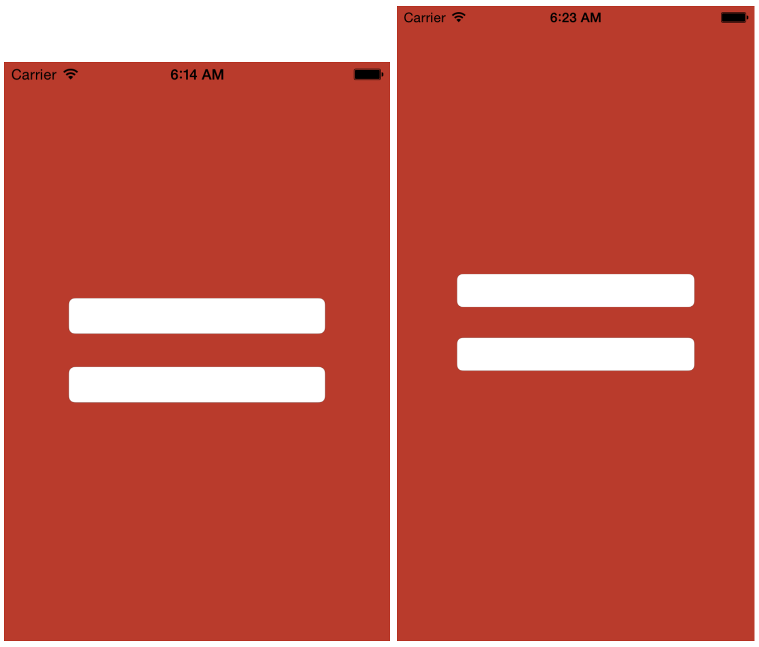
iOS: How To Make AutoLayout Work On A ScrollView

Posted on June 11th, 2014

Ok, I'll admit. I've been seriously struggling with AutoLayout ever since it's been introduced. I understand the concept, and I LOVE the idea of it, but when I actually do it, it almost never behaves as it does in my head.

So when I had a chance to go talk to an actual Apple Engineer about AutoLayout last week at WWDC, I made sure to go. I thought of my most painful experience using AutoLayout recently – when I was making a login screen with username and password fields on a ScrollView (so it scrolls up when the keyboard comes up) – and had the Apple engineer walk me through the example.

Here is what we made:



This is just two input fields centered on a ScrollView. You can see the AutoLayout at work here – the two input fields are centered correctly both on a 4s and a 5s device.

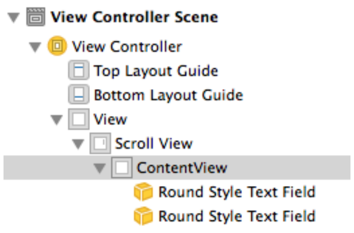
This “simple” solution took the Apple Engineer 40 minutes to solve! However, several senior engineers I know said that they’ve never been able to get AutoLayout working quite right on a ScrollView, so 40 minutes is actually not bad!

Here are the key tricks to getting AutoLayout to work on a ScrollView:

One View

The ScrollView should have only ONE child view. This is forced in Android, so I should have made the connection, but I just didn’t think of it – it’s too easy to put the two input text fields right onto the ScrollView, especially in Interface Builder.

Here is what the View Hierarchy should actually look like:



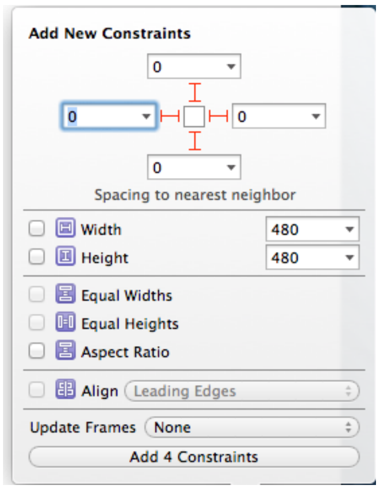
```
▼ View Controller Scene
  ▼ View Controller
      Top Layout Guide
      Bottom Layout Guide
    ▼ View
      ▼ Scroll View
        ▼ ContentView
            Round Style Text Field
            Round Style Text Field
```

Again, make sure to put all your fields and custom views inside the one child view of the ScrollView!

Equal Widths

I’m going to start with the constraints from the outside (on the main view) in (to the stuff inside the ContentView).

The obvious starting point is to bind the ScrollView to the View – just select the ScrollView from the view hierarchy, and add the following constraints:



Add New Constraints

Top: 0; Left: 0; Right: 0; Bottom: 0

Spacing to nearest neighbor

Width: 480

Height: 480

Equal Widths

Equal Heights

Aspect Ratio

Align: Leading Edges

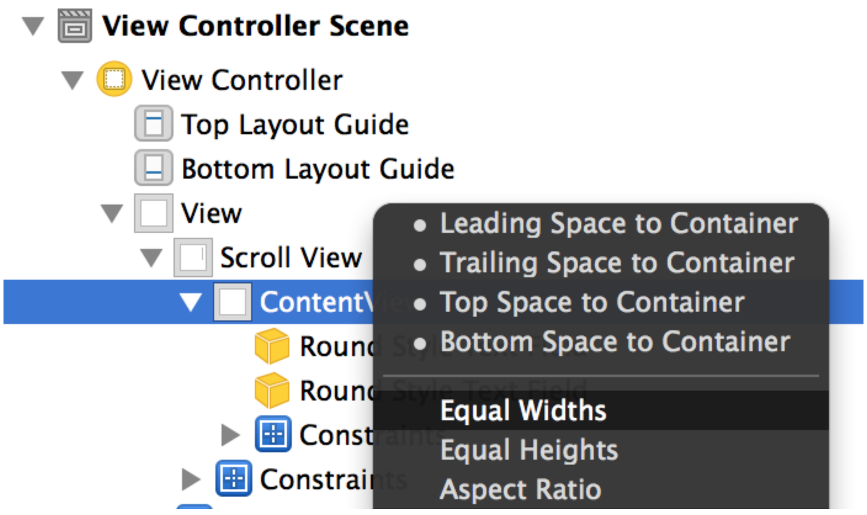
Update Frames: None

Add 4 Constraints

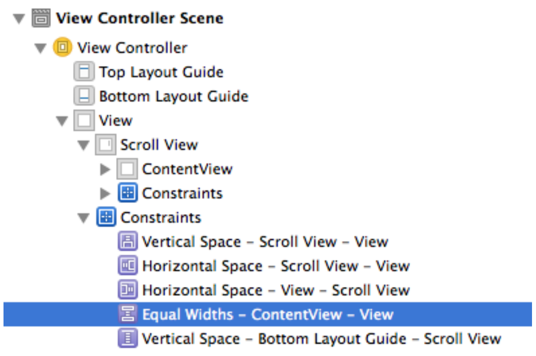
The key to getting the constraints to work properly however, is adding an *Equal Width constraint between the main View and the ContentView*. The ScrollView adjusts to the size of the content inside of it, so setting the

ContentView to the Width of the ScrollView leaves the width of the ContentView ambiguous.

To create the Equal Width Constraint between the ContentView and the View, select the ContentView on the view hierarchy and Control + Drag to the View – you should get a pop-up that gives you the "Equal Widths" option:

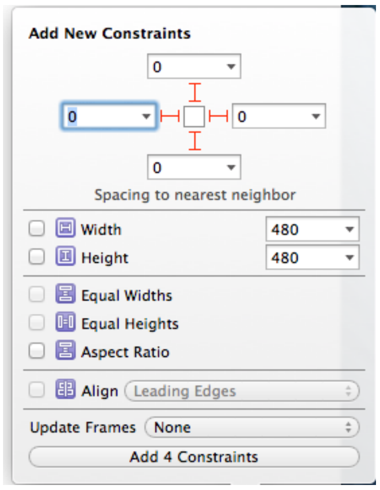


Your constraints between the main View, ScrollView, and ContentView should look like this:


View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Scroll View
ContentView
Constraints
Constraints
Vertical Space – Scroll View – View
Horizontal Space – Scroll View – View
Horizontal Space – View – Scroll View
Equal Widths – ContentView – View
Vertical Space – Bottom Layout Guide – Scroll View

Content Insets

The constraints between the ScrollView and the ContentView are surprisingly straight forward – just bind the ContentView to the ScrollView (make sure the constant to the bottom layout guide is 0):



Add New Constraints

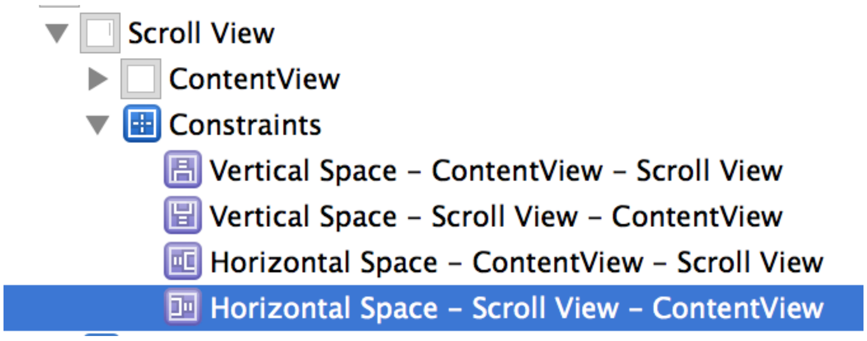
0
0 0
0

Spacing to nearest neighbor

- [ ] Width 480
- [ ] Height 480
- [ ] Equal Widths
- [ ] Equal Heights
- [ ] Aspect Ratio
- [ ] Align Leading Edges

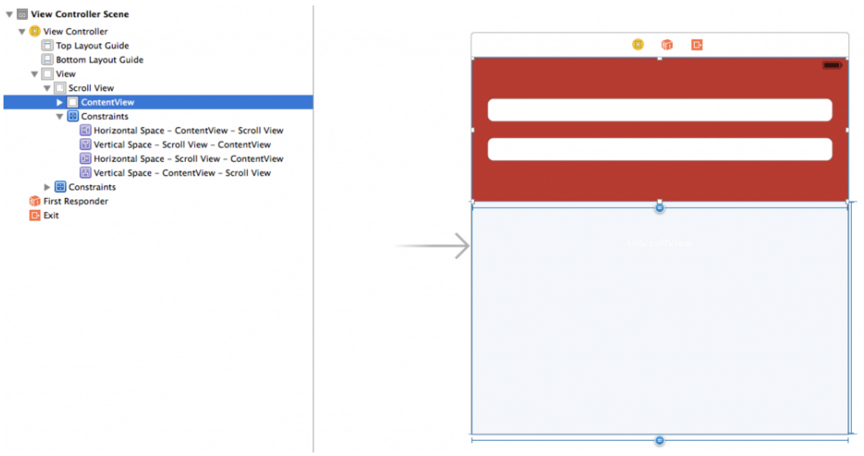
Update Frames None

The constraints between the ContentView and ScrollView are now as follows with all constants set at 0:



Scroll View
ContentView
Constraints
Vertical Space – ContentView – Scroll View
Vertical Space – Scroll View – ContentView
Horizontal Space – ContentView – Scroll View
Horizontal Space – Scroll View – ContentView

If your storyboard is like mine, you might notice that the actual ContentView is not the full height of the main view or the ScrollView:

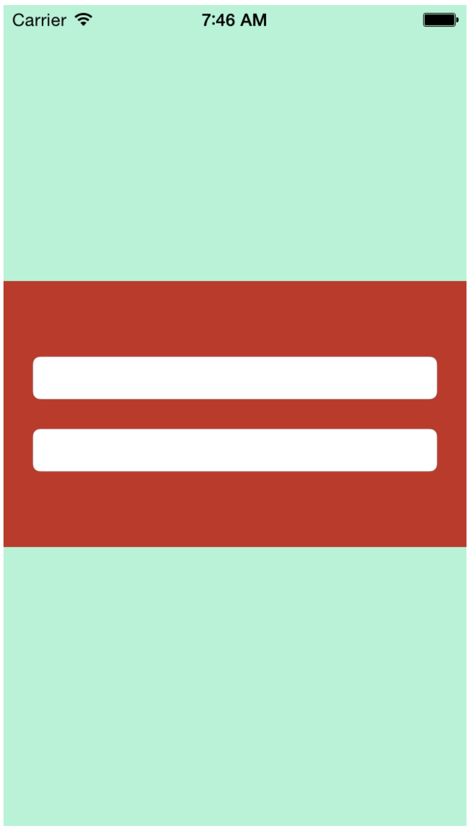


View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Scroll View
ContentView
Constraints
Horizontal Space – ContentView – Scroll View
Vertical Space – Scroll View – ContentView
Horizontal Space – Scroll View – ContentView
Vertical Space – ContentView – Scroll View
Constraints
First Responder
Exit

However, we do want to make sure the ContentView is centered when it's rendered on a device. To do that we need to write some code to property set the Content Insets in the ViewController:

```swift
// ViewController.swift

import UIKit

class ViewController: UIViewController {

    @IBOutlet var scrollView : UIScrollView
    @IBOutlet var contentView : UIView


    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically fr
    }

    override func viewDidLayoutSubviews()
    {
        let scrollViewBounds = scrollView.bounds
        let containerViewBounds = contentView.bounds

        var scrollViewInsets = UIEdgeInsetsZero
        scrollViewInsets.top = scrollViewBounds.size.height/2.0;
        scrollViewInsets.top -= contentView.bounds.size.height/2.0;

        scrollViewInsets.bottom = scrollViewBounds.size.height/2.0
        scrollViewInsets.bottom -= contentView.bounds.size.height/2.0;
        scrollViewInsets.bottom += 1

        scrollView.contentInset = scrollViewInsets
    }


}
```

Once you add the proper constraints into the ContentView (see next step), your final result will look like this:



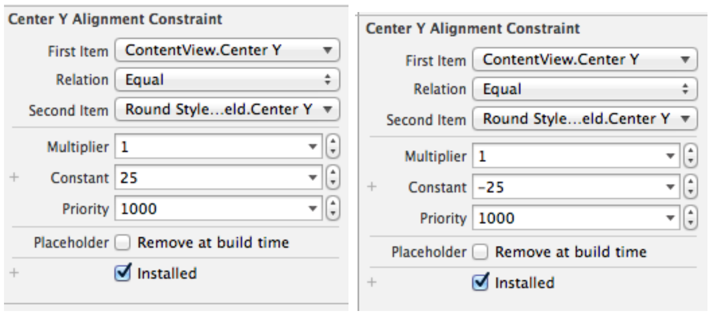
The ugly colors are meant to differentiate the ScrollView (green) from the ContentView (red). Again, in the storyboard, the ContentView is at the top of the ScrollView, but with our content insets set in code, it now becomes centered.

Centering Multiple Views

The final step is to add AutoLayout to the ContentView. This is the same as adding layout normally to any view, so I won't go into much detail here.

The one thing I did learn that I'd like to share (although now it seems obvious) is how to center the two text fields in the view. Previously, I put the two text fields into a container view, and centered the container view in the parent view. However, that is not necessary.

Instead, you can center each text field horizontally in container (so they're now centered and on top of each other), and then add a constant of 25 to one (so it's moved up 25 pixels from the center), and add a constant of -25 to the other (so it's moved down 25 pixels from the center).



Center Y Alignment Constraint

First Item	ContentView.Center Y
Relation	Equal
Second Item	Round Style...eld.Center Y
Multiplier	1
Constant	25
Priority	1000
Placeholder	☐ Remove at build time
	✓ Installed

Center Y Alignment Constraint

First Item	ContentView.Center Y
Relation	Equal
Second Item	Round Style...eld.Center Y
Multiplier	1
Constant	-25
Priority	1000
Placeholder	☐ Remove at build time
	✓ Installed

This will leave you with a space of 50 pixels between the two text fields, but the space exactly in between them will be the center of the view.

Do you have any other AutoLayout tips? I'm always looking to learn more and improve, so please let me know in the comments!

You can view the *source code on Github here*.

Enjoy the article? Join over 9,500+ Swift developers and enthusiasts who get my weekly updates.

Email

Subscribe

50 Comments Natasha The Robot 1 Login

Recommend 9 Share Sort by Best

Join the discussion…

Paul Silvis · 2 years ago
Very helpful. For anyone using nibs, you won't be able to set the "Equal Width" constraint on the main view. Instead, create another top-level UIView that contains your scrollview.

Set its Top,Bottom,Leading,Trailing to superview and link your existing contentView to the new view's width
10 · Reply · Share ›

Mat Cegiela → Paul Silvis · a year ago
In a xib situation, as a workaround, you can also create a second top-level view and drop your view into it. This allows you to link constraints to it form deeper level views. After you create all the constraints, simply lift it out again and delete the temporary containing top-level view.
2 · Reply · Share ›

Tim Smith → Paul Silvis · a year ago
This helped me hugely, but for clarity the ScrollView should not be contained in the new view .. otherwise you still don't get the equal widths option.
· Reply · Share ›

Benjamin Horner → Paul Silvis · a year ago
Wow ! Thanks !!!! you saved me a headache !!! I was going crazy with the iPhone 6 screen size !
· Reply · Share ›

RK · a year ago
Thanks for this usefull explanation but ... on my iPhone 6+ with iOS8.1, when the keyboard is open, it is not possible to scroll in the view (impossible to see the textfield). What's wrong ? I have used our source code from Github. Any help will be great.
4 · Reply · Share ›

Tommy Juszczyk → RK · 3 months ago
Also seeing this in 9.1
· Reply · Share ›

James Clutterbuck · 9 months ago
The "scrollview has ambiguous scrollable content height" fix for me was to add a constraint on my last element for bottom spacing to ContentView and all the other elements in the ContentView have constraints on their Top/Bottom to the element Above/Below them, giving a finite height and no more warnings :-)
3 · Reply · Share ›

Scotty Fister → James Clutterbuck · 5 months ago
This did it for me too, thanks! Also ensuring that my scrollview/content view had constraints relative to the superview - it kept trying to use "bottom layout guide" and that screwed it up, too. Clicking the carot when setting pin values you can check view instead of layout guide.
· Reply · Share ›

Levi Bostian → James Clutterbuck · 6 months ago
I think this is a big reason why scroll view does not work for lots of people. Setting the spacing between elements gives the scrollview a fixed height.

This is how I got success for my scrollview. Thanks for mentioning it here!
· Reply · Share ›

ximmyxiao · a year ago
I just do as the list told ,After set Equal Width ,but the IB still complains about has

← **WWDC SURVIVAL TIPS FOR NOOBS** **CALABASH IOS TUTORIAL** →