Beginning iOS Animation

Hands-on challenges

Beginning iOS Animation Hands-On Challenges

Challenge A: Constraint outlets

I hope you enjoyed that first video in the series! You didn't cover that much but you did learn an important technique you are going to use on daily basis. Creating an outlet to a constraint is the easiest way to create animations with Auto Layout.

In this video's challenge you are going to create one more outlet in order to exercise the process of adding one and connecting it in Interface Builder. In particular you are going to animate the trailing space from the close button to the right edge of the screen - this will make the change of the button's role a bit more visual. Each time the user opens the menu the close button will move slightly to the left and when they close the menu will go back to its initial place.

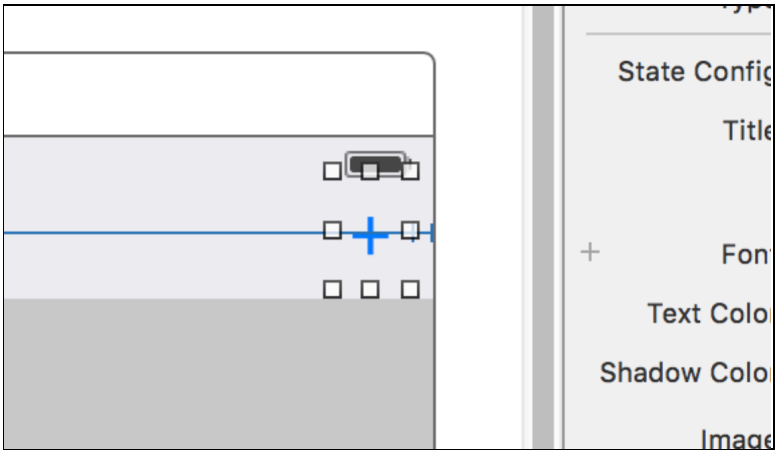
Let's get started.

Open **ViewController.swift** and add a new outlet property to the `ViewController` class:

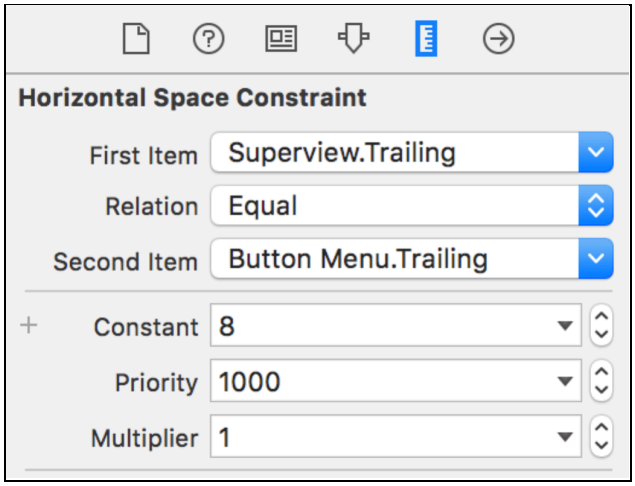
```
@IBOutlet weak var closeButtonTrailing: NSLayoutConstraint!
```

Just as before the type of the property is `NSLayoutConstraint` - no matter if its a trailing space, alignment or size constraint the type is always the same.

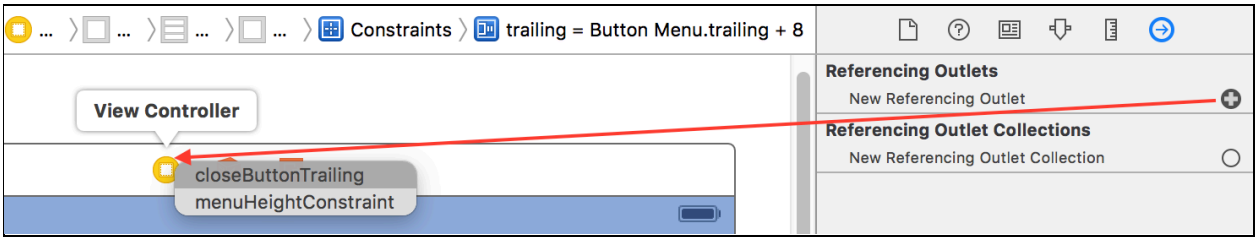
Now open **Main.storyboard** and select the add/close button:



Switch to the size inspector and double click the trailing space constraint - that will show you the constraint properties.



Click on the **Connections Inspector** and drag from the connection circle to your view controller:



From the popup menu select **closeButtonTrailing** and you are ready.

Now that the constraint outlet is connected you can simply change its constant value from code as you did for `menuHeightConstraint` earlier.

Find the line where you alternate between higher and shorter menu:

```
menuHeightConstraint.constant = isMenuOpen ? 200.0 : 60.0
```

Add one more line to toggle between 8 and 16 points of trailing space for the close button:

```
closeButtonTrailing.constant = isMenuOpen ? 16.0 : 8.0
```

Now when you open the menu you will have the close button jump a bit away from the screen edge - this way the user is more likely to notice the change in the button's function.

Challenge B: Combine constraint and non layout animations

You might have seen already animation code that did not have to deal with constraints and you know that it does look a bit different.

With non-constraint based animations you directly change a view's properties from within an `animations` block and UIKit figures out how to create the animation automatically.

You are going to look more into this kind of animations later on in the video series but just to give you a little taste let's add two more animations to your menu bar animation.

Scroll to `actionToggleMenu(_)` and inside the `animations` block add:

```
let angle = self.isMenuOpen ? CGFloat(M_PI_4) : 0
self.buttonMenu.transform = CGAffineTransformMakeRotation(angle)
```

This code creates a local constant angle and sets it to `45` degrees or `0` degress depending on whether the menu is currently open.

Now the **+** button becomes a **x** button when the menu is open, which makes it more clear to the user what the current function of the button is.

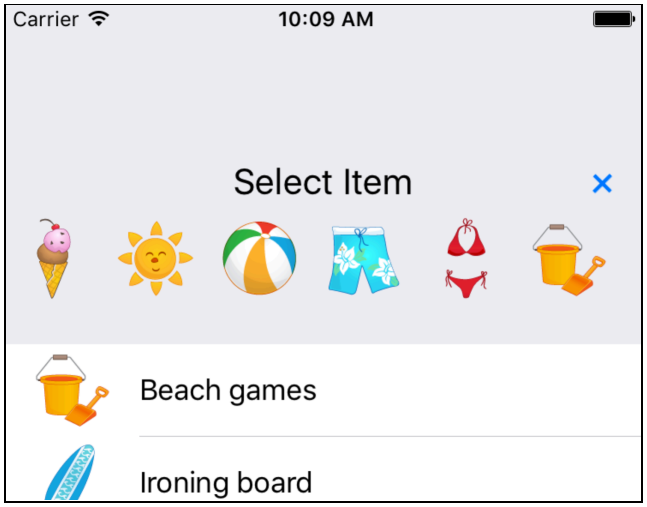
Run the project and enjoy the little animation :] You learn more about animating the transform property later on in the video series.

To wrap-up with this challenge add one more line:

```
self.slider.alpha = self.isMenuOpen ? 1 : 0
```

This code fades in the packing items slider control when the menu opens and fades it out when you close the menu.

Run the app one final time and you will see the available packing items appear as you open the menu:



Next you will look into how to create animations with Auto Layout that require you to change the multiplayer property of a constraint.