Structure

Toggle

A control that toggles between on and off states.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | tvOS 13.0+ | visionOS 1.0+ | watchOS 6.0+

```
struct Toggle<Label> where Label : View
```

Mentioned in

- Building and customizing the menu bar with SwiftUI
- Declaring a custom view
- Laying out a simple view
- Populating SwiftUI menus with adaptive controls

Overview

You create a toggle by providing an `isOn` binding and a label. Bind `isOn` to a Boolean property that determines whether the toggle is on or off. Set the label to a view that visually describes the purpose of switching between toggle states. For example:

```
@State private var vibrateOnRing = false

var body: some View {
    Toggle(isOn: $vibrateOnRing) {
        Text("Vibrate on Ring")
    }
}
```

For the common case of `Label` based labels, you can use the convenience initializer that takes a title string (or localized string key) and the name of a system image:

```
@State private var vibrateOnRing = true

var body: some View {
    Toggle(
        "Vibrate on Ring",
        systemImage: "dot.radiowaves.left.and.right",
        isOn: $vibrateOnRing
    )
}
```

For text-only labels, you can use the convenience initializer that takes a title string (or localized string key) as its first parameter, instead of a trailing closure:

```
@State private var vibrateOnRing = true

var body: some View {
    Toggle("Vibrate on Ring", isOn: $vibrateOnRing)
}
```

For cases where adding a subtitle to the label is desired, use a view builder that creates multiple `Text` views where the first text represents the title and the second text represents the subtitle:

```
@State private var vibrateOnRing = false

var body: some View {
    Toggle(isOn: $vibrateOnRing) {
        Text("Vibrate on Ring")
        Text("Enable vibration when the phone rings")
    }
}
```

Note

This behavior does not apply to `button`.

Styling toggles

Toggles use a default style that varies based on both the platform and the context. For more information, read about the `automatic` toggle style.

You can customize the appearance and interaction of toggles by applying styles using the `toggleStyle(_:)` modifier. You can apply built-in styles, like `switch`, to either a toggle, or to a view hierarchy that contains toggles:

```
VStack {
    Toggle("Vibrate on Ring", isOn: $vibrateOnRing)
    Toggle("Vibrate on Silent", isOn: $vibrateOnSilent)
}
.toggleStyle(.switch)
```

You can also define custom styles by creating a type that conforms to the `ToggleStyle` protocol.

Topics

Creating a toggle

`init(_:isOn:)`

Creates a toggle that generates its label from a localized string key.

`init(isOn: Binding<Bool>, label: () -> Label)`

Creates a toggle that displays a custom label.

`init(_:image:isOn:)`

Creates a toggle that generates its label from a localized string key and image resource.

`init(_:systemImage:isOn:)`

Creates a toggle that generates its label from a localized string key and system image.

Creating a toggle for a collection

`init(_:sources:isOn:)`

Creates a toggle representing a collection of values that generates its label from a localized string key.

`init<C>(sources: C, isOn: KeyPath<C.Element, Binding<Bool>>, label: () -> Label)`

Creates a toggle representing a collection of values with a custom label.

`init(_:image:sources:isOn:)`

Creates a toggle representing a collection of values that generates its label from a localized string key and image resource.

`init(_:systemImage:sources:isOn:)`

Creates a toggle representing a collection of values that generates its label from a localized string key and system image.

Creating a toggle from a configuration

`init(ToggleStyleConfiguration)`

Creates a toggle based on a toggle style configuration.

Creating a toggle for an App Intent

`init<I>(isOn: Bool, intent: I, label: () -> Label)`

Creates a toggle performing an `AppIntent`.

`init(_:isOn:intent:)`

Creates a toggle performing an `AppIntent` and generates its label from a localized string key.

Relationships

Conforms To

`View`

See Also

Getting numeric inputs

`struct Slider`

A control for selecting a value from a bounded linear range of values.

`struct Stepper`

A control that performs increment and decrement actions.

`func toggleStyle<S>(S) -> some View`

Sets the style for toggles in a view hierarchy.