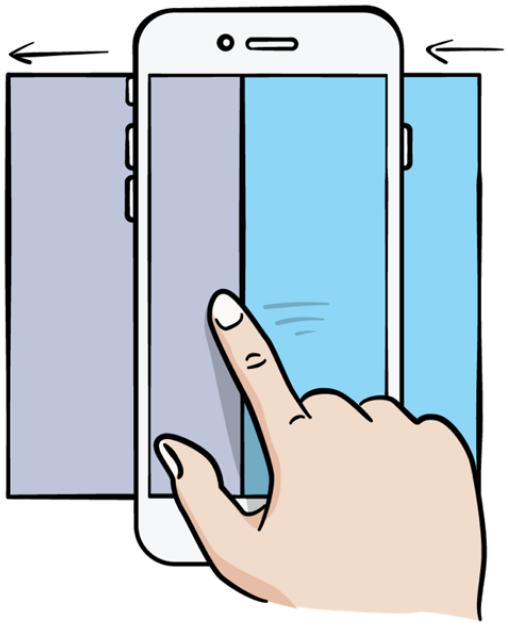
SCROLL VIEW SCHOOL



Scroll View School

Brian Moakley

Notice of Rights

Notice of Liability

Trademarks

Table of Contents: Overview

Table of Contents: Extended

Scroll View School for Video 14: Sidebar 1

By Brian Moakley

If you followed along with the demo – congratulations! You have a working sidebar with most of the functionality there.

However, there are a few small bits of polish you can look at before continuing on to the rest of the sidebar implementation in the next video.

Challenge

First is adding a little bit of shadow to the main view. This will make it look like it's floating above the sidebar and stand out more.

Open **SidebarViewController.swift** and add the following helper method to the class:

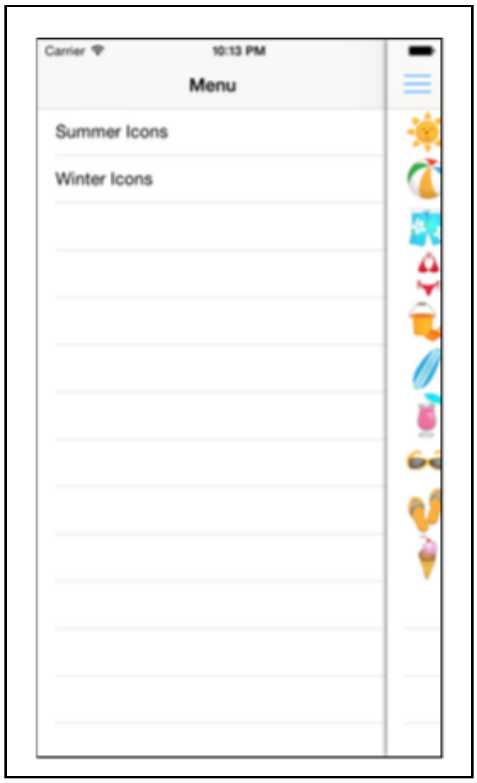
```
fileprivate func addShadowToView(_ destView: UIView) {
  destView.layer.shadowPath = UIBezierPath(rect: destView.bounds).cgPath
  destView.layer.shadowRadius = 2.5
  destView.layer.shadowOffset = CGSize(width: 0, height: 0)
  destView.layer.shadowOpacity = 1.0
  destView.layer.shadowColor = UIColor.black.cgColor
}
```

This will add the shadow parameters to a view's layer to give it a nice effect. All you need to do is call it on the main view controller.

Find `setupViewControllers()` and add the following code just below the calls to `addViewController()`:

```
addShadowToView(mainViewController.view)
```

This will apply the shadow effect to the main view controller's view. Build and run and have a look for yourself!



Challenge 2

When you launch the app, the scroll view's content offset is at the origin so the sidebar menu is visible. In many cases, it's better to have the main view visible on launch instead.

Later on, you'll need some way to open and close the sidebar menu from code so we'll also get a head start on that problem!

Still in **SidebarViewController.swift**, add the following method to the class:

```
func closeMenuAnimated(_ animated: Bool) {
  scrollView.setContentOffset(
    CGPoint(x: leftViewController.view.frame.width, y: 0),
    animated: animated)
}
```

This will close the sidebar menu and return to the main view. If you think about the placement within the scroll view, the left sidebar is at position x=0 and has a certain width. So if you set the content offset x to that sidebar width, then you'll have scrolled over to the main content – and that's exactly what this method does.

You might add a call to `closeMenuAnimated` somewhere like `viewWillAppear`, but that will close the menu every time the view appears which could happen many times. You really only want to close the menu on launch, the very first time.

Add the following property to the class:

```
var firstTime = true
```

This Boolean property starts out as true, and you'll set it to false after calling `closeMenuAnimated`.

Since you're using auto layout to position the sidebar and main view within the scroll view, you can't call `closeMenuAnimated` until after auto layout has had a chance to use all the constraints.

Rather than `viewWillAppear`, the correct place to call `closeMenuAnimated` is `viewDidLayoutSubviews`. Add the following method override to the class:

```
override func viewDidLayoutSubviews() {
  if firstTime {
    firstTime = false
    closeMenuAnimated(false)
  }
}
```

This will call `closeMenuAnimated()` only once, but after the scroll view's content has been set up.

Build and run, and you should see the main view is there when you launch the app. To access the sidebar, you just swipe as usual.