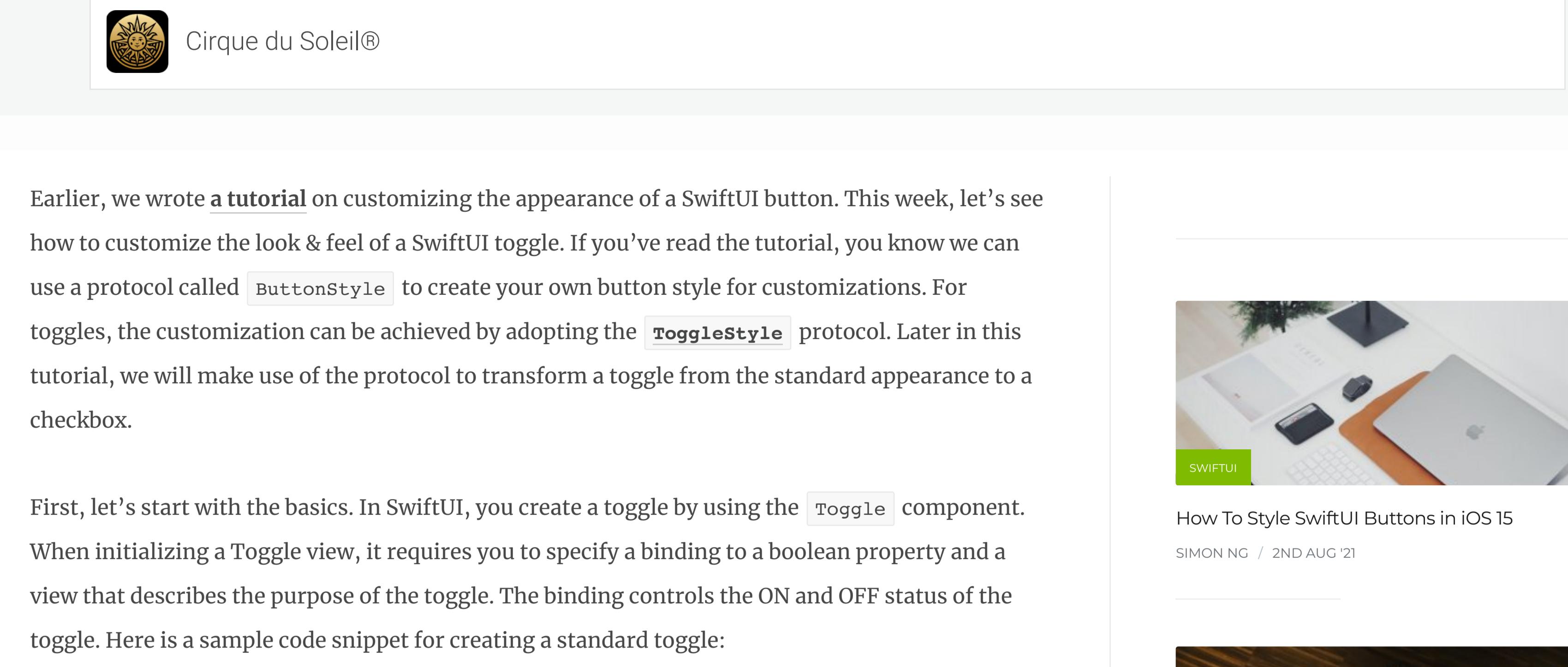


# How to Customize the Appearance of SwiftUI

## Toggle

SIMON NG | 4TH SEP '20 | 0



[Book Now](#)

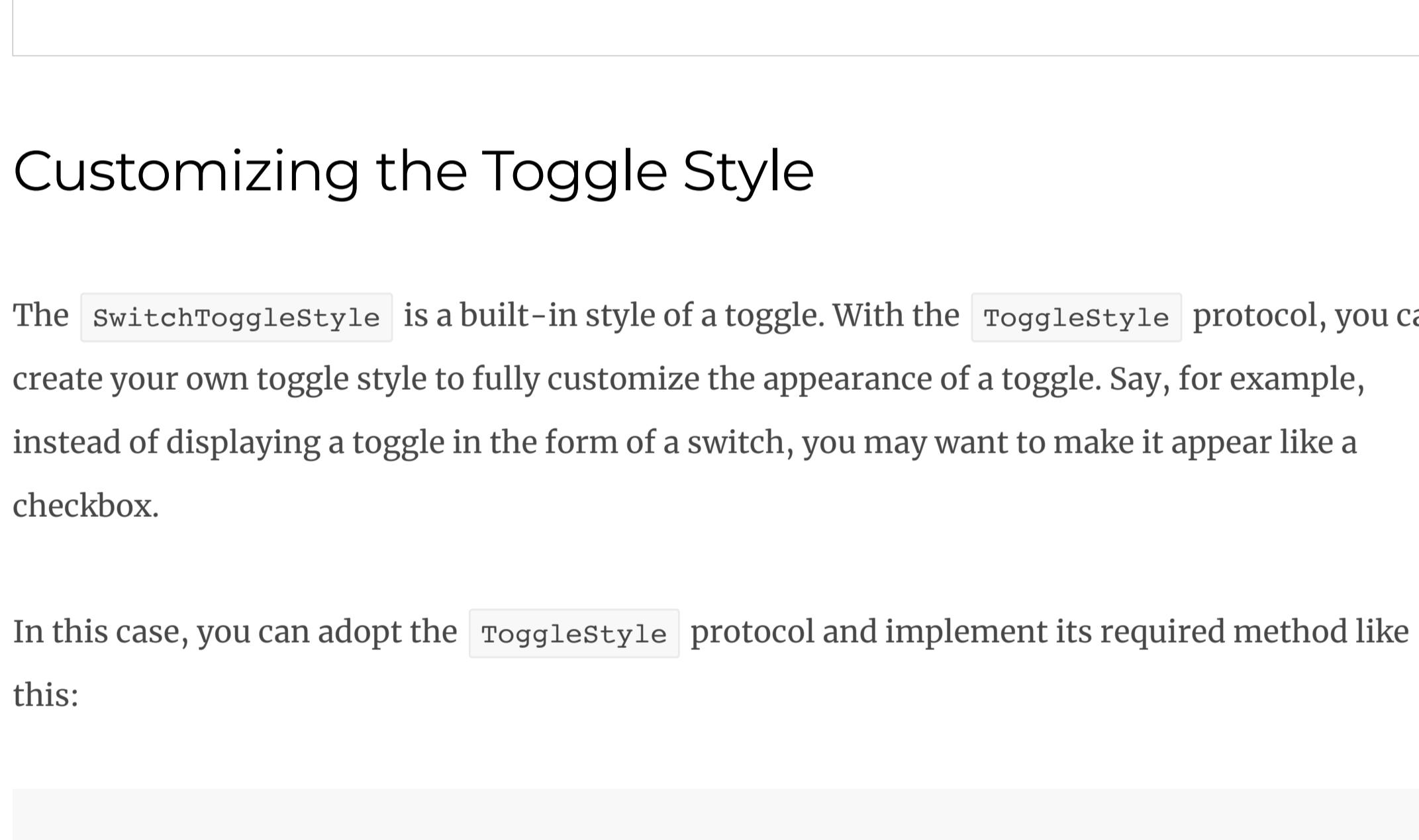
Earlier, we wrote a [tutorial](#) on customizing the appearance of a SwiftUI button. This week, let's see how to customize the look & feel of a SwiftUI toggle. If you've read the tutorial, you know we can use a protocol called `ButtonStyle` to create your own button style for customizations. For toggles, the customization can be achieved by adopting the `ToggleStyle` protocol. Later in this tutorial, we will make use of the protocol to transform a toggle from the standard appearance to a checkbox.

First, let's start with the basics. In SwiftUI, you create a toggle by using the `Toggle` component. When initializing a Toggle view, it requires you to specify a binding to a boolean property and a view that describes the purpose of the toggle. The binding controls the ON and OFF status of the toggle. Here is a sample code snippet for creating a standard toggle:

```
struct ContentView: View {
    @State var airplaneMode = true

    var body: some View {
        Toggle(isOn: $airplaneMode, label: {
            Image(systemName: "airplane")
            Text("Airplane Mode")
        })
        .padding()
    }
}
```

If you place the code in Xcode and preview it in the canvas, you should see a switch, which is the default appearance of a toggle.



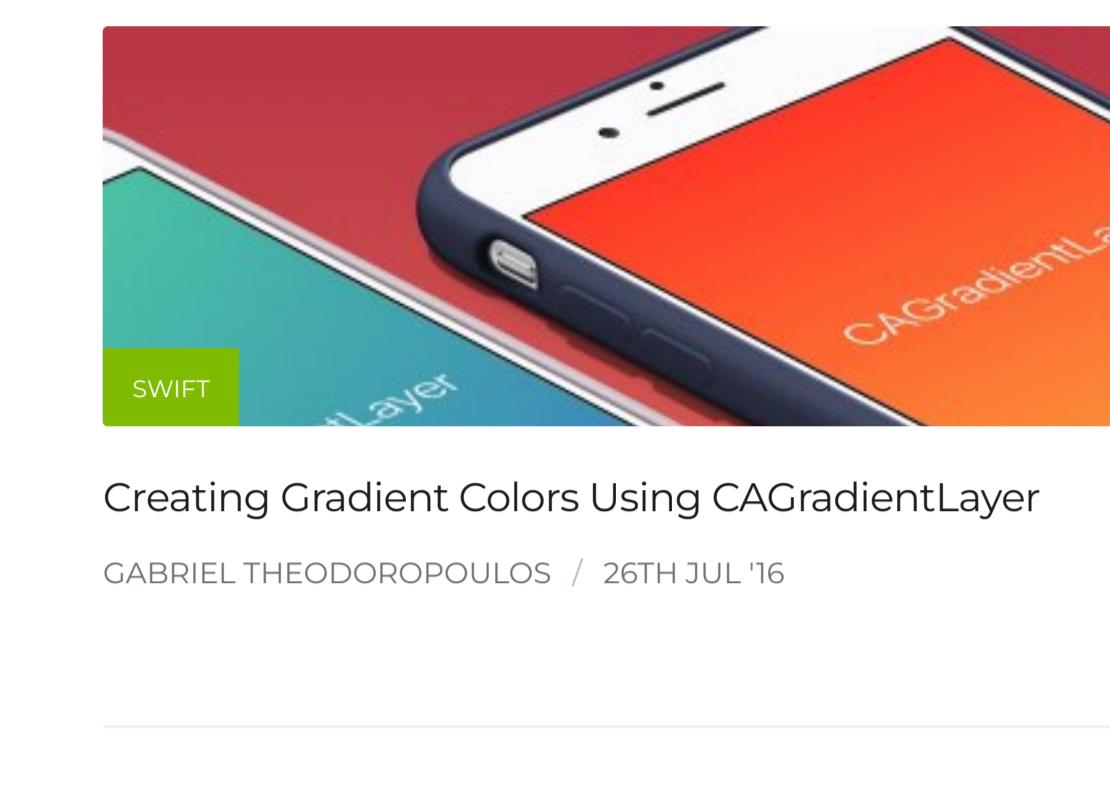
## Changing the Switch Color

In iOS 13, Apple doesn't provide APIs for changing the tint color of the switch. However, you can easily alter the tint color in [iOS 14](#) with a line of code like this:

```
Toggle(isOn: $airplaneMode, label: {
    Image(systemName: "airplane")
    Text("Airplane Mode")
})
.padding()
.toggleStyle(SwitchToggleStyle(tint: .orange))
```

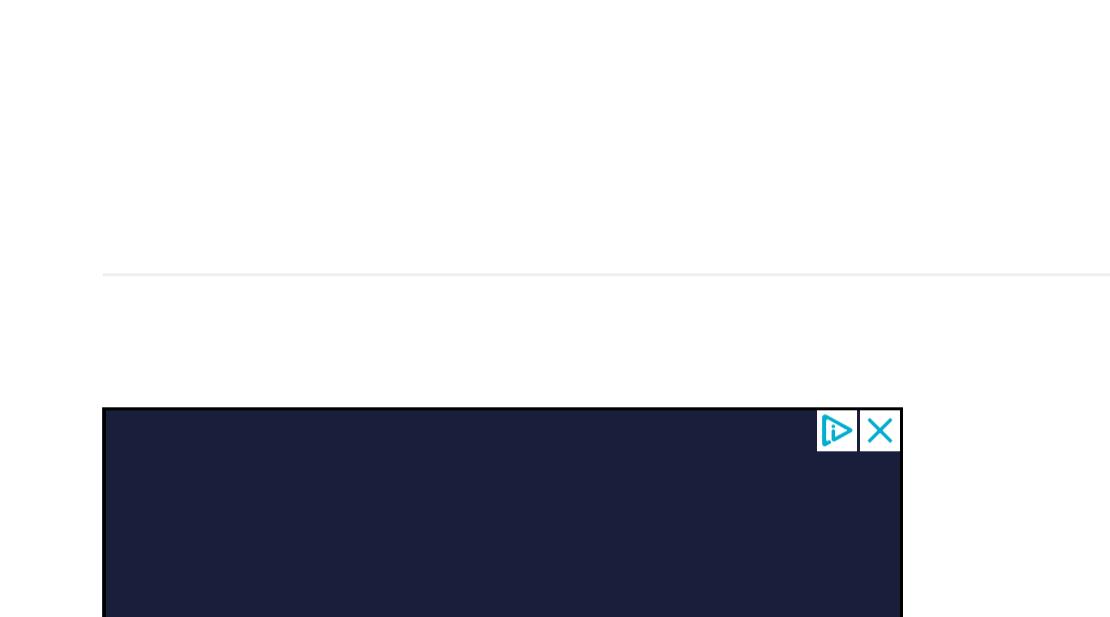
You can attach a `.toggleStyle` modifier to the `Toggle` component and pass it with an instance of `SwitchToggleStyle` with an alternate tint color.

[swiftui-toggle-tint-color](#)



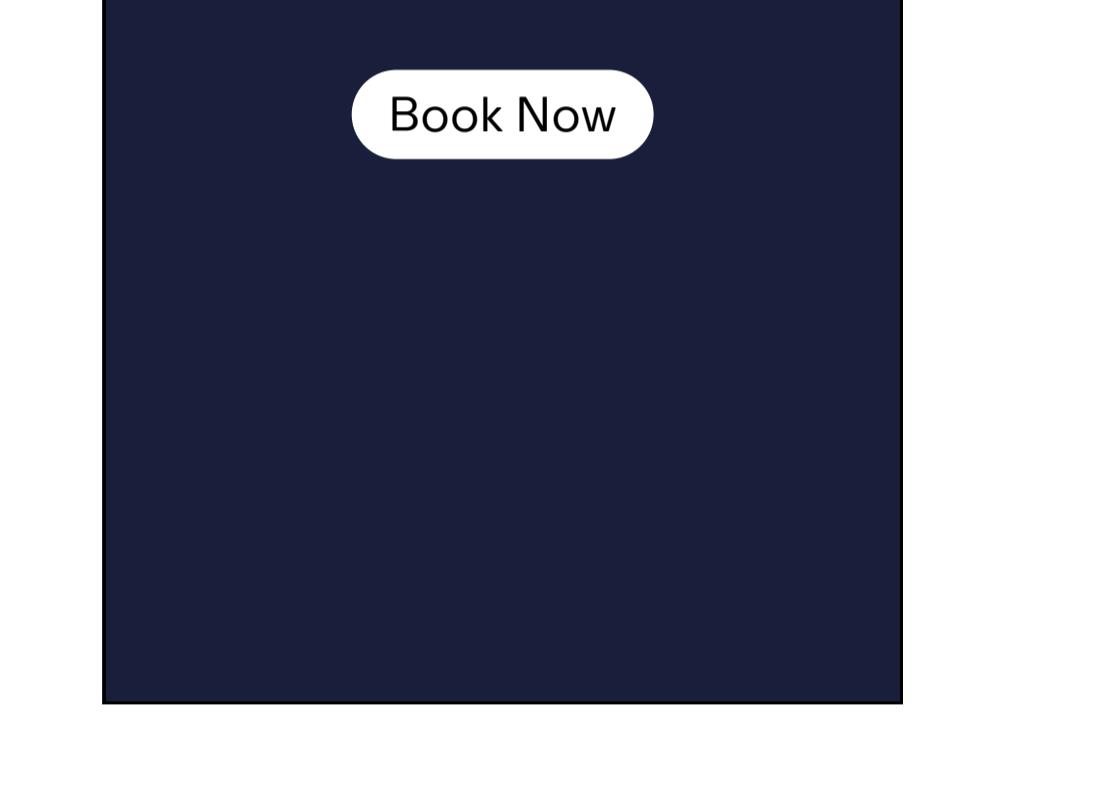
How To Style SwiftUI Buttons in iOS 15

SIMON NG | 2ND AUG '21



Building a Simple ARKit Demo with SceneKit in Swift 4 and Xcode 9

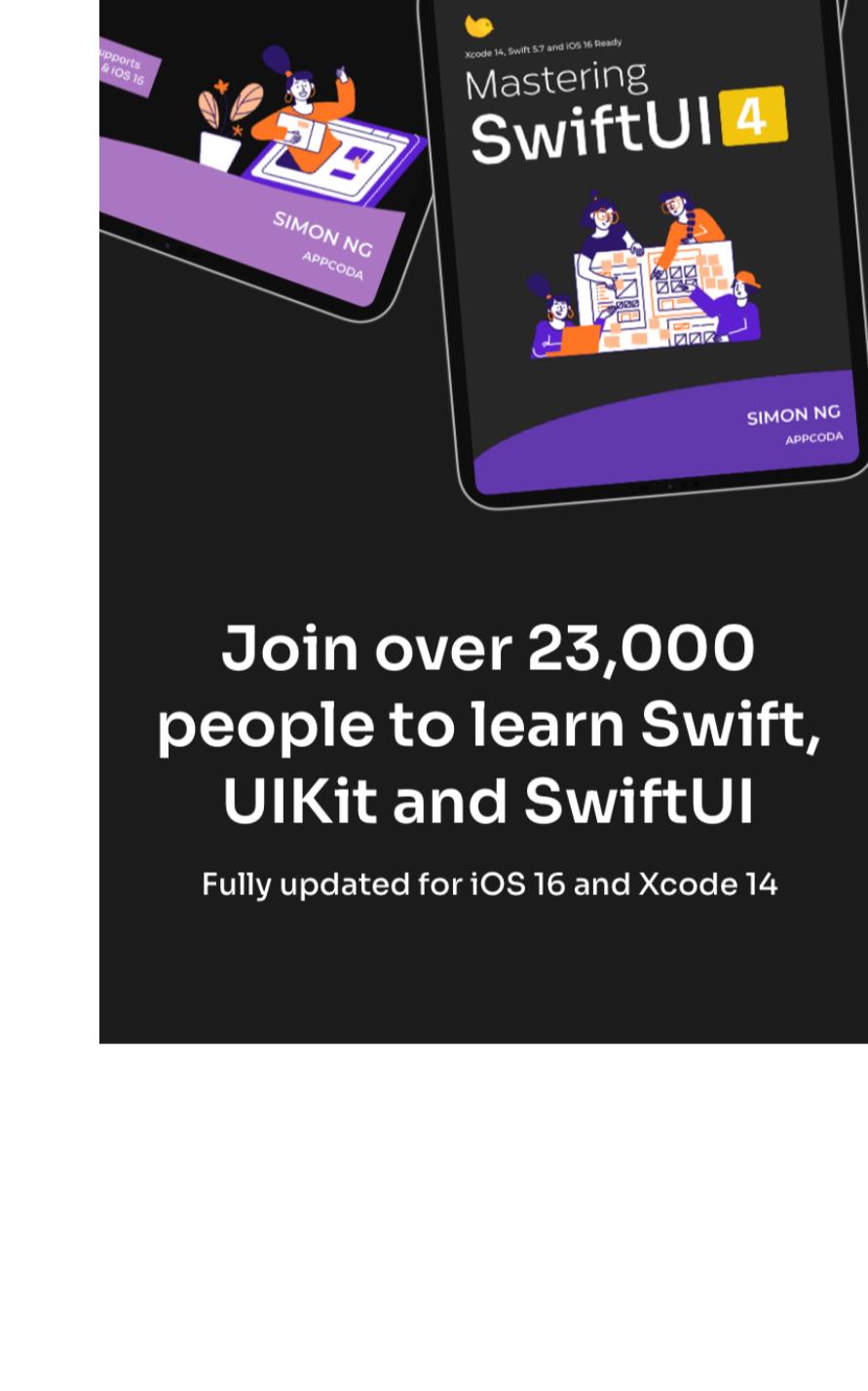
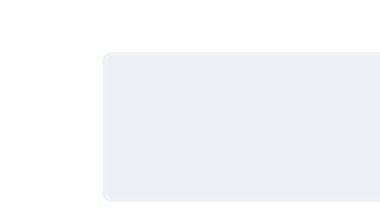
JAYVEN NHAN | 29TH SEP '17



Creating Gradient Colors Using CAGradientLayer

GABRIEL THEODOROPoulos | 26TH JUL '16

## CONNECT WITH US



## MORE TUTORIALS

> iOS

> watchOS

> tvOS

> Swift

SEARCH

## OUR COURSE/BOOKS



Join over 23,000 people to learn Swift, UIKit and SwiftUI

Fully updated for iOS 16 and Xcode 14

## Customizing the Toggle Style

The `SwitchToggleStyle` is a built-in style of a toggle. With the `ToggleStyle` protocol, you can create your own toggle style to fully customize the appearance of a toggle. Say, for example, instead of displaying a toggle in the form of a switch, you may want to make it appear like a checkbox.

In this case, you can adopt the `ToggleStyle` protocol and implement its required method like this:

```
struct CheckboxStyle: ToggleStyle {
    func makeBody(configuration: Self.Configuration) -> some View {
        return HStack {
            configuration.label
            Spacer()
            Image(systemName: configuration.isOn ? "checkmark.circle.fill" : "circle")
                .resizable()
                .frame(width: 24, height: 24)
                .foregroundColor(configuration.isOn ? .purple : .gray)
                .font(.system(size: 20, weight: .bold, design: .default))
                .onTapGesture {
                    configuration.isOn.toggle()
                }
        }
    }
}
```

You just need to implement the `makeBody` function and create the view for the toggle. In the code above, we utilize the SF symbols to create a circular checkbox. The `makeBody` function provides the `configuration` parameter for you to access the label and control the status of the toggle.

Once you created your own toggle style, you can apply it by attaching the `toggleStyle` modifier like this:

```
Toggle(isOn: $airplaneMode, label: {
    Image(systemName: "airplane")
    Text("Airplane Mode")
})
.padding()
.toggleStyle(CheckboxStyle())
```

Now we've completely changed the style of the toggle by transforming it from a standard switch to a checkbox.

[swiftui-toggle-checkbox-style](#)

AppCoda is one of the leading iOS programming communities. Our aim is to teach everyone how to build apps with quality and easy-to-read tutorials. Learn by doing is the heart of our learning materials.

MEET APPCODA

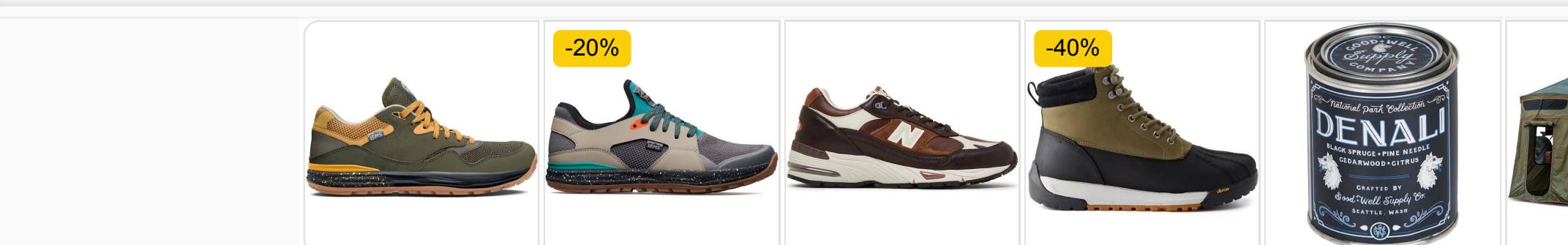
About

Our Team

Write for Us

Advertise

## RELATED ARTICLES



PREVIOUS POST

Deploying Mac Apps Outside App Store: How to Remember User Intent for Folders

NEXT POST

iOS 14 Updates in UIKit: ColorPicker, Datepicker, Menus and Actions

