306: Animating Auto Layout Constraints

Part 2: Demo Instructions

306: Animating Auto Layout Constraints
Part 2: Demo Instructions

Basic Animations

In this demo you will add some basic auto layout constraints animations to the starter project.

The steps here will be explained in the demo, but here are the raw steps in case you miss a step or get stuck.

Step 1: Find & Replace Constraints

In **ViewController.swift**, add in `adjustHeights(...)`:

```
for constraint in viewToSelect.superview!.constraints() as
[NSLayoutConstraint] {
  if contains(views, constraint.firstItem as UIView) &&
    constraint.firstAttribute == .Height {

      println("height constraint found")
  }
}
```

Add after the `println()`:

```
viewToSelect.superview!.removeConstraint(constraint)
```

Add the code to calculate the multiplier:

```
var multiplier: CGFloat = 0.34
if shouldSelect {
  multiplier = (viewToSelect == constraint.firstItem as UIView) ?
0.55 : 0.23
}
```

Finally add the new constraint to replace the old one:

```
let newConstraint = NSLayoutConstraint(
  item: constraint.firstItem,
  attribute: .Height,
  relatedBy: .Equal,
  toItem: (constraint.firstItem as UIView).superview!,
  attribute: .Height,
  multiplier: multiplier,
```

```
  constant: 0.0)
viewToSelect.superview!.addConstraint(newConstraint)
```

Step 2: Add animations

At the bottom of toggleView(...) add:

```
UIView.animateWithDuration(1.0, delay: 0.00,
usingSpringWithDamping: 0.4, initialSpringVelocity: 1.0, options:
.CurveEaseIn | .AllowUserInteraction | .BeginFromCurrentState,
animations: {
    self.view.layoutIfNeeded()
}, completion: nil)
```

Step 3: Reading Animations

In `viewDidLoad()` replace:

```
Selector("toggleView:")
```

with:

```
Selector("toggleSpeaking:")
```

Add the new tap handler:

```
func toggleSpeaking(tap: UITapGestureRecognizer) {
    toggleView(tap)
    let isSelected = (selectedView==tap.view!)

}
```

Append to the new method:

```
UIView.animateWithDuration(1.0, delay: 0.00,
usingSpringWithDamping: 0.4, initialSpringVelocity: 10.0, options:
.CurveEaseIn, animations: {
  self.changeDetailsTo(isSelected ? kSelectedDetailsText :
kDeselectedDetailsText)
  self.view.layoutIfNeeded()
}, completion: nil)
```

This will spawn an error because `changeDetailsTo(...)` is still not implemented – add it to the class:

```
func changeDetailsTo(text: String) {
    speakingDetails.text = text

}
```

Add inside the method the code to move the text out of the screen:

```
for constraint in speakingDetails.superview!.constraints() as
[NSLayoutConstraint] {
  if constraint.firstItem as UIView == speakingDetails &&
  constraint.firstAttribute == .Leading {

    constraint.constant = -view.frame.size.width/2
    speakingView.layoutIfNeeded()

    break
  }
}
```

And insert the code to animate the text back to its original location just before `break`:

```
constraint.constant = 0.0
UIView.animateWithDuration(0.5, delay: 0.1, options:
.CurveEaseOut, animations: {
  self.speakingView.layoutIfNeeded()
}, completion: nil)
```

Step 4: Deselect views

When you click not on the same view but on another – the old text stays in the Speaking strip. Append a "deselect" handler to the `toggleSpeaking(...)` method:

```
deselectCurrentView = {
  self.changeDetailsTo(kDeselectedDetailsText)
}
```

Now call the "deselect" handler before you select any new views. In `toggleView(...)` insert the following code before assigning value to `selectedView`:

```
if !wasSelected {
    UIView.animateWithDuration(1.0, delay: 0.00,
usingSpringWithDamping: 0.4, initialSpringVelocity: 10.0, options:
.CurveEaseIn | .AllowUserInteraction | .BeginFromCurrentState,
animations: {
```

```
      self.deselectCurrentView?()
      self.deselectCurrentView = nil
      self.view.layoutIfNeeded()
    }, completion: nil)
  }
```

Conclusion

Congrats, at this time you should have the basic animations complete, and learned a lot about Auto Layout along the way! You are ready to move on to the lab.