UIKit / UISlider

Class

UISlider

A control for selecting a single value from a continuous range of values.

iOS 2.0+ | iPadOS 2.0+ | Mac Catalyst 13.1+ | visionOS 1.0+

```
@MainActor
class UISlider
```

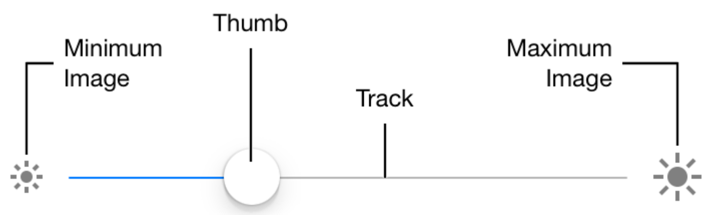
Mentioned in

- Choosing a user interface idiom for your Mac app
- Attaching gesture recognizers to UIKit controls
- Responding to control-based events using target-action

Overview

As you move the *thumb* of a slider, it passes its updated value to any actions attached to it. The appearance of sliders is configurable; you can tint the track and the thumb, and provide images to appear at the ends of the slider. You can add sliders to your interface programmatically or by using Interface Builder.

The following image shows the terms used to describe the constituent parts of a `UISlider` object in a left-to-right configuration.



To add a slider to your interface:

- Specify the range of values the slider represents.
- Optionally, configure the appearance of the slider with appropriate tint colors, and limit images.
- Connect one or more action methods to the slider.
- Set up Auto Layout rules to govern the size and position of the slider in your interface.

Respond to user interaction

Sliders use the target-action design pattern to notify your app when the user moves the slider. To be notified when the slider's value changes, register your action method with the `valueChanged` event. At runtime, the slider calls your method in response to the user changing the slider's value.

By default, the slider sends value-changed events continuously as the user moves the slider's thumb control. Setting the `isContinuous` property to `false` causes the slider to send an event only when the user releases the slider's thumb control, setting the final value.

You connect a slider to your action method by using the `addTarget(_:action:for:)` method or by creating a connection in Interface Builder. The signature of an action method takes one of three forms, as shown in the following code. Choose the form that provides the information that you need to respond to the value change in the slider.

Swift Objective-C

```swift
@IBAction func doSomething()
@IBAction func doSomething(sender: UISlider)
@IBAction func doSomething(sender: UISlider, forEvent event: UIEvent)
```

Debug sliders

When debugging issues with sliders, follow these tips to avoid common pitfalls:

- **Use either a custom tint color or a custom image, but not both.** When customizing slider appearance with images or tint, use one option or the other, but not both. Conflicting settings for track and thumb appearance are resolved in favor of the most recently set value. For example, setting a new minimum track image for any state clears any custom tint color you may have provided for minimum track images. Similarly, setting the thumb tint color removes any custom thumb images associated with the slider.
- **The current value must be between the minimum and maximum values.** If you try to programmatically set a slider's current value to be below the minimum or above the maximum,

it’s set to the minimum or maximum instead. However, if you set the value beyond the range of the minimum or maximum in Interface Builder, the minimum or minimum values are updated instead.

- **Set custom images for all control states.** If you use custom track and thumb images for your slider, remember to set an image for every possible `UIControl.State`. Any control state that doesn’t have a corresponding custom image assigned to it displays the standard image instead. If you set one custom image, be sure to set them all.

Configure slider attributes in Interface Builder

The following table lists the core attributes that you configure for sliders in Interface Builder.

Attribute	Description
Value Minimum / Maximum	Specifies the values attached to the endpoints of the slider, the minimum representing the leading end of the slider and the maximum representing the trailing end. Access these values at runtime using the `minimumValue` and `maximumValue` properties.
Value Current	Represents the initial value of the slider. The value must be between the minimum and maximum values. Access this value at runtime with the `value` property.

The following table lists the attributes that control the appearance of a slider.

Attribute	Description
Min Image	Specifies the image displayed at the leading end of the slider. If blank, no image is displayed. Access this value at runtime with the `minimumValueImage` property.
Max Image	Specifies the image displayed at the trailing end of the slider. If blank, no image is displayed. Access this value at runtime with the `maximumValueImage` property.
Min Track Tint	Specifies the tint color of the track to the leading side of the slider’s thumb. The value defaults to the slider’s inherited tint color. Access this value at runtime with the `minimumTrackTintColor` property.
Max Track Tint	Specifies the tint color of the track to the trailing side of the slider’s thumb. Access this value at runtime with the `maximumTrackTintColor` property.

Attribute	Description
Thumb Tint	Controls the tint color of the slider's thumb. Access this value at runtime with the `thumbTintColor` property.

The following table lists the attributes that configure the events associated with a slider.

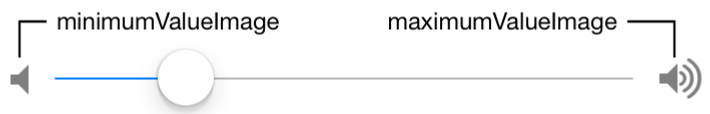
Attribute	Description
Events: Continuous Updates	Controls when attached actions are triggered: when checked, action events are called whenever the thumb is moved during user interaction. When not checked, attached actions are triggered only on completion of user interaction. Access this value at runtime with the `isContinuous` property.

For information about the sliders's inherited Interface Builder attributes, see `UIControl` and `UIView`.

Customize the slider's appearance

Use Auto Layout to specify the position and width of a slider. The intrinsic height is determined by the intrinsic heights of the minimum and maximum images, if present. The width of the track automatically adjusts to accommodate the minimum and maximum images.

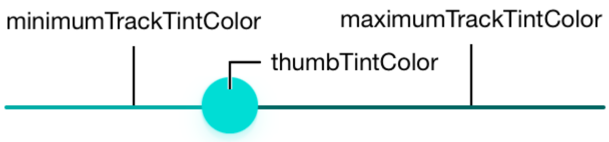
The most common way to customize the slider's appearance is to provide custom minimum and maximum value images. These images sit at either end of the slider control and indicate which value that end of the slider represents. Set the values of the `minimumValueImage` and `maximumValueImage` properties to appropriate `UIImage` objects to display images at the ends of the slider. The following image shows a slider configured with minimum and maximum images that imply volume adjustment.



Note

Sliders respond to user interaction with dynamic effects and appearance. If you set custom tint colors for the track or thumb, the slider maintains this behavior. If you use images to customize the appearance of the track, then the slider doesn't apply the dynamic effects or alter the appearance.

To set custom tint colors for both the track and the thumb of a slider, use the minimumTrackTintColor, maximumTrackTintColor, and thumbTintColor properties, as shown in the following image.



By default, the minimum track tint color defers to the tint color of the slider control.

To completely change the appearance of the slider, you can specify images for the thumb and the track. Provide images for each of the control states (normal, highlighted, and so on) with the setMinimumTrackImage(_:for:), setMaximumTrackImage(_:for:), and setThumbImage(_:for:) methods. Set the capInsets property for the track images to facilitate horizontal stretching. To access the images used in the current control state, use the currentMinimumTrackImage, currentMaximumTrackImage, and currentThumbImage properties, as shown in the following image.

Provide localized strings

Sliders have no special properties related to internationalization. However, if you use a slider with a label, make sure you provide localized strings for the label.

Sliders automatically adjust to the appropriate interface direction, ensuring that the minimum end of the slider is always at the leading end and the maximum end at the trailing end. If you override minimumValueImageRect(forBounds:) or maximumValueImageRect(forBounds:) in a subclass of UISlider, be sure to take the user interface layout direction into account.

For more information, see Internationalization and Localization Guide.

Make sliders accessible

Sliders are accessible by default. The default accessibility traits for a slider are User Interaction Enabled and Adjustable.

When enabled on a device, VoiceOver speaks the accessibility label, value, traits, and hint to the user. VoiceOver speaks this information when a user swipes up and down (not left and right) over the slider. For example, using the Ringer and Alerts volume slider (Settings > Sounds > Ringer and Alerts), VoiceOver speaks the following:

```
"Sound volume: 13 percent. Adjustable. Swipe up or down with one finger to adjust th
```

For more information about making iOS controls accessible, see the accessibility information in `UIControl`. For general information about making your interface accessible, see Accessibility Programming Guide for iOS.

Topics

Accessing the slider’s value

`var value: Float`

The slider’s current value.

`func setValue(Float, animated: Bool)`

Sets the slider’s current value, allowing you to animate the change visually.

Accessing the slider’s value limits

`var minimumValue: Float`

The minimum value of the slider.

`var maximumValue: Float`

The maximum value of the slider.

Modifying the slider’s behavior

`var isContinuous: Bool`

A Boolean value indicating whether changes in the slider’s value generate continuous update events.

`var behavioralStyle: UIBehavioralStyle`

The style that determines how the slider behaves.

`var preferredBehavioralStyle: UIBehavioralStyle`

The preferred behavioral style.

`enum UIBehavioralStyle`

Constants that indicate how a control behaves in apps built with Mac Catalyst.

Changing the slider’s style

`var sliderStyle: UISlider.Style`

`enum Style`

Changing the slider’s appearance

`var minimumValueImage: UIImage?`

The image that represents the slider’s minimum value.

`var maximumValueImage: UIImage?`

The image representing the slider’s maximum value.

`var minimumTrackTintColor: UIColor?`

The color used to tint the default minimum track images.

`var currentMinimumTrackImage: UIImage?`

The minimum track image currently being used to render the slider.

`func minimumTrackImage(for: UIControl.State) -> UIImage?`

Returns the minimum track image associated with the specified control state.

`func setMinimumTrackImage(UIImage?, for: UIControl.State)`

Assigns a minimum track image to the specified control states.

`var maximumTrackTintColor: UIColor?`

The color used to tint the default maximum track images.

`var currentMaximumTrackImage: UIImage?`

Contains the maximum track image currently being used to render the slider.

`func maximumTrackImage(for: UIControl.State) -> UIImage?`

Returns the maximum track image associated with the specified control state.

`func setMaximumTrackImage(UIImage?, for: UIControl.State)`

Assigns a maximum track image to the specified control states.

`var thumbTintColor: UIColor?`

The color used to tint the default thumb images.

`var currentThumbImage: UIImage?`

The thumb image currently being used to render the slider.

`func thumbImage(for: UIControl.State) -> UIImage?`

Returns the thumb image associated with the specified control state.

```
func setThumbImage(UIImage?, for: UIControl.State)
```

Assigns a thumb image to the specified control states.

Configuring the track

```
var trackConfiguration: UISlider.TrackConfiguration?
```

```
struct TrackConfiguration
```

Overrides for subclasses

```
func maximumValueImageRect(forBounds: CGRect) -> CGRect
```

Returns the drawing rectangle for the maximum value image.

```
func minimumValueImageRect(forBounds: CGRect) -> CGRect
```

Returns the drawing rectangle for the minimum value image.

```
func trackRect(forBounds: CGRect) -> CGRect
```

Returns the drawing rectangle for the slider’s track.

```
func thumbRect(forBounds: CGRect, trackRect: CGRect, value: Float) ->
CGRect
```

Returns the drawing rectangle for the slider’s thumb image.

Relationships

Inherits From

`UIControl`

Conforms To

`CALayerDelegate`
`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`

```
Hashable
NSCoding
NSObjectProtocol
NSTouchBarProvider
Sendable
SendableMetatype
UIAccessibilityIdentification
UIActivityItemsConfigurationProviding
UIAppearance
UIAppearanceContainer
UIContextMenuInteractionDelegate
UICoordinateSpace
UIDynamicItem
UIFocusEnvironment
UIFocusItem
UIFocusItemContainer
UILargeContentViewerItem
UIPasteConfigurationSupporting
UIPopoverPresentationControllerSourceItem
UIResponderStandardEditActions
UITraitChangeObservable
UITraitEnvironment
UIUserActivityRestoring
```

See Also

Controls

Responding to control-based events using target-action

Handle user input by connecting buttons, sliders, and other controls to your app's code using the target-action design pattern.

`class UIControl`

The base class for controls, which are visual elements that convey a specific action or intention in response to user interactions.

`class UIButton`

A control that executes your custom code in response to user interactions.

`class UIColorWell`

A control that displays a color picker.

`class UIDatePicker`

A control for inputting date and time values.

`class UIPageControl`

A control that displays a horizontal series of dots, each of which corresponds to a page in the app's document or other data-model entity.

`class UISegmentedControl`

A horizontal control that consists of multiple segments, each segment functioning as a discrete button.

`class UIStepper`

A control for incrementing or decrementing a value.

`class UISwitch`

A control that offers a binary choice, such as on/off.