Scroll View School

Hands-On Challenges

Scroll View School
Hands-On Challenges

Challenge B: Frame It!

So far, you have only added one subview to your scroll view, but you can add more than one subview if you'd like. The scroll view will display all of its subviews, scrolling accoridng to how you set its content size (either directly or via constraints).

However, when you implement `viewForZoomingInScrollView:`, you can only return one subview for zooming. So the workaround is to make a new "content view" that contains all the subviews that you want to zoom, like this:

Scroll View

|--Content View

    |--Subview 1

    |--Subview 2

In this tutorial, you will get practice with this concept by framing the zombie image: first with a plain gray frame, and then with a resizable frame image.

Part 1: Gray Frame

Open the starter project for this challenge (which is the same spot where the demo from the lecture left off).

Open **RWTCenteredImageScrollView.m**, and add a new property for the content view:

```
@property (nonatomic, strong) UIView *contentView;
```

At the beginning of `initWithFrame:`, add these lines to initialize the view:

```
self.contentView = [[UIView alloc] initWithFrame:CGRectZero];
self.contentView.translatesAutoresizingMaskIntoConstraints = NO;
self.contentView.backgroundColor = [UIColor grayColor];
[self addSubview:self.contentView];
```

You want to make the image view a subview of the content view now, rather than a subview of the scroll view. So modify the line that adds the image view as a subview to the following:

```
[self.contentView addSubview:self.imageView];
```

Next, you need to update the constraints to pin the content view to the edges of the scroll view (rather than the image view), since the content view is now the subiew of the scroll view. To do this, replace `self.imageView` with `self.contentView` in for `constraintRight`:

```
self.constraintRight = [NSLayoutConstraint
  constraintWithItem:self.contentView
  attribute:NSLayoutAttributeRight relatedBy:NSLayoutRelationEqual
  toItem:self attribute:NSLayoutAttributeRight multiplier:1
  constant:0];
[self addConstraint:self.constraintRight];
```

Then repeat this for `constraintTop`, `constraintBottom`, and `constraintLeft`.

Next, you need to add the constraints to pin the image view inside the content view. To do this, add these lines right before the part that adds the notification observer:

```
NSDictionary *viewsDictionary = @{@"imageView": self.imageView};

NSArray *horzConstraints = [NSLayoutConstraint
  constraintsWithVisualFormat:@"|-(75)-[imageView]-(75)-|"
  options:0 metrics:nil views:viewsDictionary];
[self.contentView addConstraints:horzConstraints];

NSArray *vertConstraints = [NSLayoutConstraint
  constraintsWithVisualFormat:@"V:|-(75)-[imageView]-(75)-|"
  options:0 metrics:nil views:viewsDictionary];
[self.contentView addConstraints:vertConstraints];
```

These pin the image view to the edges of the content view, but leaves a margin so the background color of the border can show through, effectively making a border.

Finally, replace `viewForZoomingInScrollView:` with the following:

```
- (UIView *)viewForZoomingInScrollView:(UIScrollView *)scrollView
{
  return self.contentView;
}
```

And that's it! Build and run, and your scroll view now has a nice border:

Uber Haxx0r Challenge: Wood Frame

In the resources for this challenge, you will find an image asset for a picture frame. Your challenge is to add the picture frame to the content view, so that the picture now looks like this: