Custom Collection View Layouts

Hands-on Challenges

Custom Collection View Layouts Hands-on Challenges

Challenge D: Even More Depth!

The sense of depth you've added to the header as the user scrolls is a cool visual effect, but you can take it one step further and really accentuate the overall sense of depth by having the foreground image, which in this case is the RWDevCon logo, scale by the same amount but in the opposite direction.

SCALE.ALL.THE.THINGS

Your challenge this time is to take everything you've learnt whilst working through the video and apply it to the foreground image view. Remember that the trick is to correctly set up the Auto Layout constraints, and then take advantage of the custom layout attributes you're already passing to the header.

Before you turn the page for our solution, be sure to give it a try for yourself first!

Solution

Updating the Header Subclass

Open **ScheduleHeaderView.swift** and add the following outlet declaration just below the other outlets:

```
@IBOutlet private weak var
  foregroundImageViewHeightLayoutConstraint: NSLayoutConstraint!
```

In a moment you'll connect this to the height layout constraint of the foreground image view so that you can manipulate it's constant in the same way you do for the background image view.

Add a new private property:

```
private var foregroundImageViewHeight: CGFloat = 0
```

This will be used to store the initial height of the foreground image view when the header is first displayed. You give it a default value of `0` to avoid having to add an initializer to the class.

Next, update `awakeFromNib()` to capture the initial height of the foreground image view. Add the following to the bottom of the method:

```
foregroundImageViewHeight =
CGRectGetHeight(foregroundImageView.bounds)
```

You capture the initial height so that you can add the `deltaY` to it when the layout attributes are applied. Speaking of which, add the following statement to the bottom of `applyLayoutAttributes(_:)`:

```
foregroundImageViewHeightLayoutConstraint.constant =
  foregroundImageViewHeight + attributes.deltaY
```

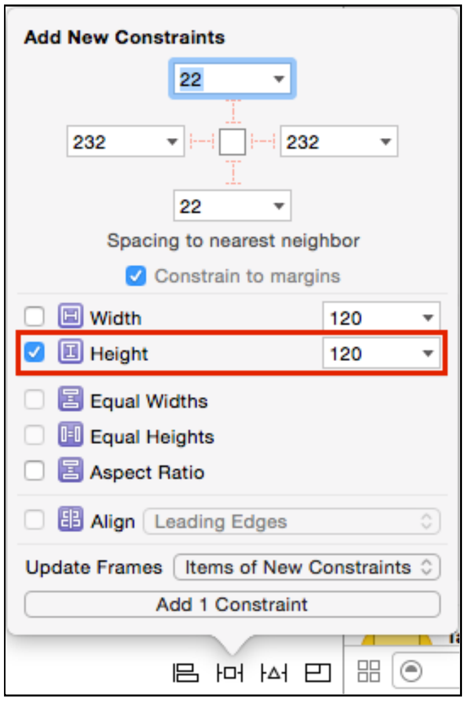
Here you're updating the constant of the foreground image view's height layout constraint just as you do for the background image view, except this time you're adding `deltaY` to it, rather than subtracting it. This will result in the image view growing instead of shrinking when the user scrolls.

With the subclass updated, it's time to move onto the storyboard.

Updating the Storyboard

Open **Main.storyboard**, select the foreground image view – the one containing the RWDevCon logo – and choose **Editor\Resolve AutoLayout Issues\Selected Views\Clear Constraints**. This is a simple and quick way to remove all the existing layout constraints from a selected view, and saves you having to delete each one individually by hand. You're removing the constraints as you need to set up some new ones to properly position and scale things.

With the foreground image view still selected, click the **Pin** button and add a **Height** constraint with a value of **120** points:


Add New Constraints

22

232 232

22

Spacing to nearest neighbor

☑ Constrain to margins

☐ Width 120

☑ Height 120

☐ Equal Widths

☐ Equal Heights

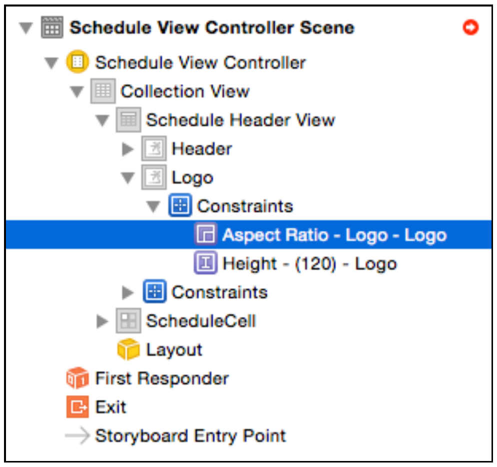
☐ Aspect Ratio

☐ Align Leading Edges

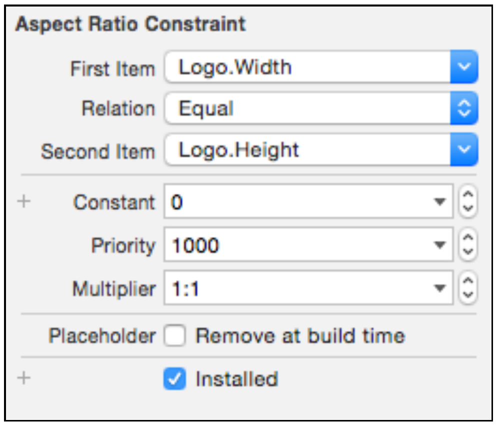
Update Frames Items of New Constraints

Make sure Update Frames is set to **Items of New Constraints** and click **Add 1 Constraint**.

Next, **ctrl-drag** from the image view to the **Logo** object in the Document Outline and choose **Aspect Ratio** from the popup menu. Expand the constraints underneath the **Logo** object in the Document Outline and select the new **Aspect Ratio** constraint:



Then, in the Attributes inspector set the Multiplier to **1:1**:


Aspect Ratio Constraint
First Item: Logo.Width
Relation: Equal
Second Item: Logo.Height
Constant: 0
Priority: 1000
Multiplier: 1:1
Placeholder: [ ] Remove at build time
[✓] Installed

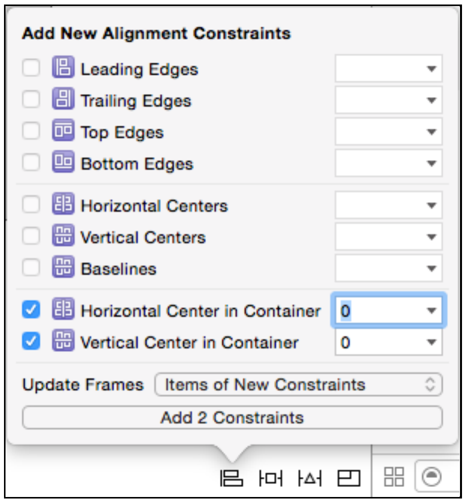
This will result in the width being pinned to the height; when ever you change the height from now on the width will automatically update to the same value.

Next, click the **Align** button and check the following:

• **Horizontal Center in Container**

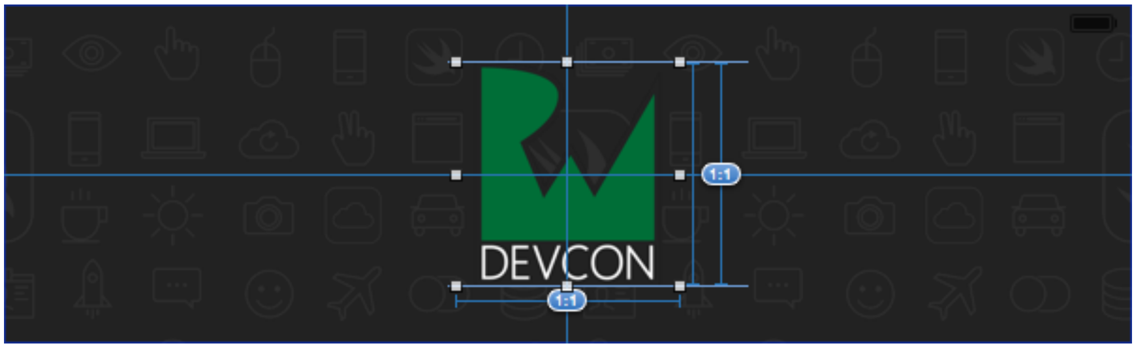
• **Vertical Center in Constainer**

As shown here:


Add New Alignment Constraints
[ ] Leading Edges
[ ] Trailing Edges
[ ] Top Edges
[ ] Bottom Edges
[ ] Horizontal Centers
[ ] Vertical Centers
[ ] Baselines
[✓] Horizontal Center in Container: 0
[✓] Vertical Center in Container: 0
Update Frames: Items of New Constraints
Add 2 Constraints

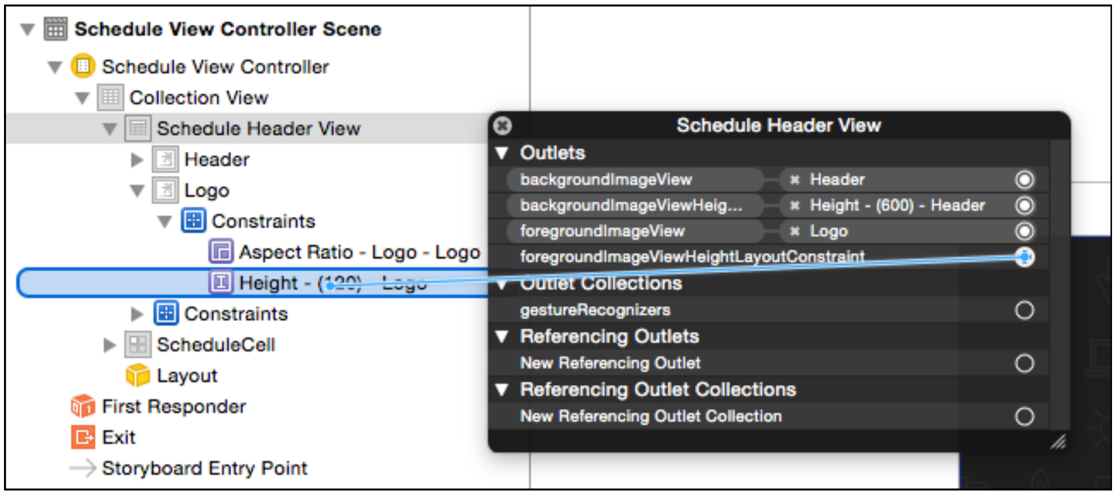
Make sure Update Frames is set to **Items of New Constraints** and click **Add 2 Constraints**.

With the necessary constraints now set up, your header should look like this:



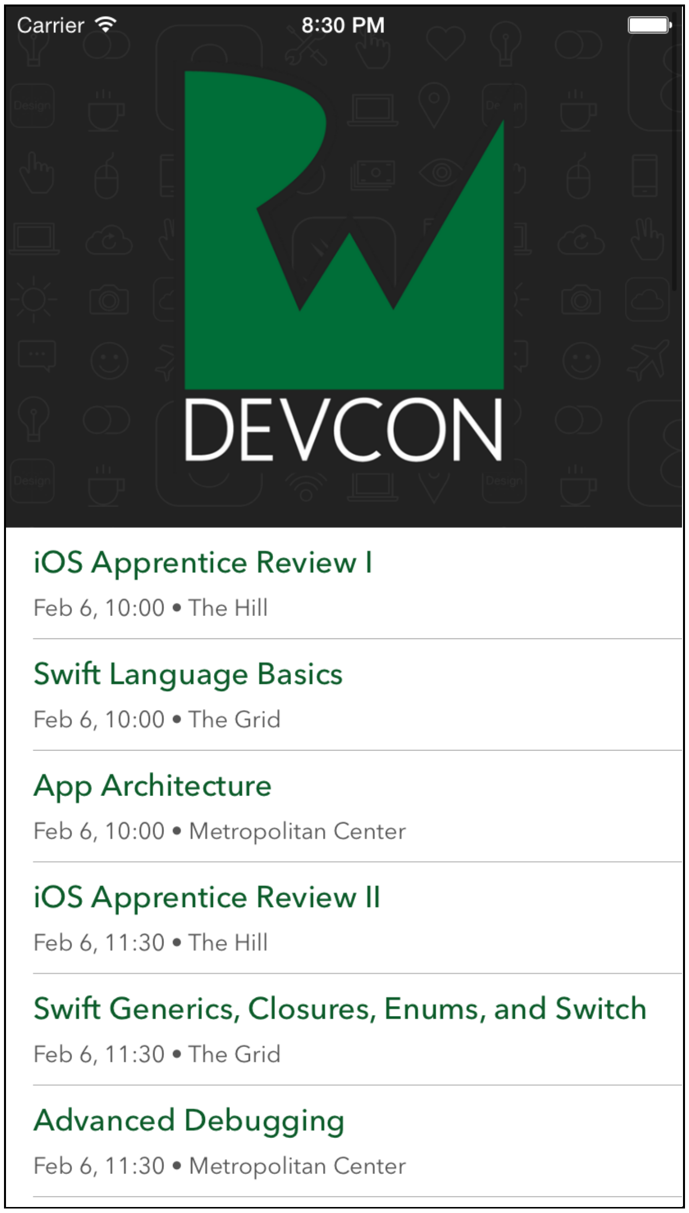
Select the foreground image view once again, and in the Attributes inspector set View\Mode to **Scale to Fill**. You do this because you want the image to scale as the containing image view scales – the logo image you're using is much larger than 120x120 points so it won't distort or pixelate as it scales.

Finally, right-click on **Schedule Header View** in the Document Outline to invoke the Outlets and Actions menu, and then connect `foregroundImageView` to the **Logo** image view containing the RWDevCon logo, and connect `foregroundImageViewHeightLayoutConstraint` to the **Height – (120) – Logo** constraint, as shown below:



Those are the final changes you need to make to the storyboard.

Build and run. You'll notice that as you scroll the foreground image view grows as the background image shrinks, which really accentuates the overall sense of depth:



iOS Apprentice Review I
Feb 6, 10:00 • The Hill

Swift Language Basics
Feb 6, 10:00 • The Grid

App Architecture
Feb 6, 10:00 • Metropolitan Center

iOS Apprentice Review II
Feb 6, 11:30 • The Hill

Swift Generics, Closures, Enums, and Switch
Feb 6, 11:30 • The Grid

Advanced Debugging
Feb 6, 11:30 • Metropolitan Center

Congratulations! You’ve applied everything you learnt in the video to the foreground image view, which now scales in tandem with the background image view to create a really nice visual effect.