API Collection

Accessibility for AppKit

Make your AppKit apps accessible to everyone who uses macOS.

Overview

Making your app accessible means making it usable by everyone. By designing your app with accessibility in mind, you make it possible for everyone to enjoy your app. For more information, see Accessibility.

AppKit controls and views come with built-in accessibility, providing an accessible user experience by default. Typically, you don’t need to do extra work to enable the standard accessibility features.

In some cases, you might want to modify the default values to better represent your app, to provide additional context, or to modify the user’s flow through the app. AppKit makes these customizations straightforward, involving a few lines of code or Interface Builder adjustments as you define your user interface. For more information about customizing accessibility for AppKit elements, see `NSAccessibilityProtocol`.

If your app contains custom user interface elements that subclass `NSView`, enhance the accessibility of those elements using the role-based protocols in Custom Controls. If your app contains custom user interface elements that don’t inherit from `NSView` or one of the other AppKit classes with built-in accessibility, make those elements accessible by subclassing `NSAccessibilityElement`.

If you build your app with SwiftUI, see Accessibility modifiers.

Topics

Essentials

{} Integrating accessibility into your app

Make your app more accessible to users with disabilities by adding accessibility features.

{} Accessibility design for Mac Catalyst

Improve navigation in your app by using keyboard shortcuts and accessibility containers.

AppKit Elements

If you're using a standard AppKit user interface element, you can override its existing accessibility attributes or use it as-is.

`protocol NSAccessibilityProtocol`

The complete list of properties and methods for accessible elements.

`struct NSAccessibility`

A namespace for accessibility symbols for AppKit apps.

Custom View Subclasses

If you're subclassing an AppKit view to create a custom user interface element, you can adopt one or more role-specific protocols to enhance that element's accessibility.

Custom Controls

Support accessibility for custom user interface elements by adopting a role-specific protocol and implementing its methods.

Accessibility Functions

Global accessibility functions for custom views and controls.

Custom Elements

If you're designing a completely custom user interface element that doesn't subclass an AppKit view, you must subclass the accessibility element class.

`class NSAccessibilityElement`

The basic infrastructure necessary for interacting with an assistive app.

Accessibility Types

`struct Action`

Constants that describe types of actions.

`struct AnnotationAttributeKey`

Keys for annotation attributes.

`enum NSAccessibilityAnnotationPosition`

Constants that specify the position where the annotation applies.

`struct Attribute`

Constants that describe attributes.

`struct FontAttributeKey`

Keys for font attributes.

`enum NSAccessibilityOrientation`

Values that indicate the orientation of accessibility elements, such as scroll bars and split views.

`struct OrientationValue`

Values that indicate the orientation of user interface elements, such as scroll bars and split views.

`struct ParameterizedAttribute`

Values that describe parameterized attributes.

`struct Role`

Values that describe types of objects that accessibility elements represent.

`enum NSAccessibilityRulerMarkerType`

Values that indicate the marker type of an accessibility element.

`struct RulerMarkerTypeValue`

Values that describe ruler marker types.

`struct RulerUnitValue`

Values that indicate the unit values of a ruler or layout area.

`struct SortDirectionValue`

Values that indicate the sort direction of a column.

`enum NSAccessibilitySortDirection`

Values that indicate the sort direction of a column.

`struct Subrole`

Values that describe specialized object subtypes that accessibility elements represent.

`enum NSAccessibilityUnits`

Values that indicate the unit values of a ruler or layout area.

See Also

User Interactions

Mouse, Keyboard, and Trackpad

Handle events related to mouse, keyboard, and trackpad input.

Menus, Cursors, and the Dock

Implement menus and cursors to facilitate interactions with your app, and use your app’s Dock tile to convey updated information.

Gestures

Encapsulate your app’s event-handling logic in gesture recognizers so that you can reuse that code throughout your app.

Touch Bar

Display interactive content and controls in the Touch Bar.

Drag and Drop

Support the direct manipulation of your app’s content using drag and drop.