Class

UISegmentedControl

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

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

```
@MainActor
class UISegmentedControl
```

Mentioned in

Attaching gesture recognizers to UIKit controls

Overview

A segmented control can display a title (an `NSString` object) or an image (`UIImage` object). The `UISegmentedControl` object automatically resizes segments to fit proportionally within their superview unless they have a specific width set. When you add and remove segments, you can request that the action be animated with sliding and fading effects.

You register the target-action methods for a segmented control using the `valueChanged` constant as shown below.

Swift | Objective-C

```
segmentedControl.addTarget(self, action: "action:", forControlEvents: .valueChanged)
```

How you configure a segmented control can affect its display behavior:

- If you set a segmented control to have a momentary style, a segment doesn't show itself as selected (blue background) when the user touches it. The disclosure button is always momentary and doesn't affect the actual selection.
- In versions of iOS prior to 3.0, if a segmented control has only two segments, then it behaves like a switch — tapping the currently-selected segment causes the other segment to be selected. In iOS 3.0 and later, tapping the currently-selected segment doesn't cause the other segment to be selected.

Customize appearance

You can customize the appearance of segmented controls using the methods listed in Customizing appearance. You can customize the appearance of all segmented controls using the appearance proxy (for example, `[UISegmentedControl appearance]`), or just of a single control.

When customizing appearance, in general, you should specify a value for the normal state of a property to be used by other states which don't have a custom value set. Similarly, when a property is dependent on the bar metrics (on the iPhone in landscape orientation, bars have a different height from standard), you should make sure you specify a value for `UIBarMetrics.default`.

In the case of the segmented control, appearance properties for `landscapePhone` are only respected for segmented controls in the smaller navigation and toolbars that are used in landscape orientation on the iPhone.

To provide complete customization, you need to provide divider images for different state combinations, using `setDividerImage(_:forLeftSegmentState:rightSegmentState:barMetrics:)`:

Swift | Objective-C

```
// Image between two unselected segments.
mySegmentedControl.setDividerImage(myImage, forLeftSegmentState: UIControlState.Norm
                                   rightSegmentState: UIControlState.Normal, barMetr

// Image between segment selected on the left and unselected on the right.
mySegmentedControl.setDividerImage(myImage, forLeftSegmentState: UIControlState.Sele
                                   rightSegmentState: UIControlState.Normal, barMetr

// Image between segment selected on the right and unselected on the left.
mySegmentedControl.setDividerImage(myImage, forLeftSegmentState: UIControlState.Norm
                                   rightSegmentState: UIControlState.Selected, barMe
```

Topics

Creating a segmented control

`init(items: [Any]?)`

Creates a segmented control with segments having the given titles or images.

`convenience init(frame: CGRect, actions: [UIAction])`

Creates a segmented control with the given frame and adds segments for the actions you specify.

`init(frame: CGRect)`

Creates an empty segmented control with the frame you specify.

`init?(coder: NSCoder)`

Creates a segmented control with data from an unarchiver.

Managing segment content

`func setImage(UIImage?, forSegmentAt: Int)`

Sets the content of a segment to a given image.

`func imageForSegment(at: Int) -> UIImage?`

Returns the image for a specific segment.

`func setTitle(String?, forSegmentAt: Int)`

Sets the title of a segment.

`func titleForSegment(at: Int) -> String?`

Returns the title of the specified segment.

Managing segment actions

`func actionForSegment(at: Int) -> UIAction?`

Fetches the action of the segment at the index you specify, if one exists.

`func setAction(UIAction, forSegmentAt: Int)`

Sets the action for the segment at the index you specify.

Managing segments

`var numberOfSegments: Int`

Returns the number of segments the segmented control has.

`func segmentIndex(identifiedBy: UIAction.Identifier) -> Int`

The index of a segment with an action that has an identifier matching the identifier you specify.

`func insertSegment(action: UIAction, at: Int, animated: Bool)`

Insert a segment with the action you specify at the given index.

`func insertSegment(with: UIImage?, at: Int, animated: Bool)`

Inserts a segment at the position you specify and gives it an image as content.

`func insertSegment(withTitle: String?, at: Int, animated: Bool)`

Inserts a segment at the position you specify and gives it a title as content.

`func removeAllSegments()`

Removes all segments of the segmented control.

`func removeSegment(at: Int, animated: Bool)`

Removes the segment you specify from the segmented control, optionally animating the transition.

`var selectedSegmentIndex: Int`

The index number that identifies the selected segment that the user last touched.

`class var noSegment: Int`

A segment index value indicating that there's no selected segment.

Managing segment behavior and appearance

`var isMomentary: Bool`

A Boolean value that determines whether segments in the segmented control show selected state.

`func setEnabled(Bool, forSegmentAt: Int)`

Enables the segment you specify.

`func isEnabledForSegment(at: Int) -> Bool`

Returns whether the indicated segment is enabled.

`func setContentOffset(CGSize, forSegmentAt: Int)`

Adjusts the offset for drawing the content (image or text) of the specified segment.

`func contentOffsetForSegment(at: Int) -> CGSize`

Returns the offset for drawing the content (image or text) of the segment you specify.

`func setWidth(CGFloat, forSegmentAt: Int)`

Sets the width of the segment at the index you specify.

`func widthForSegment(at: Int) -> CGFloat`

Returns the width of the segment at the index you specify.

`var apportionsSegmentWidthsByContent: Bool`

Indicates whether the control attempts to adjust segment widths based on their content widths.

Customizing appearance

`var selectedSegmentTintColor: UIColor?`

The color to use for highlighting the currently selected segment.

`func backgroundImage(for: UIControl.State, barMetrics: UIBarMetrics) -> UIImage?`

Returns the background image for a given state and bar metrics.

`func setBackgroundImage(UIImage?, for: UIControl.State, barMetrics: UIBarMetrics)`

Sets the background image for given state and bar metrics.

`func contentPositionAdjustment(forSegmentType: UISegmentedControl.Segment, barMetrics: UIBarMetrics) -> UIOffset`

Returns the positioning offset for a given segment and bar metrics.

`func setContentPositionAdjustment(UIOffset, forSegmentType: UISegmentedControl.Segment, barMetrics: UIBarMetrics)`

Sets the content positioning offset for a given segment and bar metrics.

`enum Segment`

Constants for specifying a segment in a control.

```
func dividerImage(forLeftSegmentState: UIControl.State, rightSegment
State: UIControl.State, barMetrics: UIBarMetrics) -> UIImage?
```

Returns the divider image used for a given combination of left and right segment states and bar metrics.

```
func setDividerImage(UIImage?, forLeftSegmentState: UIControl.State,
rightSegmentState: UIControl.State, barMetrics: UIBarMetrics)
```

Sets the divider image to use for a given combination of left and right segment states and bar metrics.

```
func titleTextAttributes(for: UIControl.State) -> [NSAttributedString.
Key : Any]?
```

Returns the text attributes of the title for a given control state.

```
func setTitleTextAttributes([NSAttributedString.Key : Any]?, for:
UIControl.State)
```

Sets the text attributes of the title for a given control state.

Relationships

Inherits From

UIControl

Conforms To

CALayerDelegate
CVarArg
Copyable
CustomDebugStringConvertible
CustomStringConvertible
Equatable
Hashable
NSCoding
NSObjectProtocol
NSTouchBarProvider
Sendable
SendableMetatype
UIAccessibilityIdentification

`UIActivityItemsConfigurationProviding`
`UIAppearance`
`UIAppearanceContainer`
`UIContextMenuInteractionDelegate`
`UICoordinateSpace`
`UIDynamicItem`
`UIFocusEnvironment`
`UIFocusItem`
`UIFocusItemContainer`
`UILargeContentViewerItem`
`UIPasteConfigurationSupporting`
`UIPopoverPresentationControllerSourceItem`
`UIResponderStandardEditActions`
`UISpringLoadedInteractionSupporting`
`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` `UISlider`

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

`class` `UIStepper`

A control for incrementing or decrementing a value.

`class` `UISwitch`

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