  
        let keyboardFrameValue = info[UIKeyboardFrameBeginUserInfoKey] as? NSValue else { return }  
  
    let keyboardFrame = keyboardFrameValue.cgRectValue  
    let keyboardSize = keyboardFrame.size  
  
    let contentInsets = UIEdgeInsetsMake(0.0, 0.0, keyboardSize.height, 0.0)  
    scrollView.contentInset = contentInsets  
    scrollView.scrollIndicatorInsets = contentInsets  
}  
  
func keyboardWillBeHidden(_ notificiation: NSNotification) {  
    let contentInsets = UIEdgeInsets.zero  
    scrollView.contentInset = contentInsets  
    scrollView.scrollIndicatorInsets = contentInsets  
}
```

# Content insets

Allows you to pad the content at the top and bottom of the scroll view

Useful if you have toolbars floating above your scroll view

```
scrollView.contentInset.top  
scrollView.contentInset.bottom  
scrollView.contentInset.left  
scrollView.contentInset.right
```

The diagram illustrates a scroll view with a vertical toolbar at the top and a horizontal toolbar at the bottom. The scroll view's content is a form with several input fields. The top toolbar is shown as a dashed line, indicating it is not part of the scroll view's content. The bottom toolbar is a solid black bar. The form fields are labeled: First Name, Last Name, Address Line 1, Address Line 2, City, State, Zip Code, and Phone Number. Each field has a corresponding input box. The scroll view's content is shown with a vertical dashed line on the left and a horizontal dashed line at the bottom, indicating the scrollable area. The top toolbar is shown as a dashed line, indicating it is not part of the scroll view's content. The bottom toolbar is a solid black bar. The form fields are labeled: First Name, Last Name, Address Line 1, Address Line 2, City, State, Zip Code, and Phone Number. Each field has a corresponding input box. The scroll view's content is shown with a vertical dashed line on the left and a horizontal dashed line at the bottom, indicating the scrollable area.

# Scroll indicator

```
let contentInsets = UIEdgeInsetsMake(0.0, 0.0,  
    keyboardSize.height, 0.0)  
scrollView.contentInset = contentInsets  
scrollView.scrollIndicatorInsets = contentInsets
```

The diagram illustrates a scroll view layout. At the top, a dashed line indicates the top edge of the scroll view's content area. Below this, a solid black bar represents the header. The main content area consists of several form fields, each with a label and a text input field: 'First Name', 'Last Name', 'Address Line 1', 'Address Line 2', 'City', 'State', 'Zip Code', and 'Phone Number'. The 'Phone Number' field contains the text 'Phone number'. A dashed line at the bottom indicates the bottom edge of the scroll view's content area. Vertical dashed lines on the left and right sides of the form fields indicate the scroll view's content insets, which are the areas where the scroll indicator (a vertical bar on the right) is visible. The scroll indicator is shown as a vertical bar on the right side of the scroll view, with a dashed line indicating its position.

# Scroll View family

UITableView

UICollectionView

# Unit 4—Lesson 4

## Scroll Views



Learn how to use a UIScrollView to display content that's larger than the device screen and allow the user to interact with it.

# Unit 4—Lesson 4

## Lab: I Spy



Implement a scroll view on an image view that will enable users to zoom in and pan an image.

Your app will have a single view with a single picture.



