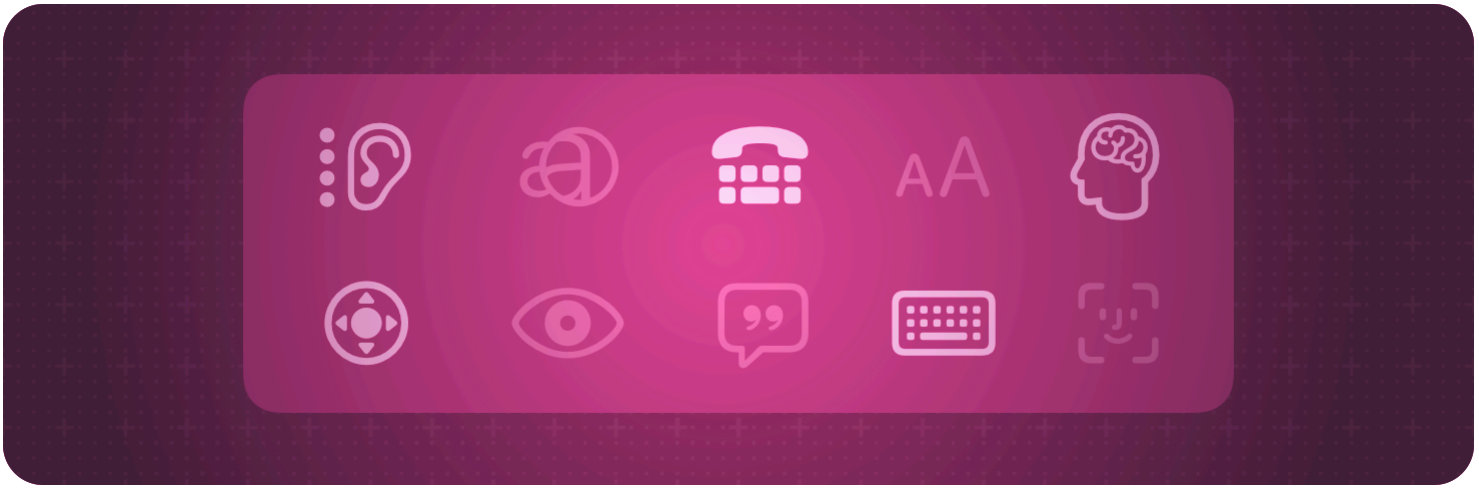
API Collection

Accessibility fundamentals

Make your SwiftUI apps accessible to everyone, including people with disabilities.

Overview

Like all Apple UI frameworks, SwiftUI comes with built-in accessibility support. The framework introspects common elements like navigation views, lists, text fields, sliders, buttons, and so on, and provides basic accessibility labels and values by default. You don't have to do any extra work to enable these standard accessibility features.



SwiftUI also provides tools to help you enhance the accessibility of your app. To find out what enhancements you need, try using your app with accessibility features like VoiceOver, Voice Control, and Switch Control, or get feedback from users of your app that regularly use these features. Then use the accessibility view modifiers that SwiftUI provides to improve the experience. For example, you can explicitly add accessibility labels to elements in your UI using the `accessibilityLabel(_:)` or the `accessibilityValue(_:)` view modifier.

Customize your use of accessibility modifiers for all the platforms that your app runs on. For example, you may need to adjust the accessibility elements for a companion Apple Watch app that shares a common code base with an iOS app. If you integrate AppKit or UIKit controls in SwiftUI,

expose any accessibility labels and make them accessible from your `NSViewRepresentable` or `UIViewRepresentable` views, or provide custom accessibility information if the underlying accessibility labels aren't available.

For design guidance, see Accessibility in the Human Interface Guidelines.

Topics

Essentials

{} Creating accessible views

Make your app accessible to everyone by applying accessibility modifiers to your SwiftUI views.

Creating accessible elements

`func accessibilityElement(children: AccessibilityChildBehavior) -> some View`

Creates a new accessibility element, or modifies the `AccessibilityChildBehavior` of the existing accessibility element.

`func accessibilityChildren<V>(children: () -> V) -> some View`

Replaces the existing accessibility element's children with one or more new synthetic accessibility elements.

`func accessibilityRepresentation<V>(representation: () -> V) -> some View`

Replaces one or more accessibility elements for this view with new accessibility elements.

`struct AccessibilityChildBehavior`

Defines the behavior for the child elements of the new parent element.

Identifying elements

`func accessibilityIdentifier(String) -> ModifiedContent<Self, AccessibilityAttachmentModifier>`

Uses the string you specify to identify the view.

`func accessibilityIdentifier(String, isEnabled: Bool) -> ModifiedContent<Self, AccessibilityAttachmentModifier>`

Uses the string you specify to identify the view.

Hiding elements

```
func accessibilityHidden(Bool) -> ModifiedContent<Self, AccessibilityAttachmentModifier>
```

Specifies whether to hide this view from system accessibility features.

```
func accessibilityHidden(Bool, isEnabled: Bool) -> ModifiedContent<Self, AccessibilityAttachmentModifier>
```

Specifies whether to hide this view from system accessibility features.

Supporting types

```
struct AccessibilityTechnologies
```

Accessibility technologies available to the system.

```
struct AccessibilityAttachmentModifier
```

A view modifier that adds accessibility properties to the view

---

See Also

Accessibility

- Accessible appearance

  Enhance the legibility of content in your app's interface.

- Accessible controls

  Improve access to actions that your app can undertake.

- Accessible descriptions

  Describe interface elements to help people understand what they represent.

- Accessible navigation

  Enable users to navigate to specific user interface elements using rotors.