```
import Accessibility
import CoreGraphics
import CoreText
import DataDetection
import DeveloperToolsSupport
import Dispatch
import Foundation
import QuartzCore
import Symbols
import UIKit.DocumentManager
import UIKit.NSAdaptiveImageGlyph
import UIKit.NSAttributedString
import UIKit.NSDataAsset
import
UIKit.NSDiffableDataSourceSectionSnapshot
import UIKit.NSFileProviderExtension
import UIKit.NSIndexPath_UIKitAdditions
import
UIKit.NSItemProvider_UIKitAdditions
import UIKit.NSLayoutAnchor
import UIKit.NSLayoutConstraint
import UIKit.NSLayoutManager
import UIKit.NSParagraphStyle
import UIKit.NSShadow
import UIKit.NSStringDrawing
import UIKit.NSText
import UIKit.NSTextAttachment
import UIKit.NSTextContainer
import UIKit.NSTextContentManager
import UIKit.NSTextElement
import UIKit.NSTextLayoutFragment
import UIKit.NSTextLayoutManager
import UIKit.NSTextLineFragment
import UIKit.NSTextList
import UIKit.NSTextListElement
import UIKit.NSTextRange
import UIKit.NSTextSelection
import UIKit.NSTextSelectionNavigation
import UIKit.NSTextStorage
import
UIKit.NSTextViewportLayoutController
import UIKit.NSToolbar_UIKitAdditions
import UIKit.NSTouchBar_UIKitAdditions
import
UIKit.NSUserActivity_NSItemProvider
import UIKit.PrintKitUI
import UIKit.ShareSheet
import UIKit.UIAccelerometer
import UIKit.UIAccessibility
import UIKit.UIAccessibilityAdditions
import UIKit.UIAccessibilityConstants
import UIKit.UIAccessibilityContainer
import
UIKit.UIAccessibilityContentSizeCategoryI
mageAdjusting
import UIKit.UIAccessibilityCustomAction
import UIKit.UIAccessibilityCustomRotor
import UIKit.UIAccessibilityElement
import
UIKit.UIAccessibilityIdentification
import
UIKit.UIAccessibilityLocationDescriptor
import UIKit.UIAccessibilityZoom
import UIKit.UIAction
import UIKit.UIActionSheet
import UIKit.UIActivity
```

```
import
UIKit.UIActivityCollaborationModeRestrict
ion
import UIKit.UIActivityIndicatorView
import UIKit.UIActivityItemProvider
import UIKit.UIActivityItemsConfiguration
import
UIKit.UIActivityItemsConfigurationReading
import
UIKit.UIActivityItemsConfigurationReading
_ShareSheet
import UIKit.UIActivityViewController
import UIKit.UIAlert
import UIKit.UIAlertController
import UIKit.UIAlertView
import UIKit.UIAppearance
import UIKit.UIApplication
import UIKit.UIApplicationShortcutItem
import UIKit.UIAttachmentBehavior
import UIKit.UIBackgroundConfiguration
import UIKit.UIBandSelectionInteraction
import UIKit.UIBarAppearance
import UIKit.UIBarButtonItem
import UIKit.UIBarButtonItemAppearance
import UIKit.UIBarButtonItemGroup
import UIKit.UIBarCommon
import UIKit.UIBarItem
import UIKit.UIBehavioralStyle
import UIKit.UIBezierPath
import UIKit.UIBlurEffect
import UIKit.UIButton
import UIKit.UIButtonConfiguration
import UIKit.UICalendarSelection
import UIKit.UICalendarSelectionMultiDate
import
UIKit.UICalendarSelectionSingleDate
import
UIKit.UICalendarSelectionWeekOfYear
import UIKit.UICalendarView
import UIKit.UICalendarViewDecoration
import UIKit.UICanvasFeedbackGenerator
import UIKit.UICellAccessory
import UIKit.UICellConfigurationState
import UIKit.UICloudSharingController
import UIKit.UICollectionLayoutList
import UIKit.UICollectionView
import UIKit.UICollectionViewCell
import
UIKit.UICollectionViewCompositionalLayout
import UIKit.UICollectionViewController
import UIKit.UICollectionViewFlowLayout
import
UIKit.UICollectionViewItemRegistration
import UIKit.UICollectionViewLayout
import UIKit.UICollectionViewListCell
import
UIKit.UICollectionViewTransitionLayout
import UIKit.UICollectionViewUpdateItem
import UIKit.UICollisionBehavior
import UIKit.UIColor
import UIKit.UIColorPickerViewController
import UIKit.UIColorWell
import UIKit.UICommand
import
UIKit.UIConfigurationColorTransformer
import UIKit.UIConfigurationState
import UIKit.UIContentConfiguration
import UIKit.UIContentSizeCategory
import
UIKit.UIContentSizeCategoryAdjusting
import
UIKit.UIContentUnavailableButtonPropertie
s
import
UIKit.UIContentUnavailableConfiguration
import
UIKit.UIContentUnavailableConfigurationSt
ate
import
UIKit.UIContentUnavailableImageProperties
import
UIKit.UIContentUnavailableTextProperties
import UIKit.UIContentUnavailableView
import UIKit.UIContextMenuConfiguration
import UIKit.UIContextMenuInteraction
import UIKit.UIContextualAction
import UIKit.UIControl
import UIKit.UIDataDetectors
import UIKit.UIDataSourceTranslating
import UIKit.UIDatePicker
import UIKit.UIDeferredMenuElement
import UIKit.UIDevice
import UIKit.UIDiffableDataSource
import UIKit.UIDocument
import UIKit.UIDocumentBrowserAction
import
UIKit.UIDocumentBrowserViewController
import
UIKit.UIDocumentInteractionController
```

```
import UIKit.UIDocumentMenuViewController
import
UIKit.UIDocumentPickerExtensionViewContro
ller
import
UIKit.UIDocumentPickerViewController
import UIKit.UIDocumentProperties
import UIKit.UIDocumentViewController
import
UIKit.UIDocumentViewControllerLaunchOptio
ns
import UIKit.UIDragInteraction
import UIKit.UIDragItem
import UIKit.UIDragPreview
import UIKit.UIDragPreviewParameters
import UIKit.UIDragSession
import UIKit.UIDropInteraction
import UIKit.UIDynamicAnimator
import UIKit.UIDynamicBehavior
import UIKit.UIDynamicItemBehavior
import UIKit.UIEditMenuInteraction
import UIKit.UIEvent
import UIKit.UIEventAttribution
import UIKit.UIEventAttributionView
import UIKit.UIFeedbackGenerator
import UIKit.UIFieldBehavior
import UIKit.UIFindInteraction
import UIKit.UIFindSession
import UIKit.UIFocus
import UIKit.UIFocusAnimationCoordinator
import UIKit.UIFocusDebugger
import UIKit.UIFocusEffect
import UIKit.UIFocusGuide
import UIKit.UIFocusMovementHint
import UIKit.UIFocusSystem
import UIKit.UIFont
import UIKit.UIFontDescriptor
import UIKit.UIFontMetrics
import UIKit.UIFontPickerViewController
import
UIKit.UIFontPickerViewControllerConfigura
tion
import UIKit.UIFoundation
import UIKit.UIGeometry
import UIKit.UIGestureRecognizer
import UIKit.UIGestureRecognizerSubclass
import UIKit.UIGraphics
import UIKit.UIGraphicsImageRenderer
import UIKit.UIGraphicsPDFRenderer
import UIKit.UIGraphicsRenderer
import UIKit.UIGraphicsRendererSubclass
import UIKit.UIGravityBehavior
import UIKit.UIGuidedAccess
import UIKit.UIGuidedAccessRestrictions
import UIKit.UIHoverEffect
import UIKit.UIHoverEffectLayer
import UIKit.UIHoverGestureRecognizer
import UIKit.UIHoverStyle
import UIKit.UIImage
import UIKit.UIImageAsset
import UIKit.UIImageConfiguration
import UIKit.UIImagePickerController
import UIKit.UIImageReader
import UIKit.UIImageSymbolConfiguration
import UIKit.UIImageView
import UIKit.UIImpactFeedbackGenerator
```

```
import
UIKit.UIIndirectScribbleInteraction
import UIKit.UIInputView
import UIKit.UIInputViewController
import UIKit.UIInteraction
import UIKit.UIInterface
import UIKit.UIKey
import UIKit.UIKeyCommand
import UIKit.UIKeyConstants
import UIKit.UIKeyboardLayoutGuide
import UIKit.UIKitCore
import UIKit.UIKitDefines
import UIKit.UILabel
import UIKit.UILargeContentViewer
import UIKit.UILayoutGuide
import UIKit.UILetterformAwareAdjusting
import UIKit.UILexicon
import UIKit.UIListContentConfiguration
import UIKit.UIListContentImageProperties
import UIKit.UIListContentTextProperties
import UIKit.UIListSeparatorConfiguration
import UIKit.UILocalNotification
import UIKit.UILocalizedIndexedCollation
import UIKit.UILongPressGestureRecognizer
import UIKit.UIManagedDocument
import UIKit.UIMenu
import UIKit.UIMenuBuilder
import UIKit.UIMenuController
import UIKit.UIMenuDisplayPreferences
import UIKit.UIMenuElement
import UIKit.UIMenuLeaf
import UIKit.UIMenuSystem
import UIKit.UIMotionEffect
import UIKit.UINavigationBar
import UIKit.UINavigationBarAppearance
import UIKit.UINavigationController
import UIKit.UINavigationItem
import UIKit.UINib
import UIKit.UINibDeclarations
import UIKit.UINibLoading
import
UIKit.UINotificationFeedbackGenerator
import UIKit.UIOpenURLContext
import UIKit.UIOrientation
import UIKit.UIPageControl
import UIKit.UIPageControlProgress
import UIKit.UIPageViewController
import UIKit.UIPanGestureRecognizer
import UIKit.UIPasteConfiguration
import
UIKit.UIPasteConfigurationSupporting
import UIKit.UIPasteControl
import UIKit.UIPasteboard
import UIKit.UIPencilInteraction
import UIKit.UIPickerView
import UIKit.UIPinchGestureRecognizer
import UIKit.UIPointerAccessory
import UIKit.UIPointerInteraction
import UIKit.UIPointerLockState
import UIKit.UIPointerRegion
import UIKit.UIPointerStyle
import UIKit.UIPopoverBackgroundView
import UIKit.UIPopoverController
import
UIKit.UIPopoverPresentationController
import
```

```
UIKit.UIPopoverPresentationControllerSour
ceItem
import UIKit.UIPopoverSupport
import UIKit.UIPresentationController
import UIKit.UIPress
import UIKit.UIPressesEvent
import UIKit.UIPreviewInteraction
import UIKit.UIPreviewParameters
import UIKit.UIPrintError
import UIKit.UIPrintFormatter
import UIKit.UIPrintInfo
import UIKit.UIPrintInteractionController
import UIKit.UIPrintPageRenderer
import UIKit.UIPrintPaper
import UIKit.UIPrintServiceExtension
import UIKit.UIPrinter
import UIKit.UIPrinterPickerController
import UIKit.UIProgressView
import UIKit.UIPushBehavior
import
UIKit.UIReferenceLibraryViewController
import UIKit.UIRefreshControl
import UIKit.UIRegion
import UIKit.UIResponder
import
UIKit.UIResponder_UIActivityItemsConfigur
ation
import UIKit.UIRotationGestureRecognizer
import UIKit.UIScene
import UIKit.UISceneActivationConditions
import UIKit.UISceneDefinitions
import
UIKit.UISceneEnhancedStateRestoration
```

```
import UIKit.UISceneOptions
import UIKit.UISceneSession
import
UIKit.UISceneSessionActivationRequest
import
UIKit.UISceneSystemProtectionManager
import UIKit.UISceneWindowingBehaviors
import UIKit.UIScene_AVAudioSession
import UIKit.UIScreen
import
UIKit.UIScreenEdgePanGestureRecognizer
import UIKit.UIScreenMode
import UIKit.UIScreenshotService
import UIKit.UIScribbleInteraction
import UIKit.UIScrollView
import UIKit.UISearchBar
import
UIKit.UISearchContainerViewController
import UIKit.UISearchController
import UIKit.UISearchDisplayController
import UIKit.UISearchSuggestion
import UIKit.UISearchTab
import UIKit.UISearchTextField
import UIKit.UISegmentedControl
import UIKit.UISelectionFeedbackGenerator
import UIKit.UIShadowProperties
import UIKit.UIShape
import
UIKit.UISheetPresentationController
import UIKit.UISlider
import UIKit.UISnapBehavior
import UIKit.UISplitViewController
import UIKit.UISpringLoadedInteraction
```

```
import
UIKit.UISpringLoadedInteractionSupporting
import UIKit.UIStackView
import UIKit.UIStandardTextCursorView
import UIKit.UIStateRestoration
import UIKit.UIStatusBarManager
import UIKit.UIStepper
import UIKit.UIStoryboard
import UIKit.UIStoryboardPopoverSegue
import UIKit.UIStoryboardSegue
import UIKit.UIStringDrawing
import UIKit.UISwipeActionsConfiguration
import UIKit.UISwipeGestureRecognizer
import UIKit.UISwitch
import UIKit.UISymbolEffectCompletion
import UIKit.UITab
import UIKit.UITabBar
import UIKit.UITabBarAppearance
import UIKit.UITabBarController
import UIKit.UITabBarControllerSidebar
import UIKit.UITabBarItem
import UIKit.UITabGroup
import UIKit.UITabSidebarItem
import UIKit.UITableView
import UIKit.UITableViewCell
import UIKit.UITableViewController
import UIKit.UITableViewHeaderFooterView
import UIKit.UITapGestureRecognizer
import UIKit.UITargetedDragPreview
import UIKit.UITargetedPreview
import UIKit.UITextChecker
import
UIKit.UITextCursorDropPositionAnimator
```

```
import UIKit.UITextCursorView
import UIKit.UITextDragPreviewRenderer
import UIKit.UITextDragURLPreviews
import UIKit.UITextDragging
import UIKit.UITextDropProposal
import UIKit.UITextDropping
import UIKit.UITextField
import UIKit.UITextFormattingCoordinator
import
UIKit.UITextFormattingViewController
import
UIKit.UITextFormattingViewControllerChang
eValue
import
UIKit.UITextFormattingViewControllerCompo
nent
import
UIKit.UITextFormattingViewControllerConfi
guration
import
UIKit.UITextFormattingViewControllerForma
ttingDescriptor
import
UIKit.UITextFormattingViewControllerForma
ttingStyle
import UIKit.UITextInput
import UIKit.UITextInputContext
import UIKit.UITextInputTraits
import UIKit.UITextInteraction
import UIKit.UITextItem
import UIKit.UITextItemInteraction
import UIKit.UITextLoupeSession
import
```

```
UIKit.UITextPasteConfigurationSupporting
import UIKit.UITextPasteDelegate
import UIKit.UITextSearching
import
UIKit.UITextSelectionDisplayInteraction
import UIKit.UITextSelectionHandleView
import UIKit.UITextSelectionHighlightView
import UIKit.UITextView
import UIKit.UITimingCurveProvider
import UIKit.UITimingParameters
import UIKit.UIToolTipInteraction
import UIKit.UIToolbar
import UIKit.UIToolbarAppearance
import UIKit.UITouch
import UIKit.UITrackingLayoutGuide
import UIKit.UITrait
import UIKit.UITraitCollection
import UIKit.UITraitListEnvironment
import UIKit.UIUpdateActionPhase
import UIKit.UIUpdateInfo
import UIKit.UIUpdateLink
import UIKit.UIUserActivity
import UIKit.UIUserNotificationSettings
import UIKit.UIVibrancyEffect
import UIKit.UIVideoEditorController
import UIKit.UIView
import UIKit.UIViewAnimating
import UIKit.UIViewConfigurationState
import UIKit.UIViewController
import UIKit.UIViewControllerTransition
import
UIKit.UIViewControllerTransitionCoordinat
or
```

```
import
UIKit.UIViewControllerTransitioning
import UIKit.UIViewPropertyAnimator
import UIKit.UIVisualEffect
import UIKit.UIVisualEffectView
import UIKit.UIWebView
import UIKit.UIWindow
import UIKit.UIWindowScene
import
UIKit.UIWindowSceneActivationAction
import
UIKit.UIWindowSceneActivationConfiguratio
n
import
UIKit.UIWindowSceneActivationInteraction
import
UIKit.UIWindowSceneActivationRequestOptio
ns
import UIKit.UIWindowSceneDragInteraction
import UIKit.UIWindowSceneGeometry
import
UIKit.UIWindowSceneGeometryPreferences
import
UIKit.UIWindowSceneGeometryPreferencesIOS
import
UIKit.UIWindowSceneGeometryPreferencesMac
import
UIKit.UIWindowSceneGeometryPreferencesVis
ion
import UIKit.UIWindowScenePlacement
import
UIKit.UIWindowSceneProminentPlacement
import UIKit.UIWindowScenePushPlacement
```

```
import
UIKit.UIWindowSceneReplacePlacement
import
UIKit.UIWindowSceneStandardPlacement
import UIKit.UIWritingToolsCoordinator
import
UIKit.UIWritingToolsCoordinatorAnimationP
arameters
import
UIKit.UIWritingToolsCoordinatorContext
import UIKit.UIZoomTransitionOptions
import
UIKit.UNNotificationResponse_UIKitAdditio
ns
import _Concurrency
import _StringProcessing
import _SwiftConcurrencyShims

@available(iOS 14.0, tvOS 14.0, *)
@preconcurrency public struct
NSDiffableDataSourceSectionSnapshot<ItemI
dentifierType> : @unchecked Sendable
where ItemIdentifierType : Hashable,
ItemIdentifierType : Sendable {

    public init()

    public init(_ snapshot:
NSDiffableDataSourceSectionSnapshot<ItemI
dentifierType>)

    public mutating func append(_ items:
[ItemIdentifierType], to parent:
```

```
ItemIdentifierType? = nil)

    public mutating func insert(_ items:
[ItemIdentifierType], before item:
ItemIdentifierType)

    public mutating func insert(_ items:
[ItemIdentifierType], after item:
ItemIdentifierType)

    public mutating func delete(_ items:
[ItemIdentifierType])

    public mutating func deleteAll()

    public mutating func expand(_ items:
[ItemIdentifierType])

    public mutating func collapse(_
items: [ItemIdentifierType])

    public mutating func
replace(childrenOf parent:
ItemIdentifierType, using snapshot:
NSDiffableDataSourceSectionSnapshot<ItemI
dentifierType>)

    public mutating func insert(_
snapshot:
NSDiffableDataSourceSectionSnapshot<ItemI
dentifierType>, before item:
(ItemIdentifierType))
```

```
    public mutating func insert(_
snapshot:
NSDiffableDataSourceSectionSnapshot<ItemI
dentifierType>, after item:
(ItemIdentifierType))

    public func isExpanded(_ item:
ItemIdentifierType) -> Bool

    public func isVisible(_ item:
ItemIdentifierType) -> Bool

    public func contains(_ item:
ItemIdentifierType) -> Bool

    public func level(of item:
ItemIdentifierType) -> Int

    public func index(of item:
ItemIdentifierType) -> Int?

    public func parent(of child:
ItemIdentifierType) ->
ItemIdentifierType?

    public func snapshot(of parent:
ItemIdentifierType, includingParent: Bool
= false) ->
NSDiffableDataSourceSectionSnapshot<ItemI
dentifierType>

    public var items:
[ItemIdentifierType] { get }
```

```
    public var rootItems:
[ItemIdentifierType] { get }

    public var visibleItems:
[ItemIdentifierType] { get }

    @available(iOS 18.1, tvOS 18.1,
visionOS 2.1, *)
    public var expandedItems:
[ItemIdentifierType] { get }

    public func visualDescription() ->
String
}

@available(iOS 14.0, tvOS 14.0, *)
@preconcurrency public struct
NSDiffableDataSourceSectionTransaction<Se
ctionIdentifierType,
ItemIdentifierType> : @unchecked Sendable
where SectionIdentifierType : Hashable,
SectionIdentifierType : Sendable,
ItemIdentifierType : Hashable,
ItemIdentifierType : Sendable {

    public var sectionIdentifier:
SectionIdentifierType { get }

    public var initialSnapshot:
NSDiffableDataSourceSectionSnapshot<ItemI
dentifierType> { get }
```

```
    public var finalSnapshot:
NSDiffableDataSourceSectionSnapshot<ItemI
dentifierType> { get }

    public var difference:
CollectionDifference<ItemIdentifierType>
{ get }
}

@available(iOS 13.0, tvOS 13.0, *)
@preconcurrency public struct
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType> :
@unchecked Sendable where
SectionIdentifierType : Hashable,
SectionIdentifierType : Sendable,
ItemIdentifierType : Hashable,
ItemIdentifierType : Sendable {

    public init()

    public var numberOfItems: Int { get }

    public var numberOfSections: Int {
get }

    public var sectionIdentifiers:
[SectionIdentifierType] { get }

    public var itemIdentifiers:
[ItemIdentifierType] { get }

    @available(iOS 15.0, tvOS 15.0, *)
    public var
reloadedSectionIdentifiers:
[SectionIdentifierType] { get }

    @available(iOS 15.0, tvOS 15.0, *)
    public var reloadedItemIdentifiers:
[ItemIdentifierType] { get }

    @available(iOS 15.0, tvOS 15.0, *)
    public var
reconfiguredItemIdentifiers:
[ItemIdentifierType] { get }

    public func numberOfItems(inSection
identifier: SectionIdentifierType) -> Int

    public func itemIdentifiers(inSection
identifier: SectionIdentifierType) ->
[ItemIdentifierType]

    public func
sectionIdentifier(containingItem
identifier: ItemIdentifierType) ->
SectionIdentifierType?

    public func indexOfItem(_ identifier:
ItemIdentifierType) -> Int?

    public func indexOfSection(_
identifier: SectionIdentifierType) ->
Int?

    public mutating func appendItems(_
identifiers: [ItemIdentifierType],
toSection sectionIdentifier:
SectionIdentifierType? = nil)

    public mutating func insertItems(_
identifiers: [ItemIdentifierType],
beforeItem beforeIdentifier:
ItemIdentifierType)

    public mutating func insertItems(_
identifiers: [ItemIdentifierType],
afterItem afterIdentifier:
ItemIdentifierType)

    public mutating func deleteItems(_
identifiers: [ItemIdentifierType])

    public mutating func deleteAllItems()

    public mutating func moveItem(_
identifier: ItemIdentifierType,
beforeItem toIdentifier:
ItemIdentifierType)

    public mutating func moveItem(_
identifier: ItemIdentifierType, afterItem
toIdentifier: ItemIdentifierType)

    public mutating func reloadItems(_
identifiers: [ItemIdentifierType])

    @available(iOS 15.0, tvOS 15.0, *)
    public mutating func
```

```
reconfigureItems(_ identifiers:
[ItemIdentifierType])

    public mutating func appendSections(_
identifiers: [SectionIdentifierType])

    public mutating func insertSections(_
identifiers: [SectionIdentifierType],
beforeSection toIdentifier:
SectionIdentifierType)

    public mutating func insertSections(_
identifiers: [SectionIdentifierType],
afterSection toIdentifier:
SectionIdentifierType)

    public mutating func deleteSections(_
identifiers: [SectionIdentifierType])

    public mutating func moveSection(_
identifier: SectionIdentifierType,
beforeSection toIdentifier:
SectionIdentifierType)

    public mutating func moveSection(_
identifier: SectionIdentifierType,
afterSection toIdentifier:
SectionIdentifierType)

    public mutating func reloadSections(_
identifiers: [SectionIdentifierType])
}
```

```
@available(iOS 14.0, tvOS 14.0, *)
@preconcurrency public struct
NSDiffableDataSourceTransaction<SectionId
entifierType, ItemIdentifierType> :
@unchecked Sendable where
SectionIdentifierType : Hashable,
SectionIdentifierType : Sendable,
ItemIdentifierType : Hashable,
ItemIdentifierType : Sendable {

    public var initialSnapshot:
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType> { get }

    public var finalSnapshot:
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType> { get }

    public var difference:
CollectionDifference<ItemIdentifierType>
{ get }

    public var sectionTransactions:
[NSDiffableDataSourceSectionTransaction<S
ectionIdentifierType,
ItemIdentifierType>] { get }
}

@available(iOS 17.0, macOS 14.0, tvOS
17.0, *)
@freestanding(declaration) public macro
Preview(_ name: String? = nil, traits:
PreviewTrait<Preview.ViewTraits>...,
```

```swift
@PreviewMacroBodyBuilder<UIView> body:
@escaping @MainActor () -> UIView) =
#externalMacro(module: "PreviewsMacros",
type: "KitViewMacro")

@available(iOS 17.0, macOS 14.0, tvOS
17.0, *)
@freestanding(declaration) public macro
Preview(_ name: String? = nil, traits:
PreviewTrait<Preview.ViewTraits>...,
@PreviewMacroBodyBuilder<UIViewController
> body: @escaping @MainActor () ->
UIViewController) =
#externalMacro(module: "PreviewsMacros",
type: "KitViewMacro")

@available(swift, deprecated: 4.2,
message: "Use the overload of
UIApplicationMain where the type of the
second parameter is
UnsafeMutablePointer<UnsafeMutablePointer
<Int8>?>, which is the same as the type
of CommandLine.unsafeArgv.")
public func UIApplicationMain(_ argc:
Int32, _ argv:
UnsafeMutablePointer<UnsafeMutablePointer
<Int8>>!, _ principalClassName: String?,
_ delegateClassName: String?) -> Int32

@available(iOS 14.0, tvOS 14.0, *)
public struct UIBackgroundConfiguration :
Hashable {
    public static func clear() ->
UIBackgroundConfiguration

    @available(iOS 18.0, tvOS 18.0,
visionOS 2.0, *)
    public static func listCell() ->
UIBackgroundConfiguration

    @available(iOS 18.0, tvOS 18.0,
visionOS 2.0, *)
    public static func listHeader() ->
UIBackgroundConfiguration

    @available(iOS 18.0, tvOS 18.0,
visionOS 2.0, *)
    public static func listFooter() ->
UIBackgroundConfiguration

    public static func
listAccompaniedSidebarCell() ->
UIBackgroundConfiguration

    public func updated(for state: any
UIConfigurationState) ->
UIBackgroundConfiguration

    public var customView: UIView?

    public var cornerRadius: CGFloat

    public var backgroundInsets:
NSDirectionalEdgeInsets
```

```
    public var
edgesAddingLayoutMarginsToBackgroundInset
s: NSDirectionalRectEdge

    public var backgroundColor: UIColor?

    public var
backgroundColorTransformer:
UIConfigurationColorTransformer?

    public func
resolvedBackgroundColor(for tintColor:
UIColor) -> UIColor

    public var visualEffect:
UIVisualEffect?

    @available(iOS 15.0, tvOS 15.0, *)
    public var image: UIImage?

    @available(iOS 15.0, tvOS 15.0, *)
    public var imageContentMode:
UIView.ContentMode

    public var strokeColor: UIColor?

    public var strokeColorTransformer:
UIConfigurationColorTransformer?

    public func resolvedStrokeColor(for
tintColor: UIColor) -> UIColor

    public var strokeWidth: CGFloat
    public var strokeOutset: CGFloat

    @available(iOS 18.0, tvOS 18.0,
visionOS 2.0, *)
    public var shadowProperties:
UIShadowProperties

    /// Hashes the essential components
of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform
to the `Hashable` protocol. The
    /// components used for hashing must
be the same as the components compared
    /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your
implementation of `hash(into:)`,
    ///   don't call `finalize()` on the
`hasher` instance provided,
    ///   or replace it with a different
instance.
    ///   Doing so may become a compile-
time error in the future.
    ///
    /// - Parameter hasher: The hasher to
use when combining the components
    ///   of this instance.
    public func hash(into hasher: inout
Hasher)

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (a:
UIBackgroundConfiguration, b:
UIBackgroundConfiguration) -> Bool

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
```

```
    public var hashValue: Int { get }
}

@available(iOS 14.0, tvOS 14.0, *)
extension UIBackgroundConfiguration {

    @available(iOS, introduced: 14.0,
deprecated: 18.0, renamed: "listCell")
    @available(tvOS, introduced: 14.0,
deprecated: 18.0, renamed: "listCell")
    public static func listPlainCell() ->
UIBackgroundConfiguration

    @available(iOS, introduced: 14.0,
deprecated: 18.0, message: "Use the
generic listHeader() or listFooter()
background configuration")
    @available(tvOS, introduced: 14.0,
deprecated: 18.0, message: "Use the
generic listHeader() or listFooter()
background configuration")
    public static func
listPlainHeaderFooter() ->
UIBackgroundConfiguration

    @available(iOS, introduced: 14.0,
deprecated: 18.0, renamed: "listCell")
    @available(tvOS, introduced: 14.0,
deprecated: 18.0, renamed: "listCell")
    public static func listGroupedCell()
-> UIBackgroundConfiguration

    @available(iOS, introduced: 14.0,
deprecated: 18.0, message: "Use the
generic listHeader() or listFooter()
background configuration")
    @available(tvOS, introduced: 14.0,
deprecated: 18.0, message: "Use the
generic listHeader() or listFooter()
background configuration")
    public static func
listGroupedHeaderFooter() ->
UIBackgroundConfiguration

    @available(iOS, introduced: 14.0,
deprecated: 18.0, message: "Use the
generic listHeader() or listFooter()
background configuration")
    @available(tvOS, unavailable)
    public static func
listSidebarHeader() ->
UIBackgroundConfiguration

    @available(iOS, introduced: 14.0,
deprecated: 18.0, renamed: "listCell")
    public static func listSidebarCell()
-> UIBackgroundConfiguration
}

@available(iOS 14.0, tvOS 14.0, *)
extension UIBackgroundConfiguration :
CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {

    /// A textual representation of this
```

```
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }
```

```
    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
```

```
get }

    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}

@available(iOS 14.0, tvOS 14.0, *)
public struct UICellAccessory {

    public typealias ActionHandler = ()
-> Void

    public enum DisplayedState {

        case always

        case whenEditing

        case whenNotEditing

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UICellAccessory.DisplayedState, b:
UICellAccessory.DisplayedState) -> Bool

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
```

```
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    public enum LayoutDimension {

        case actual

        case standard
        case custom(CGFloat)
    }

    public struct
DisclosureIndicatorOptions {

        public var isHidden: Bool

        public var reservedLayoutWidth:
UICellAccessory.LayoutDimension

        public var tintColor: UIColor?

        public init(isHidden: Bool? =
nil, reservedLayoutWidth:
UICellAccessory.LayoutDimension? = nil,
tintColor: UIColor? = nil)
    }

    public static func
disclosureIndicator(displayed:
UICellAccessory.DisplayedState = .always,
options:
UICellAccessory.DisclosureIndicatorOption
s = DisclosureIndicatorOptions()) ->
UICellAccessory

    @available(iOS 15.4, tvOS 15.4, *)
    public struct DetailOptions {

        public var isHidden: Bool
        public var reservedLayoutWidth:
UICellAccessory.LayoutDimension

        public var tintColor: UIColor?

        public init(isHidden: Bool? =
nil, reservedLayoutWidth:
UICellAccessory.LayoutDimension? = nil,
tintColor: UIColor? = nil)
    }

    @available(iOS 15.4, tvOS 15.4, *)
    public static func detail(displayed:
UICellAccessory.DisplayedState = .always,
options: UICellAccessory.DetailOptions =
DetailOptions(), actionHandler:
UICellAccessory.ActionHandler? = nil) ->
UICellAccessory

    public struct CheckmarkOptions {

        public var isHidden: Bool

        public var reservedLayoutWidth:
UICellAccessory.LayoutDimension

        public var tintColor: UIColor?

        public init(isHidden: Bool? =
nil, reservedLayoutWidth:
UICellAccessory.LayoutDimension? = nil,
tintColor: UIColor? = nil)
    }
    public static func
checkmark(displayed:
UICellAccessory.DisplayedState = .always,
options: UICellAccessory.CheckmarkOptions
= CheckmarkOptions()) -> UICellAccessory

    public struct DeleteOptions {

        public var isHidden: Bool

        public var reservedLayoutWidth:
UICellAccessory.LayoutDimension

        public var tintColor: UIColor?

        public var backgroundColor:
UIColor?

        public init(isHidden: Bool? =
nil, reservedLayoutWidth:
UICellAccessory.LayoutDimension? = nil,
tintColor: UIColor? = nil,
backgroundColor: UIColor? = nil)
    }

    public static func delete(displayed:
UICellAccessory.DisplayedState
= .whenEditing, options:
UICellAccessory.DeleteOptions =
DeleteOptions(), actionHandler:
UICellAccessory.ActionHandler? = nil) ->
UICellAccessory
```

```
    public struct InsertOptions {

        public var isHidden: Bool

        public var reservedLayoutWidth:
UICellAccessory.LayoutDimension

        public var tintColor: UIColor?

        public var backgroundColor:
UIColor?

        public init(isHidden: Bool? =
nil, reservedLayoutWidth:
UICellAccessory.LayoutDimension? = nil,
tintColor: UIColor? = nil,
backgroundColor: UIColor? = nil)
    }

    public static func insert(displayed:
UICellAccessory.DisplayedState
= .whenEditing, options:
UICellAccessory.InsertOptions =
InsertOptions(), actionHandler:
UICellAccessory.ActionHandler? = nil) ->
UICellAccessory

    public struct ReorderOptions {

        public var isHidden: Bool

        public var reservedLayoutWidth:
```

```
UICellAccessory.LayoutDimension

        public var tintColor: UIColor?

        public var
showsVerticalSeparator: Bool

        public init(isHidden: Bool? =
nil, reservedLayoutWidth:
UICellAccessory.LayoutDimension? = nil,
tintColor: UIColor? = nil,
showsVerticalSeparator: Bool? = nil)
    }

    public static func reorder(displayed:
UICellAccessory.DisplayedState
= .whenEditing, options:
UICellAccessory.ReorderOptions =
ReorderOptions()) -> UICellAccessory

    public struct MultiselectOptions {

        public var isHidden: Bool

        public var reservedLayoutWidth:
UICellAccessory.LayoutDimension

        public var tintColor: UIColor?

        public var backgroundColor:
UIColor?

        public init(isHidden: Bool? =
nil, reservedLayoutWidth:
UICellAccessory.LayoutDimension? = nil,
tintColor: UIColor? = nil,
backgroundColor: UIColor? = nil)
    }

    public static func
multiselect(displayed:
UICellAccessory.DisplayedState
= .whenEditing, options:
UICellAccessory.MultiselectOptions =
MultiselectOptions()) -> UICellAccessory

    public struct
OutlineDisclosureOptions {

        public enum Style {

            case automatic

            case header

            case cell

            /// Returns a Boolean value
indicating whether two values are equal.
            ///
            /// Equality is the inverse
of inequality. For any values `a` and
`b`,
            /// `a == b` implies that
`a != b` is `false`.
            ///
            /// - Parameters:
            ///   - lhs: A value to
compare.
            ///   - rhs: Another value to
compare.
            public static func == (a:
UICellAccessory.OutlineDisclosureOptions.
Style, b:
UICellAccessory.OutlineDisclosureOptions.
Style) -> Bool

            /// Hashes the essential
components of this value by feeding them
into the
            /// given hasher.
            ///
            /// Implement this method to
conform to the `Hashable` protocol. The
            /// components used for
hashing must be the same as the
components compared
            /// in your type's `==`
operator implementation. Call
`hasher.combine(_:)`
            /// with each of these
components.
            ///
            /// - Important: In your
implementation of `hash(into:)`,
            ///   don't call `finalize()`
on the `hasher` instance provided,
            ///   or replace it with a
different instance.
```

```
            ///   Doing so may become a
compile-time error in the future.
            ///
            /// - Parameter hasher: The
hasher to use when combining the
components
            ///   of this instance.
            public func hash(into hasher:
inout Hasher)

            /// The hash value.
            ///
            /// Hash values are not
guaranteed to be equal across different
executions of
            /// your program. Do not save
hash values to use during a future
execution.
            ///
            /// - Important: `hashValue`
is deprecated as a `Hashable`
requirement. To
            ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
            ///   The compiler provides
an implementation for `hashValue` for
you.
            public var hashValue: Int {
get }
        }

        public var style:
```

```
UICellAccessory.OutlineDisclosureOptions.
Style

        public var isHidden: Bool

        public var reservedLayoutWidth:
UICellAccessory.LayoutDimension

        public var tintColor: UIColor?

        public init(style:
UICellAccessory.OutlineDisclosureOptions.
Style? = nil, isHidden: Bool? = nil,
reservedLayoutWidth:
UICellAccessory.LayoutDimension? = nil,
tintColor: UIColor? = nil)
    }

    public static func
outlineDisclosure(displayed:
UICellAccessory.DisplayedState = .always,
options:
UICellAccessory.OutlineDisclosureOptions
= OutlineDisclosureOptions(),
actionHandler:
UICellAccessory.ActionHandler? = nil) ->
UICellAccessory

    public typealias
MenuSelectedElementDidChangeHandler =
(UIMenu) -> Void

    @available(iOS 16.0, *)
    public struct PopUpMenuOptions {

        public var isHidden: Bool

        public var reservedLayoutWidth:
UICellAccessory.LayoutDimension

        public var tintColor: UIColor?

        public init(isHidden: Bool? =
nil, reservedLayoutWidth:
UICellAccessory.LayoutDimension? = nil,
tintColor: UIColor? = nil)
    }

    @available(iOS 16.0, *)
    public static func popUpMenu(_ menu:
UIMenu, displayed:
UICellAccessory.DisplayedState = .always,
options: UICellAccessory.PopUpMenuOptions
= PopUpMenuOptions(),
selectedElementDidChangeHandler:
UICellAccessory.MenuSelectedElementDidCha
ngeHandler? = nil) -> UICellAccessory

    public struct LabelOptions {

        public var isHidden: Bool

        public var reservedLayoutWidth:
UICellAccessory.LayoutDimension

        public var tintColor: UIColor?
        public var font: UIFont

        public var
adjustsFontForContentSizeCategory: Bool

        public init(isHidden: Bool? =
nil, reservedLayoutWidth:
UICellAccessory.LayoutDimension? = nil,
tintColor: UIColor? = nil, font: UIFont?
= nil, adjustsFontForContentSizeCategory:
Bool? = nil)
    }

    public static func label(text:
String, displayed:
UICellAccessory.DisplayedState = .always,
options: UICellAccessory.LabelOptions =
LabelOptions()) -> UICellAccessory

    public enum Placement {

        public typealias Position = (_
accessories: [UICellAccessory]) -> Int

        public static func
position(before accessory:
UICellAccessory) ->
UICellAccessory.Placement.Position

        public static func position(after
accessory: UICellAccessory) ->
UICellAccessory.Placement.Position
```

```
        case leading(displayed:
UICellAccessory.DisplayedState = .always,
at: UICellAccessory.Placement.Position =
{ $0.count })

        case trailing(displayed:
UICellAccessory.DisplayedState = .always,
at: UICellAccessory.Placement.Position =
{ _ in 0 })
    }

    public struct CustomViewConfiguration
{

        public let customView: UIView

        public let placement:
UICellAccessory.Placement

        public var isHidden: Bool

        public var reservedLayoutWidth:
UICellAccessory.LayoutDimension

        public var tintColor: UIColor?

        public var maintainsFixedSize:
Bool

        public init(customView: UIView,
placement: UICellAccessory.Placement,
isHidden: Bool? = nil,
reservedLayoutWidth:
UICellAccessory.LayoutDimension? = nil,
tintColor: UIColor? = nil,
maintainsFixedSize: Bool? = nil)
    }

    public static func
customView(configuration:
UICellAccessory.CustomViewConfiguration)
-> UICellAccessory

    public enum AccessoryType : Hashable
{

        case disclosureIndicator

        case outlineDisclosure

        case checkmark

        case delete

        case insert

        case reorder

        case multiselect

        case label

        case customView(UIView)

        @available(iOS 15.4, tvOS 15.4,
*)
        case detail

        @available(iOS 16.0, *)
        case popUpMenu

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (lhs:
UICellAccessory.AccessoryType, rhs:
UICellAccessory.AccessoryType) -> Bool

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
```

```
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    public let accessoryType:
UICellAccessory.AccessoryType
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UICellAccessory.DisplayedState :
Equatable {
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UICellAccessory.DisplayedState : Hashable
{
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UICellAccessory.OutlineDisclosureOptions.
Style : Equatable {
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UICellAccessory.OutlineDisclosureOptions.
Style : Hashable {
}

@available(iOS 14.0, tvOS 14.0, *)
public struct UICellConfigurationState :
UIConfigurationState, Hashable {

    @available(iOS 14.0, *)
    public enum DragState : Hashable {

        case none

        case lifting

        case dragging

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UICellConfigurationState.DragState, b:
UICellConfigurationState.DragState) ->
Bool

        /// Hashes the essential
```

```
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// The hash value.
        ///
        /// Hash values are not
```

```
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    @available(iOS 14.0, *)
    public enum DropState : Hashable {

        case none

        case notTargeted

        case targeted

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
```

```
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UICellConfigurationState.DropState, b:
UICellConfigurationState.DropState) ->
Bool

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
```

```
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    public var traitCollection:
UITraitCollection

    public var isDisabled: Bool
    public var isHighlighted: Bool

    public var isSelected: Bool

    public var isFocused: Bool

    @available(iOS 15.0, tvOS 15.0, *)
    public var isPinned: Bool

    public var isEditing: Bool

    public var isExpanded: Bool

    public var isSwiped: Bool

    public var isReordering: Bool

    public var cellDragState:
UICellConfigurationState.DragState

    public var cellDropState:
UICellConfigurationState.DropState

    public subscript(key:
UIConfigurationStateCustomKey) ->
AnyHashable?

    public init(traitCollection:
UITraitCollection)

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (lhs:
UICellConfigurationState, rhs:
UICellConfigurationState) -> Bool

    /// Hashes the essential components
of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform
to the `Hashable` protocol. The
    /// components used for hashing must
be the same as the components compared
    /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your
implementation of `hash(into:)`,
    ///   don't call `finalize()` on the
`hasher` instance provided,
    ///   or replace it with a different
instance.
    ///   Doing so may become a compile-
time error in the future.
```

```
    ///
    /// - Parameter hasher: The hasher to
use when combining the components
    ///   of this instance.
    public func hash(into hasher: inout
Hasher)

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }
}

@available(iOS 14.0, tvOS 14.0, *)
extension UICellConfigurationState :
CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {

    /// A textual representation of this
instance.
```

```
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }

    /// A textual representation of this
```

```
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
get }
```

```
    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}

@available(iOS 14.0, tvOS 14.0, *)
public struct
UICollectionLayoutListConfiguration {

    public enum Appearance {

        case plain

        case grouped

        @available(tvOS, unavailable)
        case insetGrouped

        @available(tvOS, unavailable)
        case sidebar

        @available(tvOS, unavailable)
        case sidebarPlain

        /// Returns a Boolean value
indicating whether two values are equal.
```

```
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UICollectionLayoutListConfiguration.Appea
rance, b:
UICollectionLayoutListConfiguration.Appea
rance) -> Bool

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }
    public enum HeaderMode {

        case none

        case supplementary

        case firstItemInSection

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UICollectionLayoutListConfiguration.Heade
rMode, b:
UICollectionLayoutListConfiguration.Heade
rMode) -> Bool

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
```

```
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
```

```
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    public enum FooterMode {

        case none

        case supplementary

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UICollectionLayoutListConfiguration.Foote
rMode, b:
```

```
UICollectionLayoutListConfiguration.Foote
rMode) -> Bool

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)
```

```
        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    @available(tvOS, unavailable)
    public typealias
SwipeActionsConfigurationProvider = (_
indexPath: IndexPath) ->
UISwipeActionsConfiguration?

    @available(iOS 14.5, *)
    @available(tvOS, unavailable)
    public typealias ItemSeparatorHandler
= (_ itemIndexPath: IndexPath, _
sectionSeparatorConfiguration:
UIListSeparatorConfiguration) ->
```

```
UIListSeparatorConfiguration

    public init(appearance:
UICollectionLayoutListConfiguration.Appea
rance)

    public var appearance:
UICollectionLayoutListConfiguration.Appea
rance { get }

    @available(tvOS, unavailable)
    public var showsSeparators: Bool

    @available(iOS 14.5, *)
    @available(tvOS, unavailable)
    public var separatorConfiguration:
UIListSeparatorConfiguration

    @available(iOS 14.5, *)
    @available(tvOS, unavailable)
    public var itemSeparatorHandler:
UICollectionLayoutListConfiguration.ItemS
eparatorHandler?

    public var backgroundColor: UIColor?

    @available(tvOS, unavailable)
    public var
leadingSwipeActionsConfigurationProvider:
UICollectionLayoutListConfiguration.Swipe
ActionsConfigurationProvider?

    @available(tvOS, unavailable)
    public var
trailingSwipeActionsConfigurationProvider
:
UICollectionLayoutListConfiguration.Swipe
ActionsConfigurationProvider?

    public var headerMode:
UICollectionLayoutListConfiguration.Heade
rMode

    public var footerMode:
UICollectionLayoutListConfiguration.Foote
rMode

    @available(iOS 15.0, tvOS 15.0, *)
    public var headerTopPadding: CGFloat?
}

@available(iOS 18.0, tvOS 18.0, visionOS
2.0, *)
extension
UICollectionLayoutListConfiguration {

    public struct
ContentHuggingElements : OptionSet {

        /// The corresponding value of
the raw type.
        ///
        /// A new instance initialized
with `rawValue` will be equivalent to
this
        /// instance. For example:
```

```
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter,
Legal
        ///     }
        ///
        ///     let selectedSize =
PaperSize.Letter
        ///
print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize ==
PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public let rawValue: Int

        /// Creates a new option set from
the given raw value.
        ///
        /// This initializer always
succeeds, even if the value passed as
`rawValue`
        /// exceeds the static properties
declared as part of the option set. This
        /// example creates an instance
of `ShippingOptions` with a raw value
beyond
        /// the highest element, with a
bit mask that effectively contains all
the
        /// declared static members.
        ///
        ///     let extraOptions =
ShippingOptions(rawValue: 255)
        ///
print(extraOptions.isStrictSuperset(of: .
all))
        ///     // Prints "true"
        ///
        /// - Parameter rawValue: The raw
value of the option set to create. Each
bit
        ///   of `rawValue` potentially
represents an element of the option set,
        ///   though raw values may
include bits that are not defined as
distinct
        ///   values of the `OptionSet`
type.
        public init(rawValue: Int)

        public static let
supplementaryHeader:
UICollectionLayoutListConfiguration.Conte
ntHuggingElements

        /// The type of the elements of
an array literal.
        @available(iOS 18.0, tvOS 18.0,
visionOS 2.0, *)
        public typealias
ArrayLiteralElement =
UICollectionLayoutListConfiguration.Conte
ntHuggingElements
```

```
        /// The element type of the
option set.
        ///
        /// To inherit all the default
implementations from the `OptionSet`
protocol,
        /// the `Element` type must be
`Self`, the default.
        @available(iOS 18.0, tvOS 18.0,
visionOS 2.0, *)
        public typealias Element =
UICollectionLayoutListConfiguration.Conte
ntHuggingElements

        /// The raw type that can be used
to represent all values of the conforming
        /// type.
        ///
        /// Every distinct value of the
conforming type has a corresponding
unique
        /// value of the `RawValue` type,
but there may be values of the `RawValue`
        /// type that don't have a
corresponding value of the conforming
type.
        @available(iOS 18.0, tvOS 18.0,
visionOS 2.0, *)
        public typealias RawValue = Int
    }

    public var contentHuggingElements:
```

```
UICollectionLayoutListConfiguration.Conte
ntHuggingElements
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UICollectionLayoutListConfiguration.Appea
rance : Equatable {
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UICollectionLayoutListConfiguration.Appea
rance : Hashable {
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UICollectionLayoutListConfiguration.Heade
rMode : Equatable {
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UICollectionLayoutListConfiguration.Heade
rMode : Hashable {
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UICollectionLayoutListConfiguration.Foote
rMode : Equatable {
}
```

```
@available(iOS 14.0, tvOS 14.0, *)
extension
UICollectionLayoutListConfiguration.Foote
rMode : Hashable {
}

@available(iOS 13.0, tvOS 13.0, *)
@MainActor @preconcurrency open class
UICollectionViewDiffableDataSource<Sectio
nIdentifierType, ItemIdentifierType> :
NSObject, UICollectionViewDataSource
where SectionIdentifierType : Hashable,
SectionIdentifierType : Sendable,
ItemIdentifierType : Hashable,
ItemIdentifierType : Sendable {

    public typealias CellProvider = (_
collectionView: UICollectionView, _
indexPath: IndexPath, _ itemIdentifier:
ItemIdentifierType) ->
UICollectionViewCell?

    public typealias
SupplementaryViewProvider = (_
collectionView: UICollectionView, _
elementKind: String, _ indexPath:
IndexPath) -> UICollectionReusableView?

    @MainActor @preconcurrency public var
supplementaryViewProvider:
UICollectionViewDiffableDataSource<Sectio
nIdentifierType,
ItemIdentifierType>.SupplementaryViewProv
ider?

    @MainActor @preconcurrency public
init(collectionView: UICollectionView,
cellProvider: @escaping
UICollectionViewDiffableDataSource<Sectio
nIdentifierType,
ItemIdentifierType>.CellProvider)

    @MainActor @preconcurrency open func
apply(_ snapshot:
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType>,
animatingDifferences: Bool = true,
completion: (() -> Void)? = nil)

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency open func
apply(_ snapshot:
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType>,
animatingDifferences: Bool = true) async

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency open func
applySnapshotUsingReloadData(_ snapshot:
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType>,
completion: (() -> Void)? = nil)

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency open func
```

```
applySnapshotUsingReloadData(_ snapshot:
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType>) async

    @MainActor @preconcurrency open func
snapshot() ->
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType>

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency open func
sectionIdentifier(for index: Int) ->
SectionIdentifierType?

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency open func
index(for sectionIdentifier:
SectionIdentifierType) -> Int?

    @MainActor @preconcurrency open func
itemIdentifier(for indexPath: IndexPath)
-> ItemIdentifierType?

    @MainActor @preconcurrency open func
indexPath(for itemIdentifier:
ItemIdentifierType) -> IndexPath?

    @MainActor @preconcurrency open func
numberOfSections(in collectionView:
UICollectionView) -> Int

    @MainActor @preconcurrency open func
collectionView(_ collectionView:
```

```
UICollectionView, numberOfItemsInSection
section: Int) -> Int

    @MainActor @preconcurrency open func
collectionView(_ collectionView:
UICollectionView, cellForItemAt
indexPath: IndexPath) ->
UICollectionViewCell

    @MainActor @preconcurrency open func
collectionView(_ collectionView:
UICollectionView,
viewForSupplementaryElementOfKind kind:
String, at indexPath: IndexPath) ->
UICollectionReusableView

    @MainActor @preconcurrency open func
collectionView(_ collectionView:
UICollectionView, canMoveItemAt
indexPath: IndexPath) -> Bool

    @MainActor @preconcurrency open func
collectionView(_ collectionView:
UICollectionView, moveItemAt
sourceIndexPath: IndexPath, to
destinationIndexPath: IndexPath)

    @MainActor @preconcurrency open func
indexTitles(for collectionView:
UICollectionView) -> [String]?

    @MainActor @preconcurrency open func
collectionView(_ collectionView:
```

```
UICollectionView, indexPathForIndexTitle
title: String, at index: Int) ->
IndexPath

    @MainActor @preconcurrency public
func description() -> String
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UICollectionViewDiffableDataSource {

    @preconcurrency public struct
SectionSnapshotHandlers<ItemIdentifierTyp
e> where ItemIdentifierType : Hashable,
ItemIdentifierType : Sendable {

        public var shouldExpandItem:
((ItemIdentifierType) -> Bool)?

        public var willExpandItem:
((ItemIdentifierType) -> Void)?

        public var shouldCollapseItem:
((ItemIdentifierType) -> Bool)?

        public var willCollapseItem:
((ItemIdentifierType) -> Void)?

        public var
snapshotForExpandingParent:
((ItemIdentifierType,
NSDiffableDataSourceSectionSnapshot<ItemI
dentifierType>) ->
NSDiffableDataSourceSectionSnapshot<ItemI
dentifierType>)?

        public init()
    }

    @MainActor @preconcurrency public var
sectionSnapshotHandlers:
UICollectionViewDiffableDataSource<Sectio
nIdentifierType,
ItemIdentifierType>.SectionSnapshotHandle
rs<ItemIdentifierType>
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UICollectionViewDiffableDataSource {

    @MainActor @preconcurrency public
func apply(_ snapshot:
NSDiffableDataSourceSectionSnapshot<ItemI
dentifierType>, to section:
SectionIdentifierType,
animatingDifferences: Bool = true,
completion: (() -> Void)? = nil)

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency public
func apply(_ snapshot:
NSDiffableDataSourceSectionSnapshot<ItemI
dentifierType>, to section:
SectionIdentifierType,
animatingDifferences: Bool = true) async

    @MainActor @preconcurrency public
func snapshot(for section:
SectionIdentifierType) ->
NSDiffableDataSourceSectionSnapshot<ItemI
dentifierType>
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UICollectionViewDiffableDataSource {

    public struct ReorderingHandlers {

        public var canReorderItem:
((ItemIdentifierType) -> Bool)?

        public var willReorder:
((NSDiffableDataSourceTransaction<Section
IdentifierType, ItemIdentifierType>) ->
Void)?

        public var didReorder:
((NSDiffableDataSourceTransaction<Section
IdentifierType, ItemIdentifierType>) ->
Void)?

        public init()
    }

    @MainActor @preconcurrency public var
reorderingHandlers:
```

```
UICollectionViewDiffableDataSource<Sectio
nIdentifierType,
ItemIdentifierType>.ReorderingHandlers
}

@available(iOS 13.0, tvOS 13.0, *)
extension
UICollectionViewDiffableDataSource :
Sendable {
}

@available(iOS 14.0, tvOS 14.0, *)
public struct
UIConfigurationColorTransformer {

    public let transform: (UIColor) ->
UIColor

    public init(_ transform: @escaping
(UIColor) -> UIColor)

    public func callAsFunction(_ input:
UIColor) -> UIColor

    public static let grayscale:
UIConfigurationColorTransformer

    public static let preferredTint:
UIConfigurationColorTransformer

    public static let monochromeTint:
UIConfigurationColorTransformer
}
```

```
@available(iOS 14.0, tvOS 14.0, *)
public protocol UIConfigurationState {

    init(traitCollection:
UITraitCollection)

    var traitCollection:
UITraitCollection { get set }

    subscript(key:
UIConfigurationStateCustomKey) ->
AnyHashable? { get set }
}

@available(iOS 15.0, tvOS 15.0, *)
public struct
UIConfigurationTextAttributesTransformer
{

    public let transform:
(AttributeContainer) ->
AttributeContainer

    public init(_ transform: @escaping
(AttributeContainer) ->
AttributeContainer)

    public func callAsFunction(_ input:
AttributeContainer) -> AttributeContainer
}

@available(iOS 14.0, tvOS 14.0, *)
```

```
public protocol UIContentConfiguration {

    @MainActor func makeContentView() ->
any UIView & UIContentView

    func updated(for state: any
UIConfigurationState) -> Self
}

@available(iOS 17.0, tvOS 17.0, *)
public struct
UIContentUnavailableConfiguration :
UIContentConfiguration, Hashable {

    public struct ImageProperties :
Hashable {

        public var
preferredSymbolConfiguration:
UIImage.SymbolConfiguration?

        public var tintColor: UIColor?

        public var cornerRadius: CGFloat

        public var maximumSize: CGSize

        public var
accessibilityIgnoresInvertColors: Bool

        /// Hashes the essential
components of this value by feeding them
into the
```

```
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UIContentUnavailableConfiguration.ImagePr
operties, b:
UIContentUnavailableConfiguration.ImagePr
operties) -> Bool

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }
    public struct TextProperties :
Hashable {

        public var font: UIFont

        public var color: UIColor

        public var lineBreakMode:
NSLineBreakMode

        public var numberOfLines: Int

        public var
adjustsFontSizeToFitWidth: Bool

        public var minimumScaleFactor:
CGFloat

        public var
allowsDefaultTighteningForTruncation:
Bool

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
```

```
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
```

```
UIContentUnavailableConfiguration.TextPro
perties, b:
UIContentUnavailableConfiguration.TextPro
perties) -> Bool

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    public struct ButtonProperties :
Hashable {

        public var primaryAction:
UIAction?

        public var menu: UIMenu?
        public var isEnabled: Bool

        public var role: UIButton.Role

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
```

```
inout Hasher)

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UIContentUnavailableConfiguration.ButtonP
roperties, b:
UIContentUnavailableConfiguration.ButtonP
roperties) -> Bool

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    public enum Alignment : Hashable {

        case natural

        case center

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UIContentUnavailableConfiguration.Alignme
nt, b:
UIContentUnavailableConfiguration.Alignme
nt) -> Bool

        /// Hashes the essential
components of this value by feeding them
```

```
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
```

```
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    public static func empty() ->
UIContentUnavailableConfiguration

    public static func loading() ->
UIContentUnavailableConfiguration

    public static func search() ->
UIContentUnavailableConfiguration

    @MainActor public func
makeContentView() -> any UIView &
UIContentView

    public func updated(for state: any
UIConfigurationState) ->
UIContentUnavailableConfiguration
```

```
    public var image: UIImage?

    public var imageProperties:
UIContentUnavailableConfiguration.ImagePr
operties

    public var text: String?

    public var attributedText:
NSAttributedString?

    public var textProperties:
UIContentUnavailableConfiguration.TextPro
perties

    public var secondaryText: String?

    public var secondaryAttributedText:
NSAttributedString?

    public var secondaryTextProperties:
UIContentUnavailableConfiguration.TextPro
perties

    public var button:
UIButton.Configuration

    public var buttonProperties:
UIContentUnavailableConfiguration.ButtonP
roperties

    public var secondaryButton:
UIButton.Configuration
```

```
    public var secondaryButtonProperties:
UIContentUnavailableConfiguration.ButtonP
roperties

    public var alignment:
UIContentUnavailableConfiguration.Alignme
nt

    public var
axesPreservingSuperviewLayoutMargins:
UIAxis

    public var directionalLayoutMargins:
NSDirectionalEdgeInsets

    public var imageToTextPadding:
CGFloat

    public var
textToSecondaryTextPadding: CGFloat

    public var textToButtonPadding:
CGFloat

    public var
buttonToSecondaryButtonPadding: CGFloat

    public var background:
UIBackgroundConfiguration

    /// Hashes the essential components
of this value by feeding them into the
```

```
    /// given hasher.
    ///
    /// Implement this method to conform
to the `Hashable` protocol. The
    /// components used for hashing must
be the same as the components compared
    /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your
implementation of `hash(into:)`,
    ///   don't call `finalize()` on the
`hasher` instance provided,
    ///   or replace it with a different
instance.
    ///   Doing so may become a compile-
time error in the future.
    ///
    /// - Parameter hasher: The hasher to
use when combining the components
    ///   of this instance.
    public func hash(into hasher: inout
Hasher)

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (a:
UIContentUnavailableConfiguration, b:
UIContentUnavailableConfiguration) ->
Bool

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }
}

@available(iOS 17.0, tvOS 17.0, *)
extension
UIContentUnavailableConfiguration :
CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {
    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
```

```
get }

    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
```

```
property.
    public var debugDescription: String {
get }

    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}

@available(iOS 17.0, tvOS 17.0, *)
extension
UIContentUnavailableConfiguration.ImagePr
operties : CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {

    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
```

```
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }

    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
```

```
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
get }

    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}

@available(iOS 17.0, tvOS 17.0, *)
extension
UIContentUnavailableConfiguration.TextPro
perties : CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {

    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }

    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
get }

    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}

@available(iOS 17.0, tvOS 17.0, *)
extension
UIContentUnavailableConfiguration.ButtonP
roperties : CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {

    /// A textual representation of this
instance.
```

```
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }

    /// A textual representation of this
```

```
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
get }
```

```
    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}

@available(iOS 17.0, tvOS 17.0, *)
public struct
UIContentUnavailableConfigurationState :
UIConfigurationState, Hashable {

    public var traitCollection:
UITraitCollection

    public var searchText: String?

    public subscript(key:
UIConfigurationStateCustomKey) ->
AnyHashable?

    public init(traitCollection:
UITraitCollection)

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
```

```
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (lhs:
UIContentUnavailableConfigurationState,
rhs:
UIContentUnavailableConfigurationState)
-> Bool

    /// Hashes the essential components
of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform
to the `Hashable` protocol. The
    /// components used for hashing must
be the same as the components compared
    /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your
implementation of `hash(into:)`,
    ///   don't call `finalize()` on the
`hasher` instance provided,
    ///   or replace it with a different
instance.
    ///   Doing so may become a compile-
```

```
time error in the future.
    ///
    /// - Parameter hasher: The hasher to
use when combining the components
    ///   of this instance.
    public func hash(into hasher: inout
Hasher)

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }
}

@available(iOS 17.0, tvOS 17.0, *)
extension
UIContentUnavailableConfigurationState :
CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {
    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }
```

```
    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
```

```
    public var debugDescription: String {
get }

    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}

@available(iOS 14.0, tvOS 14.0, *)
@MainActor public protocol
UIContentView : NSObjectProtocol {

    @MainActor var configuration: any
UIContentConfiguration { get set }

    @available(iOS 16.0, tvOS 16.0, *)
    @MainActor func supports(_
configuration: any
UIContentConfiguration) -> Bool
}

@available(iOS 16.0, tvOS 16.0, *)
extension UIContentView {

    @MainActor public func supports(_
configuration: any
UIContentConfiguration) -> Bool
```

```
}

@available(swift, introduced: 1.0,
deprecated: 4.2, message: "Use ==
operator instead.")
@available(iOS, introduced: 7.0,
deprecated: 12.0, message: "Use ==
operator instead.")
@available(watchOS, introduced: 2.0,
deprecated: 5.0, message: "Use ==
operator instead.")
public func
UIEdgeInsetsEqualToEdgeInsets(_ insets1:
UIEdgeInsets, _ insets2: UIEdgeInsets) ->
Bool

@available(swift, introduced: 1.0,
deprecated: 4.2, message: "Use ==
operator instead.")
@available(iOS, introduced: 7.0,
deprecated: 12.0, message: "Use ==
operator instead.")
public func UIFloatRangeIsEqualToRange(_
range: UIFloatRange, _ otherRange:
UIFloatRange) -> Bool

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
public struct UIHoverAutomaticEffect :
UIHoverEffect {

    public init()
```

```
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
public protocol UIHoverEffect {
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
extension UIHoverEffect where Self ==
UIHoverHighlightEffect {

    public static var highlight:
UIHoverHighlightEffect { get }
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
extension UIHoverEffect where Self ==
UIHoverLiftEffect {

    public static var lift:
UIHoverLiftEffect { get }
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
extension UIHoverEffect where Self ==
UIHoverAutomaticEffect {
    public static var automatic:
UIHoverAutomaticEffect { get }
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
public struct UIHoverHighlightEffect :
UIHoverEffect {

    public init()
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
public struct UIHoverLiftEffect :
UIHoverEffect {

    public init()
}

@available(iOS 17.0, tvOS 17.0, watchOS
10.0, *)
public struct UIImageReader {

    public struct Configuration :
Equatable {

        public var
prefersHighDynamicRange: Bool
```

```
        public var
preparesImagesForDisplay: Bool

        public var
preferredThumbnailSize: CGSize

        public var pixelsPerInch: Double

        public init()

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UIImageReader.Configuration, b:
UIImageReader.Configuration) -> Bool
    }

    public static let `default`:
UIImageReader

    public init(configuration:
UIImageReader.Configuration)
```

```
    public var configuration:
UIImageReader.Configuration { get }

    public func image(contentsOf fileURL:
URL) -> UIImage?

    public func image(data: Data) ->
UIImage?

    public func image(contentsOf fileURL:
URL) async -> UIImage?

    public func image(data: Data) async
-> UIImage?
}

@available(iOS 14.0, *)
@MainActor @preconcurrency open class
UIIndirectScribbleInteraction<Delegate> :
NSObject, UIInteraction where Delegate :
UIIndirectScribbleInteractionDelegate {

    @MainActor @preconcurrency weak
public var view: UIView? { get }

    @MainActor @preconcurrency weak
public var delegate: Delegate? { get }

    @MainActor @preconcurrency public
func willMove(to view: UIView?)

    @MainActor @preconcurrency public
func didMove(to view: UIView?)
```

```
    @MainActor @preconcurrency public
init(delegate: Delegate)

    @MainActor @preconcurrency public var
isHandlingWriting: Bool { get }
}

@available(iOS 14.0, *)
extension UIIndirectScribbleInteraction :
Sendable {
}

@available(iOS 14.0, *)
public protocol
UIIndirectScribbleInteractionDelegate :
NSObjectProtocol {

    associatedtype ElementIdentifier :
Hashable = String

    func indirectScribbleInteraction(_
interaction: any UIInteraction,
requestElementsIn rect: CGRect,
completion: @escaping
([Self.ElementIdentifier]) -> Void)

    @available(iOS 15.0, *)
    @MainActor func
indirectScribbleInteraction(_
interaction: any UIInteraction,
requestElementsIn rect: CGRect) async ->
[Self.ElementIdentifier]
```

```
    func indirectScribbleInteraction(_
interaction: any UIInteraction,
isElementFocused elementIdentifier:
Self.ElementIdentifier) -> Bool

    func indirectScribbleInteraction(_
interaction: any UIInteraction,
frameForElement elementIdentifier:
Self.ElementIdentifier) -> CGRect

    func indirectScribbleInteraction(_
interaction: any UIInteraction,
focusElementIfNeeded elementIdentifier:
Self.ElementIdentifier, referencePoint
focusReferencePoint: CGPoint, completion:
@escaping ((any UIResponder &
UITextInput)?) -> Void)

    @available(iOS 15.0, *)
    @MainActor func
indirectScribbleInteraction(_
interaction: any UIInteraction,
focusElementIfNeeded elementIdentifier:
Self.ElementIdentifier, referencePoint
focusReferencePoint: CGPoint) async ->
(any UIResponder & UITextInput)?

    func indirectScribbleInteraction(_
interaction: any UIInteraction,
shouldDelayFocusForElement
elementIdentifier:
Self.ElementIdentifier) -> Bool
```

```
    func indirectScribbleInteraction(_
interaction: any UIInteraction,
willBeginWritingInElement
elementIdentifier:
Self.ElementIdentifier)

    func indirectScribbleInteraction(_
interaction: any UIInteraction,
didFinishWritingInElement
elementIdentifier:
Self.ElementIdentifier)
}

@available(iOS 14.0, *)
extension
UIIndirectScribbleInteractionDelegate {

    public func
indirectScribbleInteraction(_
interaction: any UIInteraction,
willBeginWritingInElement
elementIdentifier:
Self.ElementIdentifier)

    public func
indirectScribbleInteraction(_
interaction: any UIInteraction,
didFinishWritingInElement
elementIdentifier:
Self.ElementIdentifier)

    public func
```

```
indirectScribbleInteraction(_
interaction: any UIInteraction,
shouldDelayFocusForElement
elementIdentifier:
Self.ElementIdentifier) -> Bool
}

@available(iOS 15.0, *)
extension
UIIndirectScribbleInteractionDelegate {

    @MainActor public func
indirectScribbleInteraction(_
interaction: any UIInteraction,
requestElementsIn rect: CGRect) async ->
[Self.ElementIdentifier]

    @MainActor public func
indirectScribbleInteraction(_
interaction: any UIInteraction,
focusElementIfNeeded elementIdentifier:
Self.ElementIdentifier, referencePoint
focusReferencePoint: CGPoint) async ->
(any UIResponder & UITextInput)?
}

@available(iOS 14.0, tvOS 14.0, *)
public struct
UIListContentConfiguration :
UIContentConfiguration, Hashable {

    public struct ImageProperties :
Hashable {
        public var
preferredSymbolConfiguration:
UIImage.SymbolConfiguration?

        public var tintColor: UIColor?

        public var tintColorTransformer:
UIConfigurationColorTransformer?

        public func resolvedTintColor(for
tintColor: UIColor) -> UIColor

        public var cornerRadius: CGFloat

        public var maximumSize: CGSize

        public var reservedLayoutSize:
CGSize

        public var
accessibilityIgnoresInvertColors: Bool

        @available(iOS 18.0, tvOS 18.0,
*)
        public var strokeColor: UIColor?

        @available(iOS 18.0, tvOS 18.0,
*)
        public var
strokeColorTransformer:
UIConfigurationColorTransformer?
```

```
        @available(iOS 18.0, tvOS 18.0,
*)
        public func
resolvedStrokeColor(for tintColor:
UIColor) -> UIColor

        @available(iOS 18.0, tvOS 18.0,
*)
        public var strokeWidth: CGFloat

        public static let
standardDimension: CGFloat

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
```

```
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UIListContentConfiguration.ImagePropertie
s, b:
UIListContentConfiguration.ImagePropertie
s) -> Bool

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
```

```
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    public struct TextProperties :
Hashable {

        public enum TextAlignment :
Hashable {

            case natural

            case center

            case justified

            /// Returns a Boolean value
indicating whether two values are equal.
            ///
            /// Equality is the inverse
of inequality. For any values `a` and
```

```
`b`,
            /// `a == b` implies that
`a != b` is `false`.
            ///
            /// - Parameters:
            ///   - lhs: A value to
compare.
            ///   - rhs: Another value to
compare.
            public static func == (a:
UIListContentConfiguration.TextProperties
.TextAlignment, b:
UIListContentConfiguration.TextProperties
.TextAlignment) -> Bool

            /// Hashes the essential
components of this value by feeding them
into the
            /// given hasher.
            ///
            /// Implement this method to
conform to the `Hashable` protocol. The
            /// components used for
hashing must be the same as the
components compared
            /// in your type's `==`
operator implementation. Call
`hasher.combine(_:)`
            /// with each of these
components.
            ///
            /// - Important: In your
implementation of `hash(into:)`,
            ///   don't call `finalize()`
on the `hasher` instance provided,
            ///   or replace it with a
different instance.
            ///   Doing so may become a
compile-time error in the future.
            ///
            /// - Parameter hasher: The
hasher to use when combining the
components
            ///   of this instance.
            public func hash(into hasher:
inout Hasher)

            /// The hash value.
            ///
            /// Hash values are not
guaranteed to be equal across different
executions of
            /// your program. Do not save
hash values to use during a future
execution.
            ///
            /// - Important: `hashValue`
is deprecated as a `Hashable`
requirement. To
            ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
            ///   The compiler provides
an implementation for `hashValue` for
you.
            public var hashValue: Int {
```

```
get }
        }

        public enum TextTransform :
Hashable {

            case none

            case uppercase

            case lowercase

            case capitalized

            /// Returns a Boolean value
indicating whether two values are equal.
            ///
            /// Equality is the inverse
of inequality. For any values `a` and
`b`,
            /// `a == b` implies that
`a != b` is `false`.
            ///
            /// - Parameters:
            ///   - lhs: A value to
compare.
            ///   - rhs: Another value to
compare.
            public static func == (a:
UIListContentConfiguration.TextProperties
.TextTransform, b:
UIListContentConfiguration.TextProperties
.TextTransform) -> Bool
```

```
            /// Hashes the essential
components of this value by feeding them
into the
            /// given hasher.
            ///
            /// Implement this method to
conform to the `Hashable` protocol. The
            /// components used for
hashing must be the same as the
components compared
            /// in your type's `==`
operator implementation. Call
`hasher.combine(_:)`
            /// with each of these
components.
            ///
            /// - Important: In your
implementation of `hash(into:)`,
            ///   don't call `finalize()`
on the `hasher` instance provided,
            ///   or replace it with a
different instance.
            ///   Doing so may become a
compile-time error in the future.
            ///
            /// - Parameter hasher: The
hasher to use when combining the
components
            ///   of this instance.
            public func hash(into hasher:
inout Hasher)
```

```
            /// The hash value.
            ///
            /// Hash values are not
guaranteed to be equal across different
executions of
            /// your program. Do not save
hash values to use during a future
execution.
            ///
            /// - Important: `hashValue`
is deprecated as a `Hashable`
requirement. To
            ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
            ///   The compiler provides
an implementation for `hashValue` for
you.
            public var hashValue: Int {
get }
        }

        public var font: UIFont

        public var color: UIColor

        public var colorTransformer:
UIConfigurationColorTransformer?

        public func resolvedColor() ->
UIColor

        public var alignment:
```

```
UIListContentConfiguration.TextProperties
.TextAlignment

        public var lineBreakMode:
NSLineBreakMode

        public var numberOfLines: Int

        public var
adjustsFontSizeToFitWidth: Bool

        public var minimumScaleFactor:
CGFloat

        public var
allowsDefaultTighteningForTruncation:
Bool

        public var
adjustsFontForContentSizeCategory: Bool

        @available(macCatalyst 16.0, *)
        public var
showsExpansionTextWhenTruncated: Bool

        public var transform:
UIListContentConfiguration.TextProperties
.TextTransform

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UIListContentConfiguration.TextProperties
, b:
UIListContentConfiguration.TextProperties
) -> Bool

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    public static func cell() ->
```

```
UIListContentConfiguration

    public static func subtitleCell() ->
UIListContentConfiguration

    public static func valueCell() ->
UIListContentConfiguration

    @available(iOS 18.0, tvOS 18.0,
visionOS 2.0, *)
    public static func header() ->
UIListContentConfiguration

    @available(iOS 18.0, tvOS 18.0,
visionOS 2.0, *)
    public static func footer() ->
UIListContentConfiguration

    @available(iOS 15.0, *)
    public static func
prominentInsetGroupedHeader() ->
UIListContentConfiguration

    @available(iOS 15.0, *)
    public static func
extraProminentInsetGroupedHeader() ->
UIListContentConfiguration

    public static func
accompaniedSidebarCell() ->
UIListContentConfiguration

    public static func
```

```
accompaniedSidebarSubtitleCell() ->
UIListContentConfiguration

    @MainActor public func
makeContentView() -> any UIView &
UIContentView

    public func updated(for state: any
UIConfigurationState) ->
UIListContentConfiguration

    public var image: UIImage?

    public var imageProperties:
UIListContentConfiguration.ImagePropertie
s

    public var text: String?

    public var attributedText:
NSAttributedString?

    public var textProperties:
UIListContentConfiguration.TextProperties

    public var secondaryText: String?

    public var secondaryAttributedText:
NSAttributedString?

    public var secondaryTextProperties:
UIListContentConfiguration.TextProperties
```

```
    public var
axesPreservingSuperviewLayoutMargins:
UIAxis

    public var directionalLayoutMargins:
NSDirectionalEdgeInsets

    public var
prefersSideBySideTextAndSecondaryText:
Bool

    public var imageToTextPadding:
CGFloat

    public var
textToSecondaryTextHorizontalPadding:
CGFloat

    public var
textToSecondaryTextVerticalPadding:
CGFloat

    @available(iOS 18.0, tvOS 18.0,
visionOS 2.0, *)
    public var alpha: CGFloat

    /// Hashes the essential components
of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform
to the `Hashable` protocol. The
    /// components used for hashing must
```

```
be the same as the components compared
    /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your
implementation of `hash(into:)`,
    ///   don't call `finalize()` on the
`hasher` instance provided,
    ///   or replace it with a different
instance.
    ///   Doing so may become a compile-
time error in the future.
    ///
    /// - Parameter hasher: The hasher to
use when combining the components
    ///   of this instance.
    public func hash(into hasher: inout
Hasher)

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (a:
```

```
UIListContentConfiguration, b:
UIListContentConfiguration) -> Bool

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }
}

extension UIListContentConfiguration {

    @available(iOS, introduced: 14.0,
deprecated: 18.0, renamed: "header")
    @available(tvOS, introduced: 14.0,
deprecated: 18.0, renamed: "header")
    public static func plainHeader() ->
UIListContentConfiguration

    @available(iOS, introduced: 14.0,
deprecated: 18.0, renamed: "footer")
    @available(tvOS, introduced: 14.0,
deprecated: 18.0, renamed: "footer")
    public static func plainFooter() ->
UIListContentConfiguration

    @available(iOS, introduced: 14.0,
deprecated: 18.0, renamed: "header")
    @available(tvOS, introduced: 14.0,
deprecated: 18.0, renamed: "header")
    public static func groupedHeader() ->
UIListContentConfiguration

    @available(iOS, introduced: 14.0,
deprecated: 18.0, renamed: "footer")
    @available(tvOS, introduced: 14.0,
deprecated: 18.0, renamed: "footer")
    public static func groupedFooter() ->
UIListContentConfiguration

    @available(iOS, introduced: 14.0,
deprecated: 18.0, renamed: "cell")
    @available(tvOS, unavailable)
    public static func sidebarCell() ->
UIListContentConfiguration

    @available(iOS, introduced: 14.0,
deprecated: 18.0, renamed:
"subtitleCell")
    @available(tvOS, unavailable)
    public static func
sidebarSubtitleCell() ->
UIListContentConfiguration

    @available(iOS, introduced: 14.0,
deprecated: 18.0, renamed: "header")
    public static func sidebarHeader() ->
UIListContentConfiguration
}

@available(iOS 14.0, tvOS 14.0, *)
extension UIListContentConfiguration :
CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {

    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }

    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
get }

    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UIListContentConfiguration.ImagePropertie
s : CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {

    /// A textual representation of this
```

```
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }
```

```
    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
```

```
get }

    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UIListContentConfiguration.TextProperties
: CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {

    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }

    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
```

```
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
get }

    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}
```

```
@available(iOS 14.5, *)
@available(tvOS, unavailable)
public struct
UIListSeparatorConfiguration : Hashable {

    public enum Visibility : Hashable {

        case automatic

        case visible

        case hidden

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UIListSeparatorConfiguration.Visibility,
b:
UIListSeparatorConfiguration.Visibility)
-> Bool

        /// Hashes the essential
```

```
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// The hash value.
        ///
        /// Hash values are not
```

```
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    public var topSeparatorVisibility:
UIListSeparatorConfiguration.Visibility

    public var bottomSeparatorVisibility:
UIListSeparatorConfiguration.Visibility

    public static let automaticInsets:
NSDirectionalEdgeInsets

    public var topSeparatorInsets:
NSDirectionalEdgeInsets

    public var bottomSeparatorInsets:
NSDirectionalEdgeInsets

    public var color: UIColor
    public var multipleSelectionColor:
UIColor

    @available(iOS 15.0, *)
    @available(tvOS, unavailable)
    public var visualEffect:
UIVisualEffect?

    public init(listAppearance:
UICollectionLayoutListConfiguration.Appea
rance)

    /// Hashes the essential components
of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform
to the `Hashable` protocol. The
    /// components used for hashing must
be the same as the components compared
    /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your
implementation of `hash(into:)`,
    ///   don't call `finalize()` on the
`hasher` instance provided,
    ///   or replace it with a different
instance.
    ///   Doing so may become a compile-
time error in the future.
```

```
    ///
    /// - Parameter hasher: The hasher to
use when combining the components
    ///   of this instance.
    public func hash(into hasher: inout
Hasher)

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (a:
UIListSeparatorConfiguration, b:
UIListSeparatorConfiguration) -> Bool

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
```

```
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }
}

@available(iOS 14.5, *)
@available(tvOS, unavailable)
extension UIListSeparatorConfiguration :
CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {

    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
```

```
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }

    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
get }

    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}

@available(iOS 17.0, tvOS 17.0, *)
public protocol UIMutableTraits {

    subscript<T>(trait: T.Type) ->
```

```
T.Value where T : UITraitDefinition { get
set }

    subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == CGFloat { get set }

    subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == CGFloat? { get set }

    subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Double { get set }

    subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Double? { get set }

    subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Int { get set }

    subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Int? { get set }

    subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Bool { get set }

    subscript<T>(trait: T.Type) ->
T.Value where T :
```

```
_UICustomRawRepresentableTraitDefinition
{ get set }

    subscript<T>(trait: T.Type) ->
T.Value where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == CGFloat { get set }

    subscript<T>(trait: T.Type) ->
T.Value where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == Double { get set }

    subscript<T>(trait: T.Type) ->
T.Value where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == Int { get set }

    subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value : RawRepresentable { get set }

    subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value : RawRepresentable,
T.Value.RawValue == CGFloat { get set }

    subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value : RawRepresentable,
T.Value.RawValue == Double { get set }

    subscript<T>(trait: T.Type) ->
```

```
T.Value where T : UITraitDefinition,
T.Value : RawRepresentable,
T.Value.RawValue == Int { get set }
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {

    public var userInterfaceIdiom:
UIUserInterfaceIdiom
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {

    public var userInterfaceStyle:
UIUserInterfaceStyle
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {

    public var layoutDirection:
UITraitEnvironmentLayoutDirection
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {

    public var displayScale: CGFloat
}

@available(iOS 17.0, tvOS 17.0, *)
```

```
extension UIMutableTraits {

    public var horizontalSizeClass:
UIUserInterfaceSizeClass
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {

    public var verticalSizeClass:
UIUserInterfaceSizeClass
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {

    public var forceTouchCapability:
UIForceTouchCapability
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {

    public var
preferredContentSizeCategory:
UIContentSizeCategory
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {

    public var displayGamut:
UIDisplayGamut
```

```
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {

    public var accessibilityContrast:
UIAccessibilityContrast
}

@available(iOS 17.0, *)
@available(tvOS, unavailable)
extension UIMutableTraits {

    public var userInterfaceLevel:
UIUserInterfaceLevel
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {

    public var legibilityWeight:
UILegibilityWeight
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {

    public var activeAppearance:
UIUserInterfaceActiveAppearance
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {
    public var
toolbarItemPresentationSize:
UINSToolbarItemPresentationSize
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {

    public var imageDynamicRange:
UIImage.DynamicRange
}

@available(iOS 17.0, tvOS 17.0, visionOS
1.0, *)
extension UIMutableTraits {

    public var sceneCaptureState:
UISceneCaptureState
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIMutableTraits {

    public var typesettingLanguage:
Locale.Language?
}

@available(iOS 18.0, tvOS 18.0, visionOS
2.0, *)
extension UIMutableTraits {

    public var listEnvironment:
```

```
UIListEnvironment
}

@available(iOS 16.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
@MainActor @preconcurrency public
protocol UINavigationItemRenameDelegate :
AnyObject {

    @MainActor @preconcurrency func
navigationItem(_: UINavigationItem,
didEndRenamingWith title: String)

    @MainActor @preconcurrency func
navigationItemShouldBeginRenaming(_:
UINavigationItem) -> Bool

    @MainActor @preconcurrency func
navigationItem(_: UINavigationItem,
willBeginRenamingWith suggestedTitle:
String, selectedRange:
Range<String.Index>) -> (String,
Range<String.Index>)

    @MainActor @preconcurrency func
navigationItem(_: UINavigationItem,
shouldEndRenamingWith title: String) ->
Bool
}

@available(iOS 16.0, *)
extension UINavigationItemRenameDelegate
```

```
{

    @MainActor @preconcurrency public
func navigationItemShouldBeginRenaming(_
navigationItem: UINavigationItem) -> Bool

    @MainActor @preconcurrency public
func navigationItem(_ navigationItem:
UINavigationItem, willBeginRenamingWith
suggestedTitle: String, selectedRange:
Range<String.Index>) -> (String,
Range<String.Index>)

    @MainActor @preconcurrency public
func navigationItem(_ navigationItem:
UINavigationItem, shouldEndRenamingWith
title: String) -> Bool
}

@available(swift, introduced: 1.0,
deprecated: 4.2, message: "Use ==
operator instead.")
@available(iOS, introduced: 7.0,
deprecated: 12.0, message: "Use ==
operator instead.")
@available(watchOS, introduced: 2.0,
deprecated: 5.0, message: "Use ==
operator instead.")
public func UIOffsetEqualToOffset(_
offset1: UIOffset, _ offset2: UIOffset)
-> Bool

@available(iOS 13.4, *)
```

```
public enum UIPointerEffect : Sendable,
Equatable {

    public enum TintMode : Sendable,
Equatable {

        case none

        case overlay

        case underlay

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
```

```
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UIPointerEffect.TintMode, b:
UIPointerEffect.TintMode) -> Bool

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
```

```
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    case automatic(UITargetedPreview)

    case highlight(UITargetedPreview)

    case lift(UITargetedPreview)

    case hover(UITargetedPreview,
preferredTintMode:
UIPointerEffect.TintMode = .overlay,
prefersShadow: Bool = false,
prefersScaledContent: Bool = true)

    public var preview: UITargetedPreview
{ get }

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (a:
UIPointerEffect, b: UIPointerEffect) ->
Bool
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
extension UIPointerEffect : UIHoverEffect
{
}

@available(iOS 13.4, *)
extension UIPointerEffect.TintMode :
Hashable {
}

@available(iOS 13.4, *)
public enum UIPointerShape {

    case path(UIBezierPath)

    case roundedRect(CGRect, radius:
CGFloat =
UIPointerShape.defaultCornerRadius)
```

```
    case verticalBeam(length: CGFloat)

    case horizontalBeam(length: CGFloat)

    public static let
defaultCornerRadius: CGFloat
}

@available(iOS 17.0, tvOS 17.0, *)
public struct
UISceneSessionActivationRequest :
Hashable {

    public var role: UISceneSession.Role
{ get }

    public var session: UISceneSession? {
get }

    public var userActivity:
NSUserActivity?

    public var options:
UIScene.ActivationRequestOptions?

    public init(role: UISceneSession.Role
= .windowApplication, userActivity:
NSUserActivity? = nil, options:
UIScene.ActivationRequestOptions? = nil)

    public init(session: UISceneSession,
userActivity: NSUserActivity? = nil,
options:
```

```
UIScene.ActivationRequestOptions? = nil)

    /// Hashes the essential components
of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform
to the `Hashable` protocol. The
    /// components used for hashing must
be the same as the components compared
    /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your
implementation of `hash(into:)`,
    ///   don't call `finalize()` on the
`hasher` instance provided,
    ///   or replace it with a different
instance.
    ///   Doing so may become a compile-
time error in the future.
    ///
    /// - Parameter hasher: The hasher to
use when combining the components
    ///   of this instance.
    public func hash(into hasher: inout
Hasher)

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
```

```
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (a:
UISceneSessionActivationRequest, b:
UISceneSessionActivationRequest) -> Bool

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }
}

@available(iOS 18.0, tvOS 18.0, visionOS
2.0, *)
public struct UIShadowProperties :
```

```
Hashable {

    public var color: UIColor

    public var opacity: CGFloat

    public var radius: CGFloat

    public var offset: CGSize

    public var path: UIBezierPath?

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (lhs:
UIShadowProperties, rhs:
UIShadowProperties) -> Bool

    /// Hashes the essential components
of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform
```

```
to the `Hashable` protocol. The
    /// components used for hashing must
be the same as the components compared
    /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your
implementation of `hash(into:)`,
    ///   don't call `finalize()` on the
`hasher` instance provided,
    ///   or replace it with a different
instance.
    ///   Doing so may become a compile-
time error in the future.
    ///
    /// - Parameter hasher: The hasher to
use when combining the components
    ///   of this instance.
    public func hash(into hasher: inout
Hasher)

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }
}

@available(iOS 18.0, tvOS 18.0, visionOS
2.0, *)
extension UIShadowProperties :
CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {

    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }

    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
```

```
String {
    ///                 return "(\(x), \(y))"
    ///             }
    ///         }
    ///
    ///         let p = Point(x: 21, y: 30)
    ///         let s = String(reflecting: p)
    ///         print(s)
    ///         // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
get }

    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
public struct UIShape : UIShapeProvider {

    public static var rect: UIShape { get
```

```
}

    public static var capsule: UIShape {
get }

    public static var circle: UIShape {
get }

    public static func rect(cornerRadius:
CGFloat, cornerCurve: UICornerCurve
= .automatic, maskedCorners: UIRectCorner
= .allCorners) -> UIShape

    public static func fixedRect(_ rect:
CGRect, cornerRadius: CGFloat = 0,
cornerCurve: UICornerCurve = .automatic,
maskedCorners: UIRectCorner
= .allCorners) -> UIShape

    public static func path(_ path:
UIBezierPath) -> UIShape

    public func inset(by insets:
UIEdgeInsets) -> UIShape

    public func inset(by amount: CGFloat)
-> UIShape

    public init(_ provider: some
UIShapeProvider)

    /// Returns a Boolean value
indicating whether two values are equal.
```

```
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (a: UIShape, b:
UIShape) -> Bool
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
extension UIShape {

    public struct ResolutionContext {

        public var contentShape:
UIShape.Resolved
    }

    public func resolve(in context:
UIShape.ResolutionContext) ->
UIShape.Resolved

    public struct Resolved : Equatable {

        public let shape: UIShape

        public var boundingRect: CGRect {
```

```
get }

        public var path: UIBezierPath {
get }

        public func inset(by insets:
UIEdgeInsets) -> UIShape.Resolved

        public func inset(by amount:
CGFloat) -> UIShape.Resolved

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (lhs:
UIShape.Resolved, rhs: UIShape.Resolved)
-> Bool
    }
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
extension UIShape :
CustomStringConvertible,
CustomDebugStringConvertible {

    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
```

```
    public var description: String {
get }

    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
```

```
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
get }
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
extension UIShape.Resolved :
CustomStringConvertible,
CustomDebugStringConvertible {

    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }

    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
get }
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
public protocol UIShapeProvider :
Equatable {

    func resolve(in context:
Self.Context) -> Self.Resolved

    typealias Context =
UIShape.ResolutionContext

    typealias Resolved = UIShape.Resolved
}

@available(iOS 17.0, tvOS 17.0, *)
public typealias UISymbolEffectCompletion
= (UISymbolEffectCompletionContext) ->
```

```
Void

@available(iOS 17.0, tvOS 17.0, *)
@MainActor public struct
UISymbolEffectCompletionContext {

    @MainActor public var isFinished:
Bool { get }

    @MainActor weak public var sender:
AnyObject? { get }

    @MainActor public var effect: any
SymbolEffect { get }
}

@available(iOS 17.0, tvOS 17.0, *)
extension UISymbolEffectCompletionContext
: Sendable {
}

@available(iOS 13.0, tvOS 13.0, *)
@MainActor @preconcurrency open class
UITableViewDiffableDataSource<SectionIden
tifierType, ItemIdentifierType> :
NSObject, UITableViewDataSource where
SectionIdentifierType : Hashable,
SectionIdentifierType : Sendable,
ItemIdentifierType : Hashable,
ItemIdentifierType : Sendable {

    public typealias CellProvider = (_
tableView: UITableView, _ indexPath:
```

```
IndexPath, _ itemIdentifier:
ItemIdentifierType) -> UITableViewCell?

    @MainActor @preconcurrency public
init(tableView: UITableView,
cellProvider: @escaping
UITableViewDiffableDataSource<SectionIden
tifierType,
ItemIdentifierType>.CellProvider)

    @MainActor @preconcurrency open func
apply(_ snapshot:
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType>,
animatingDifferences: Bool = true,
completion: (() -> Void)? = nil)

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency open func
apply(_ snapshot:
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType>,
animatingDifferences: Bool = true) async

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency open func
applySnapshotUsingReloadData(_ snapshot:
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType>,
completion: (() -> Void)? = nil)

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency open func
```

```
applySnapshotUsingReloadData(_ snapshot:
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType>) async

    @MainActor @preconcurrency open func
snapshot() ->
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType>

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency open func
sectionIdentifier(for index: Int) ->
SectionIdentifierType?

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency open func
index(for sectionIdentifier:
SectionIdentifierType) -> Int?

    @MainActor @preconcurrency open func
itemIdentifier(for indexPath: IndexPath)
-> ItemIdentifierType?

    @MainActor @preconcurrency open func
indexPath(for itemIdentifier:
ItemIdentifierType) -> IndexPath?

    @MainActor @preconcurrency public var
defaultRowAnimation:
UITableView.RowAnimation

    @MainActor @preconcurrency open func
numberOfSections(in tableView:
```

```
UITableView) -> Int

    @MainActor @preconcurrency open func
tableView(_ tableView: UITableView,
numberOfRowsInSection section: Int) ->
Int

    @MainActor @preconcurrency open func
tableView(_ tableView: UITableView,
cellForRowAt indexPath: IndexPath) ->
UITableViewCell

    @MainActor @preconcurrency open func
tableView(_ tableView: UITableView,
titleForHeaderInSection section: Int) ->
String?

    @MainActor @preconcurrency open func
tableView(_ tableView: UITableView,
titleForFooterInSection section: Int) ->
String?

    @MainActor @preconcurrency open func
tableView(_ tableView: UITableView,
canEditRowAt indexPath: IndexPath) ->
Bool

    @MainActor @preconcurrency open func
tableView(_ tableView: UITableView,
commit editingStyle:
UITableViewCell.EditingStyle, forRowAt
indexPath: IndexPath)
```

```
    @MainActor @preconcurrency open func
tableView(_ tableView: UITableView,
canMoveRowAt indexPath: IndexPath) ->
Bool

    @MainActor @preconcurrency open func
tableView(_ tableView: UITableView,
moveRowAt sourceIndexPath: IndexPath, to
destinationIndexPath: IndexPath)

    @MainActor @preconcurrency open func
sectionIndexTitles(for tableView:
UITableView) -> [String]?

    @MainActor @preconcurrency open func
tableView(_ tableView: UITableView,
sectionForSectionIndexTitle title:
String, at index: Int) -> Int

    @MainActor @preconcurrency public
func description() -> String
}

@available(iOS 13.0, tvOS 13.0, *)
extension UITableViewDiffableDataSource :
Sendable {
}

@available(iOS 16.0, *)
public struct
UITextSearchAggregator<DocumentIdentifier
> where DocumentIdentifier : Hashable {
    public var allFoundRanges:
[UITextRange] { get }

    public func foundRange(_ range:
UITextRange, searchString: String,
document: DocumentIdentifier)

    public func invalidateFoundRange(_
range: UITextRange, document:
DocumentIdentifier)

    public func invalidate()

    public func finishedSearching()
}

@available(iOS 16.0, *)
public protocol UITextSearching :
NSObjectProtocol {

    associatedtype DocumentIdentifier :
Hashable = AnyHashable?

    var selectedTextRange: UITextRange? {
get }

    func compare(_ foundRange:
UITextRange, toRange: UITextRange,
document: Self.DocumentIdentifier?) ->
ComparisonResult

    func performTextSearch(queryString:
String, options: UITextSearchOptions,
resultAggregator:
UITextSearchAggregator<Self.DocumentIdent
ifier>)

    func decorate(foundTextRange:
UITextRange, document:
Self.DocumentIdentifier?, usingStyle:
UITextSearchFoundTextStyle)

    func clearAllDecoratedFoundText()

    var supportsTextReplacement: Bool {
get }

    func shouldReplace(foundTextRange:
UITextRange, document:
Self.DocumentIdentifier?, withText:
String) -> Bool

    func replace(foundTextRange:
UITextRange, document:
Self.DocumentIdentifier?, withText:
String)

    func replaceAll(queryString: String,
options: UITextSearchOptions, withText:
String)

    func willHighlight(foundTextRange:
UITextRange, document:
Self.DocumentIdentifier?)

    func scrollRangeToVisible(_ range:
```

```
UITextRange, inDocument:
Self.DocumentIdentifier?)

    var selectedTextSearchDocument:
Self.DocumentIdentifier? { get }

    func compare(document:
Self.DocumentIdentifier, toDocument:
Self.DocumentIdentifier) ->
ComparisonResult
}

@available(iOS 16.0, *)
extension UITextSearching {

    public var supportsTextReplacement:
Bool { get }

    public func
shouldReplace(foundTextRange:
UITextRange, document:
Self.DocumentIdentifier?, withText:
String) -> Bool

    public func replace(foundTextRange:
UITextRange, document:
Self.DocumentIdentifier?, withText:
String)

    public func replaceAll(queryString:
String, options: UITextSearchOptions,
withText: String)
```

```
    public func
willHighlight(foundTextRange:
UITextRange, document:
Self.DocumentIdentifier?)

    public func scrollRangeToVisible(_
range: UITextRange, inDocument:
Self.DocumentIdentifier?)

    public var
selectedTextSearchDocument:
Self.DocumentIdentifier? { get }

    public func compare(document:
Self.DocumentIdentifier, toDocument:
Self.DocumentIdentifier) ->
ComparisonResult
}

@available(iOS 17.0, tvOS 17.0, *)
public typealias UITrait = any
UITraitDefinition.Type

@available(iOS 17.0, tvOS 17.0, *)
public struct
UITraitAccessibilityContrast :
UITraitDefinition {

    public static let defaultValue:
UIAccessibilityContrast

    public static let name: String
    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
UIAccessibilityContrast
}

@available(iOS 17.0, tvOS 17.0, *)
public struct UITraitActiveAppearance :
UITraitDefinition {

    public static let defaultValue:
UIUserInterfaceActiveAppearance

    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
UIUserInterfaceActiveAppearance
}

@available(iOS 17.0, tvOS 17.0, *)
@MainActor public protocol
UITraitChangeObservable {
    typealias
TraitChangeHandler<TraitEnvironment> = (_
traitEnvironment: TraitEnvironment, _
previousTraitCollection:
UITraitCollection) -> Void where
TraitEnvironment : UITraitEnvironment

    @discardableResult
    @MainActor func
registerForTraitChanges<TraitEnvironment>
(_ traits: [UITrait], handler: @escaping
Self.TraitChangeHandler<TraitEnvironment>
) -> any UITraitChangeRegistration where
TraitEnvironment : UITraitEnvironment

    @discardableResult
    @MainActor func
registerForTraitChanges(_ traits:
[UITrait], target: Any, action: Selector)
-> any UITraitChangeRegistration

    @discardableResult
    @MainActor func
registerForTraitChanges(_ traits:
[UITrait], action: Selector) -> any
UITraitChangeRegistration

    @MainActor func
unregisterForTraitChanges(_ registration:
any UITraitChangeRegistration)
}

@available(iOS 17.0, tvOS 17.0, *)
```

```
public protocol UITraitDefinition {

    associatedtype Value

    static var defaultValue: Self.Value {
get }

    static var identifier: String { get }

    static var name: String { get }

    static var affectsColorAppearance:
Bool { get }
}

@available(iOS 17.0, tvOS 17.0, *)
extension UITraitDefinition {

    public static var identifier: String
{ get }

    public static var name: String {
get }

    public static var
affectsColorAppearance: Bool { get }
}

@available(iOS 17.0, tvOS 17.0, *)
public struct UITraitDisplayGamut :
UITraitDefinition {

    public static let defaultValue:
```

```
UIDisplayGamut

    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
UIDisplayGamut
}

@available(iOS 17.0, tvOS 17.0, *)
public struct UITraitDisplayScale :
UITraitDefinition {

    public static let defaultValue:
CGFloat

    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value = CGFloat
}

@available(iOS 17.0, tvOS 17.0, *)
```

```
public struct UITraitForceTouchCapability
: UITraitDefinition {

    public static let defaultValue:
UIForceTouchCapability

    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
UIForceTouchCapability
}

@available(iOS 17.0, tvOS 17.0, *)
public struct
UITraitHorizontalSizeClass :
UITraitDefinition {

    public static let defaultValue:
UIUserInterfaceSizeClass

    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool
```

```
    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
UIUserInterfaceSizeClass
}

@available(iOS 17.0, tvOS 17.0, *)
public struct UITraitImageDynamicRange :
UITraitDefinition {

    public static let defaultValue:
UIImage.DynamicRange

    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
UIImage.DynamicRange
}

@available(iOS 17.0, tvOS 17.0, *)
public struct UITraitLayoutDirection :
UITraitDefinition {

    public static let defaultValue:
UITraitEnvironmentLayoutDirection

    public static let name: String
    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
UITraitEnvironmentLayoutDirection
}

@available(iOS 17.0, tvOS 17.0, *)
public struct UITraitLegibilityWeight :
UITraitDefinition {

    public static let defaultValue:
UILegibilityWeight

    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
UILegibilityWeight
}

@available(iOS 18.0, tvOS 18.0, visionOS
2.0, *)
public struct UITraitListEnvironment :
UITraitDefinition {
    public static let defaultValue:
UIListEnvironment

    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 18.0, tvOS 18.0,
visionOS 2.0, *)
    public typealias Value =
UIListEnvironment
}

@available(iOS 17.0, tvOS 17.0, *)
public struct UITraitOverrides :
UIMutableTraits {

    public func contains(_ trait:
UITrait) -> Bool

    public mutating func remove(_ trait:
UITrait)

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == CGFloat
```

```
    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == CGFloat?

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Double

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Double?

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Int

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Int?

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Bool

    public subscript<T>(trait: T.Type) ->
T.Value where T :
_UICustomRawRepresentableTraitDefinition

    public subscript<T>(trait: T.Type) ->
T.Value where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == CGFloat
```

```
    public subscript<T>(trait: T.Type) ->
T.Value where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == Double

    public subscript<T>(trait: T.Type) ->
T.Value where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == Int

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value : RawRepresentable

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value : RawRepresentable,
T.Value.RawValue == CGFloat

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value : RawRepresentable,
T.Value.RawValue == Double

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value : RawRepresentable,
T.Value.RawValue == Int
}

@available(iOS 17.0, tvOS 17.0, *)
extension UITraitOverrides :
```

```
CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {

    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
```

```
    /// `Point` type's `description`
property.
    public var description: String {
get }

    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
get }

    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}

@available(iOS 17.0, tvOS 17.0, *)
public struct
UITraitPreferredContentSizeCategory :
UITraitDefinition {

    public static let defaultValue:
UIContentSizeCategory

    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool
```

```
    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
UIContentSizeCategory
}

@available(iOS 17.0, tvOS 17.0, visionOS
1.0, *)
public struct UITraitSceneCaptureState :
UITraitDefinition {

    public static let defaultValue:
UISceneCaptureState

    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, tvOS 17.0,
visionOS 1.0, *)
    public typealias Value =
UISceneCaptureState
}

@available(iOS 17.0, tvOS 17.0, *)
public struct
UITraitToolbarItemPresentationSize :
UITraitDefinition {

    public static let defaultValue:
UINSToolbarItemPresentationSize
```

```
    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
UINSToolbarItemPresentationSize
}

@available(iOS 17.0, tvOS 17.0, *)
public struct
UITraitTypesettingLanguage :
UITraitDefinition {

    public static let defaultValue:
Locale.Language?

    public static let name: String

    public static let identifier: String

    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
Locale.Language?
}

@available(iOS 17.0, tvOS 17.0, *)
public struct UITraitUserInterfaceIdiom :
UITraitDefinition {
    public static let defaultValue:
UIUserInterfaceIdiom

    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
UIUserInterfaceIdiom
}

@available(iOS 17.0, *)
@available(tvOS, unavailable)
public struct UITraitUserInterfaceLevel :
UITraitDefinition {

    public static let defaultValue:
UIUserInterfaceLevel

    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, *)
    @available(tvOS, unavailable)
    public typealias Value =
UIUserInterfaceLevel
}

@available(iOS 17.0, tvOS 17.0, *)
public struct UITraitUserInterfaceStyle :
UITraitDefinition {

    public static let defaultValue:
UIUserInterfaceStyle

    public static let name: String

    public static let identifier: String

    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
UIUserInterfaceStyle
}

@available(iOS 17.0, tvOS 17.0, *)
public struct UITraitVerticalSizeClass :
UITraitDefinition {

    public static let defaultValue:
UIUserInterfaceSizeClass

    public static let name: String

    public static let identifier: String
    public static let
affectsColorAppearance: Bool

    @available(iOS 17.0, tvOS 17.0, *)
    public typealias Value =
UIUserInterfaceSizeClass
}

@available(iOS 14.0, tvOS 14.0, *)
public struct UIViewConfigurationState :
UIConfigurationState, Hashable {

    public var traitCollection:
UITraitCollection

    public var isDisabled: Bool

    public var isHighlighted: Bool

    public var isSelected: Bool

    public var isFocused: Bool

    @available(iOS 15.0, tvOS 15.0, *)
    public var isPinned: Bool

    public subscript(key:
UIConfigurationStateCustomKey) ->
AnyHashable?

    public init(traitCollection:
UITraitCollection)
```

```
    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (lhs:
UIViewConfigurationState, rhs:
UIViewConfigurationState) -> Bool

    /// Hashes the essential components
of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform
to the `Hashable` protocol. The
    /// components used for hashing must
be the same as the components compared
    /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your
implementation of `hash(into:)`,
    ///   don't call `finalize()` on the
`hasher` instance provided,
    ///   or replace it with a different
instance.
    ///   Doing so may become a compile-
time error in the future.
    ///
    /// - Parameter hasher: The hasher to
use when combining the components
    ///   of this instance.
    public func hash(into hasher: inout
Hasher)

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }
}

@available(iOS 14.0, tvOS 14.0, *)
extension UIViewConfigurationState :
CustomStringConvertible,
CustomDebugStringConvertible,
CustomReflectable {

    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
```

```
    public var description: String {
get }

    /// A textual representation of this
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
```

```
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
get }

    /// The custom mirror for this
instance.
    ///
    /// If this type has value semantics,
the mirror should be unaffected by
    /// subsequent mutations of the
instance.
    public var customMirror: Mirror { get
}
}

@available(swift 5.1)
@available(iOS 15, tvOS 15, *)
public protocol UIViewInvalidating {

    func invalidate(view: UIView)
}

@available(swift 5.1)
@available(iOS 15, tvOS 15, *)
extension UIViewInvalidating where Self
== UIView.Invalidations.Display {

    public static var display:
UIView.Invalidations.Display { get }
}

@available(swift 5.1)
```

```
@available(iOS 15, tvOS 15, *)
extension UIViewInvalidating where Self
== UIView.Invalidations.Layout {

    public static var layout:
UIView.Invalidations.Layout { get }
}

@available(swift 5.1)
@available(iOS 15, tvOS 15, *)
extension UIViewInvalidating where Self
== UIView.Invalidations.Constraints {

    public static var constraints:
UIView.Invalidations.Constraints { get }
}

@available(swift 5.1)
@available(iOS 15, tvOS 15, *)
extension UIViewInvalidating where Self
==
UIView.Invalidations.IntrinsicContentSize
{

    public static var
intrinsicContentSize:
UIView.Invalidations.IntrinsicContentSize
{ get }
}

@available(swift 5.1)
@available(iOS 15, tvOS 15, *)
extension UIViewInvalidating where Self
```

```
== UIView.Invalidations.Configuration {

    public static var configuration:
UIView.Invalidations.Configuration {
get }
}

@available(iOS 17.0, tvOS 17.0, *)
public protocol UIWindowScenePlacement :
Hashable {
}

@available(iOS 17.0, *)
extension UIWindowScenePlacement where
Self == UIWindowSceneProminentPlacement {

    public static func prominent() ->
Self
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIWindowScenePlacement where
Self == UIWindowSceneStandardPlacement {

    public static func standard() -> Self
}

@available(iOS 17.0, *)
public struct
UIWindowSceneProminentPlacement :
UIWindowScenePlacement {

    public init()
    /// Hashes the essential components
of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform
to the `Hashable` protocol. The
    /// components used for hashing must
be the same as the components compared
    /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your
implementation of `hash(into:)`,
    ///   don't call `finalize()` on the
`hasher` instance provided,
    ///   or replace it with a different
instance.
    ///   Doing so may become a compile-
time error in the future.
    ///
    /// - Parameter hasher: The hasher to
use when combining the components
    ///   of this instance.
    public func hash(into hasher: inout
Hasher)

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (a:
UIWindowSceneProminentPlacement, b:
UIWindowSceneProminentPlacement) -> Bool

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }
}

@available(iOS 17.0, tvOS 17.0, *)
public struct
UIWindowSceneStandardPlacement :
UIWindowScenePlacement {
    public init()

    /// Hashes the essential components
of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform
to the `Hashable` protocol. The
    /// components used for hashing must
be the same as the components compared
    /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your
implementation of `hash(into:)`,
    ///   don't call `finalize()` on the
`hasher` instance provided,
    ///   or replace it with a different
instance.
    ///   Doing so may become a compile-
time error in the future.
    ///
    /// - Parameter hasher: The hasher to
use when combining the components
    ///   of this instance.
    public func hash(into hasher: inout
Hasher)

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (a:
UIWindowSceneStandardPlacement, b:
UIWindowSceneStandardPlacement) -> Bool

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }
}

@available(iOS 17.0, tvOS 17.0, *)
extension UITraitCollection {
    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition { get
}

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == CGFloat { get }

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == CGFloat? { get }

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Double { get }

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Double? { get }

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Int { get }

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Int? { get }

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value == Bool { get }
```

```
    public subscript<T>(trait: T.Type) ->
T.Value where T :
_UICustomRawRepresentableTraitDefinition
{ get }

    public subscript<T>(trait: T.Type) ->
T.Value where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == CGFloat { get }

    public subscript<T>(trait: T.Type) ->
T.Value where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == Double { get }

    public subscript<T>(trait: T.Type) ->
T.Value where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == Int { get }

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value : RawRepresentable { get }

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value : RawRepresentable,
T.Value.RawValue == CGFloat { get }

    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value : RawRepresentable,
T.Value.RawValue == Double { get }
```

```
    public subscript<T>(trait: T.Type) ->
T.Value where T : UITraitDefinition,
T.Value : RawRepresentable,
T.Value.RawValue == Int { get }

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
UITraitDefinition

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
UITraitDefinition, T.Value == CGFloat

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
UITraitDefinition, T.Value == CGFloat?

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
UITraitDefinition, T.Value == Double

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
UITraitDefinition, T.Value == Double?

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
```

```
UITraitCollection where T :
UITraitDefinition, T.Value == Int

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
UITraitDefinition, T.Value == Int?

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
UITraitDefinition, T.Value == Bool

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
_UICustomRawRepresentableTraitDefinition

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == CGFloat

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == Double

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
```

```
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == Int

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
UITraitDefinition, T.Value :
RawRepresentable

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
UITraitDefinition, T.Value :
RawRepresentable, T.Value.RawValue ==
CGFloat

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
UITraitDefinition, T.Value :
RawRepresentable, T.Value.RawValue ==
Double

    public func replacing<T>(_ trait:
T.Type, value: T.Value) ->
UITraitCollection where T :
UITraitDefinition, T.Value :
RawRepresentable, T.Value.RawValue == Int

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
UITraitDefinition
```

```
    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
UITraitDefinition, T.Value == CGFloat

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
UITraitDefinition, T.Value == CGFloat?

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
UITraitDefinition, T.Value == Double

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
UITraitDefinition, T.Value == Double?

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
UITraitDefinition, T.Value == Int

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
UITraitDefinition, T.Value == Int?

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
UITraitDefinition, T.Value == Bool

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
_UICustomRawRepresentableTraitDefinition

    public convenience init<T>(_ trait:
```

```
T.Type, value: T.Value) where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == CGFloat

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == Double

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
_UICustomRawRepresentableTraitDefinition,
T._CustomRawValue == Int

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
UITraitDefinition, T.Value :
RawRepresentable

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
UITraitDefinition, T.Value :
RawRepresentable, T.Value.RawValue ==
CGFloat

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
UITraitDefinition, T.Value :
RawRepresentable, T.Value.RawValue ==
Double

    public convenience init<T>(_ trait:
T.Type, value: T.Value) where T :
```

```
UITraitDefinition, T.Value :
RawRepresentable, T.Value.RawValue == Int

    public typealias TraitMutations = (_
mutableTraits: inout any UIMutableTraits)
-> Void

    public convenience init(mutations: (_
mutableTraits: inout any UIMutableTraits)
-> Void)

    public func modifyingTraits(_
mutations: (_ mutableTraits: inout any
UIMutableTraits) -> Void) ->
UITraitCollection

    public func changedTraits(from
traitCollection: UITraitCollection?) ->
[UITrait]

    public static var
systemTraitsAffectingColorAppearance:
[UITrait] { get }

    public static var
systemTraitsAffectingImageLookup:
[UITrait] { get }
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIView :
UITraitChangeObservable {
    @discardableResult
    @MainActor public func
registerForTraitChanges<Self>(_ traits:
[UITrait], handler: @escaping
UIView.TraitChangeHandler<Self>) -> any
UITraitChangeRegistration where Self :
UITraitEnvironment

    @discardableResult
    @MainActor public func
registerForTraitChanges(_ traits:
[UITrait], target: Any, action: Selector)
-> any UITraitChangeRegistration

    @discardableResult
    @MainActor public func
registerForTraitChanges(_ traits:
[UITrait], action: Selector) -> any
UITraitChangeRegistration

    @MainActor public func
unregisterForTraitChanges(_ registration:
any UITraitChangeRegistration)

    @MainActor @preconcurrency public var
traitOverrides: UITraitOverrides
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIViewController :
UITraitChangeObservable {

    @discardableResult
    @MainActor public func
registerForTraitChanges<Self>(_ traits:
[UITrait], handler: @escaping
UIViewController.TraitChangeHandler<Self>
) -> any UITraitChangeRegistration where
Self : UITraitEnvironment

    @discardableResult
    @MainActor public func
registerForTraitChanges(_ traits:
[UITrait], target: Any, action: Selector)
-> any UITraitChangeRegistration

    @discardableResult
    @MainActor public func
registerForTraitChanges(_ traits:
[UITrait], action: Selector) -> any
UITraitChangeRegistration

    @MainActor public func
unregisterForTraitChanges(_ registration:
any UITraitChangeRegistration)

    @MainActor @preconcurrency public var
traitOverrides: UITraitOverrides
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIPresentationController :
UITraitChangeObservable {

    @discardableResult
    @MainActor public func
```

```
registerForTraitChanges<Self>(_ traits:
[UITrait], handler: @escaping
UIPresentationController.TraitChangeHandl
er<Self>) -> any
UITraitChangeRegistration where Self :
UITraitEnvironment

    @discardableResult
    @MainActor public func
registerForTraitChanges(_ traits:
[UITrait], target: Any, action: Selector)
-> any UITraitChangeRegistration

    @discardableResult
    @MainActor public func
registerForTraitChanges(_ traits:
[UITrait], action: Selector) -> any
UITraitChangeRegistration

    @MainActor public func
unregisterForTraitChanges(_ registration:
any UITraitChangeRegistration)

    @MainActor @preconcurrency public var
traitOverrides: UITraitOverrides
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIWindowScene :
UITraitChangeObservable {

    @discardableResult
    @MainActor public func
```

```
registerForTraitChanges<Self>(_ traits:
[UITrait], handler: @escaping
UIWindowScene.TraitChangeHandler<Self>)
-> any UITraitChangeRegistration where
Self : UITraitEnvironment

    @discardableResult
    @MainActor public func
registerForTraitChanges(_ traits:
[UITrait], target: Any, action: Selector)
-> any UITraitChangeRegistration

    @discardableResult
    @MainActor public func
registerForTraitChanges(_ traits:
[UITrait], action: Selector) -> any
UITraitChangeRegistration

    @MainActor public func
unregisterForTraitChanges(_ registration:
any UITraitChangeRegistration)

    @MainActor @preconcurrency public var
traitOverrides: UITraitOverrides
}

@available(iOS 17.0, tvOS 17.0, *)
extension UITraitCollection {

    public var typesettingLanguage:
Locale.Language? { get }

    public convenience
```

```
init(typesettingLanguage:
Locale.Language?)
}

extension UIViewController {

    @available(swift 5.1)
    @available(iOS 16.4, tvOS 16.4, *)
    @propertyWrapper public struct
ViewLoading<Value> {

        public init()

        public init(wrappedValue: Value)
    }
}

@available(iOS 14.0, tvOS 14.0, *)
extension UICollectionViewListCell {

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public var
accessories: [UICellAccessory]
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
extension UICornerCurve :
CustomStringConvertible,
CustomDebugStringConvertible {

    /// A textual representation of this
instance.
```

```
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }

    /// A textual representation of this
```

```
instance, suitable for debugging.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(reflecting:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `debugDescription` property for
types that conform to
    /// `CustomDebugStringConvertible`:
    ///
    ///     struct Point:
CustomDebugStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var debugDescription:
String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(reflecting: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `debugDescription`
property.
    public var debugDescription: String {
get }
```

```
}

@available(iOS 15.0, tvOS 15.0, *)
extension UIButton {

    public struct Configuration :
Hashable {

        public enum Size {

            case mini

            case small

            case medium

            case large

            /// Returns a Boolean value
indicating whether two values are equal.
            ///
            /// Equality is the inverse
of inequality. For any values `a` and
`b`,
            /// `a == b` implies that
`a != b` is `false`.
            ///
            /// - Parameters:
            ///   - lhs: A value to
compare.
            ///   - rhs: Another value to
compare.
            public static func == (a:
```

```
UIButton.Configuration.Size, b:
UIButton.Configuration.Size) -> Bool

            /// Hashes the essential
components of this value by feeding them
into the
            /// given hasher.
            ///
            /// Implement this method to
conform to the `Hashable` protocol. The
            /// components used for
hashing must be the same as the
components compared
            /// in your type's `==`
operator implementation. Call
`hasher.combine(_:)`
            /// with each of these
components.
            ///
            /// - Important: In your
implementation of `hash(into:)`,
            ///   don't call `finalize()`
on the `hasher` instance provided,
            ///   or replace it with a
different instance.
            ///   Doing so may become a
compile-time error in the future.
            ///
            /// - Parameter hasher: The
hasher to use when combining the
components
            ///   of this instance.
            public func hash(into hasher:
```

```
inout Hasher)

            /// The hash value.
            ///
            /// Hash values are not
guaranteed to be equal across different
executions of
            /// your program. Do not save
hash values to use during a future
execution.
            ///
            /// - Important: `hashValue`
is deprecated as a `Hashable`
requirement. To
            ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
            ///   The compiler provides
an implementation for `hashValue` for
you.
            public var hashValue: Int {
get }
        }

        public enum TitleAlignment {

            case automatic

            case leading

            case center

            case trailing
            /// Returns a Boolean value
indicating whether two values are equal.
            ///
            /// Equality is the inverse
of inequality. For any values `a` and
`b`,
            /// `a == b` implies that
`a != b` is `false`.
            ///
            /// - Parameters:
            ///   - lhs: A value to
compare.
            ///   - rhs: Another value to
compare.
            public static func == (a:
UIButton.Configuration.TitleAlignment, b:
UIButton.Configuration.TitleAlignment) ->
Bool

            /// Hashes the essential
components of this value by feeding them
into the
            /// given hasher.
            ///
            /// Implement this method to
conform to the `Hashable` protocol. The
            /// components used for
hashing must be the same as the
components compared
            /// in your type's `==`
operator implementation. Call
`hasher.combine(_:)`
```

```
            /// with each of these
components.
            ///
            /// - Important: In your
implementation of `hash(into:)`,
            ///   don't call `finalize()`
on the `hasher` instance provided,
            ///   or replace it with a
different instance.
            ///   Doing so may become a
compile-time error in the future.
            ///
            /// - Parameter hasher: The
hasher to use when combining the
components
            ///   of this instance.
            public func hash(into hasher:
inout Hasher)

            /// The hash value.
            ///
            /// Hash values are not
guaranteed to be equal across different
executions of
            /// your program. Do not save
hash values to use during a future
execution.
            ///
            /// - Important: `hashValue`
is deprecated as a `Hashable`
requirement. To
            ///   conform to `Hashable`,
implement the `hash(into:)` requirement
```

```
instead.
            ///   The compiler provides
an implementation for `hashValue` for
you.
            public var hashValue: Int {
get }
        }

        public enum CornerStyle {

            case fixed

            case dynamic

            case small

            case medium

            case large

            case capsule

            /// Returns a Boolean value
indicating whether two values are equal.
            ///
            /// Equality is the inverse
of inequality. For any values `a` and
`b`,
            /// `a == b` implies that
`a != b` is `false`.
            ///
            /// - Parameters:
            ///   - lhs: A value to
```

```
compare.
            ///   - rhs: Another value to
compare.
            public static func == (a:
UIButton.Configuration.CornerStyle, b:
UIButton.Configuration.CornerStyle) ->
Bool

            /// Hashes the essential
components of this value by feeding them
into the
            /// given hasher.
            ///
            /// Implement this method to
conform to the `Hashable` protocol. The
            /// components used for
hashing must be the same as the
components compared
            /// in your type's `==`
operator implementation. Call
`hasher.combine(_:)`
            /// with each of these
components.
            ///
            /// - Important: In your
implementation of `hash(into:)`,
            ///   don't call `finalize()`
on the `hasher` instance provided,
            ///   or replace it with a
different instance.
            ///   Doing so may become a
compile-time error in the future.
            ///
            /// - Parameter hasher: The
hasher to use when combining the
components
            ///   of this instance.
            public func hash(into hasher:
inout Hasher)

            /// The hash value.
            ///
            /// Hash values are not
guaranteed to be equal across different
executions of
            /// your program. Do not save
hash values to use during a future
execution.
            ///
            /// - Important: `hashValue`
is deprecated as a `Hashable`
requirement. To
            ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
            ///   The compiler provides
an implementation for `hashValue` for
you.
            public var hashValue: Int {
get }
        }

        public enum MacIdiomStyle {

            case automatic
            case bordered

            case borderless

            case borderlessTinted

            /// Returns a Boolean value
indicating whether two values are equal.
            ///
            /// Equality is the inverse
of inequality. For any values `a` and
`b`,
            /// `a == b` implies that
`a != b` is `false`.
            ///
            /// - Parameters:
            ///   - lhs: A value to
compare.
            ///   - rhs: Another value to
compare.
            public static func == (a:
UIButton.Configuration.MacIdiomStyle, b:
UIButton.Configuration.MacIdiomStyle) ->
Bool

            /// Hashes the essential
components of this value by feeding them
into the
            /// given hasher.
            ///
            /// Implement this method to
conform to the `Hashable` protocol. The
            /// components used for
```

```
hashing must be the same as the
components compared
            /// in your type's `==`
operator implementation. Call
`hasher.combine(_:)`
            /// with each of these
components.
            ///
            /// - Important: In your
implementation of `hash(into:)`,
            ///   don't call `finalize()`
on the `hasher` instance provided,
            ///   or replace it with a
different instance.
            ///   Doing so may become a
compile-time error in the future.
            ///
            /// - Parameter hasher: The
hasher to use when combining the
components
            ///   of this instance.
            public func hash(into hasher:
inout Hasher)

            /// The hash value.
            ///
            /// Hash values are not
guaranteed to be equal across different
executions of
            /// your program. Do not save
hash values to use during a future
execution.
            ///
            /// - Important: `hashValue`
is deprecated as a `Hashable`
requirement. To
            ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
            ///   The compiler provides
an implementation for `hashValue` for
you.
            public var hashValue: Int {
get }
        }

        @available(iOS 16.0, tvOS 16.0,
*)
        public enum Indicator {

            case automatic

            case none

            case popup

            /// Returns a Boolean value
indicating whether two values are equal.
            ///
            /// Equality is the inverse
of inequality. For any values `a` and
`b`,
            /// `a == b` implies that
`a != b` is `false`.
            ///
            /// - Parameters:
```

```
            ///   - lhs: A value to
compare.
            ///   - rhs: Another value to
compare.
            public static func == (a:
UIButton.Configuration.Indicator, b:
UIButton.Configuration.Indicator) -> Bool

            /// Hashes the essential
components of this value by feeding them
into the
            /// given hasher.
            ///
            /// Implement this method to
conform to the `Hashable` protocol. The
            /// components used for
hashing must be the same as the
components compared
            /// in your type's `==`
operator implementation. Call
`hasher.combine(_:)`
            /// with each of these
components.
            ///
            /// - Important: In your
implementation of `hash(into:)`,
            ///   don't call `finalize()`
on the `hasher` instance provided,
            ///   or replace it with a
different instance.
            ///   Doing so may become a
compile-time error in the future.
            ///
            /// - Parameter hasher: The
hasher to use when combining the
components
            ///   of this instance.
            public func hash(into hasher:
inout Hasher)

            /// The hash value.
            ///
            /// Hash values are not
guaranteed to be equal across different
executions of
            /// your program. Do not save
hash values to use during a future
execution.
            ///
            /// - Important: `hashValue`
is deprecated as a `Hashable`
requirement. To
            ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
            ///   The compiler provides
an implementation for `hashValue` for
you.
            public var hashValue: Int {
get }
        }

        public static func plain() ->
UIButton.Configuration

        public static func tinted() ->
```

```
UIButton.Configuration

        public static func gray() ->
UIButton.Configuration

        public static func filled() ->
UIButton.Configuration

        public static func borderless()
-> UIButton.Configuration

        public static func bordered() ->
UIButton.Configuration

        public static func
borderedTinted() ->
UIButton.Configuration

        public static func
borderedProminent() ->
UIButton.Configuration

        public func updated(for button:
UIButton) -> UIButton.Configuration

        public var background:
UIBackgroundConfiguration

        public var cornerStyle:
UIButton.Configuration.CornerStyle

        public var buttonSize:
UIButton.Configuration.Size
```

```
        public var macIdiomStyle:
UIButton.Configuration.MacIdiomStyle

        public var baseForegroundColor:
UIColor?

        public var baseBackgroundColor:
UIColor?

        public var image: UIImage?

        public var imageColorTransformer:
UIConfigurationColorTransformer?

        public var
preferredSymbolConfigurationForImage:
UIImage.SymbolConfiguration?

        public var
showsActivityIndicator: Bool

        public var
activityIndicatorColorTransformer:
UIConfigurationColorTransformer?

        public var title: String?

        public var attributedTitle:
AttributedString?

        public var
titleTextAttributesTransformer:
```

```
UIConfigurationTextAttributesTransformer?

        public var subtitle: String?

        public var attributedSubtitle:
AttributedString?

        public var
subtitleTextAttributesTransformer:
UIConfigurationTextAttributesTransformer?

        @available(iOS 16.0, tvOS 16.0,
*)
        public var indicator:
UIButton.Configuration.Indicator

        @available(iOS 16.0, tvOS 16.0,
*)
        public var
indicatorColorTransformer:
UIConfigurationColorTransformer?

        public var contentInsets:
NSDirectionalEdgeInsets

        public mutating func
setDefaultContentInsets()

        public var imagePlacement:
NSDirectionalRectEdge

        public var imagePadding: CGFloat
        public var titlePadding: CGFloat

        public var titleAlignment:
UIButton.Configuration.TitleAlignment

        public var
automaticallyUpdateForSelection: Bool

        /// Hashes the essential
components of this value by feeding them
into the
        /// given hasher.
        ///
        /// Implement this method to
conform to the `Hashable` protocol. The
        /// components used for hashing
must be the same as the components
compared
        /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
        /// with each of these
components.
        ///
        /// - Important: In your
implementation of `hash(into:)`,
        ///   don't call `finalize()` on
the `hasher` instance provided,
        ///   or replace it with a
different instance.
        ///   Doing so may become a
compile-time error in the future.
        ///
        /// - Parameter hasher: The
```

```
hasher to use when combining the
components
        ///   of this instance.
        public func hash(into hasher:
inout Hasher)

        /// Returns a Boolean value
indicating whether two values are equal.
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UIButton.Configuration, b:
UIButton.Configuration) -> Bool

        /// The hash value.
        ///
        /// Hash values are not
guaranteed to be equal across different
executions of
        /// your program. Do not save
hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
```

```
To
        ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
        ///   The compiler provides an
implementation for `hashValue` for you.
        public var hashValue: Int { get }
    }

    @MainActor @preconcurrency public
convenience init(configuration:
UIButton.Configuration, primaryAction:
UIAction? = nil)

    @MainActor @preconcurrency public var
configuration: UIButton.Configuration?
}

@available(iOS 16.0, *)
extension
UIWindowScene.GeometryPreferences.iOS {

    public var interfaceOrientations:
UIInterfaceOrientationMask?
}

@available(iOS 16.0, *)
extension
UIWindowScene.GeometryPreferences.Mac {

    public var systemFrame: CGRect?
}
```

```
extension UIControl {

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public
func enumerateEventHandlers(_ iterator:
(UIAction?, (Any?, Selector)?,
UIControl.Event, inout Bool) -> Void)
}

extension UIButton {

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public
convenience init(type buttonType:
UIButton.ButtonType = .system,
primaryAction: UIAction?)
}

@available(iOS 18.0, *)
@available(tvOS, unavailable)
extension
UIDocumentViewController.LaunchOptions {

    @MainActor @preconcurrency public var
background: UIBackgroundConfiguration
}

extension UINavigationItem {

    @available(iOS 16.0, *)
    @available(tvOS, unavailable)
    @available(watchOS, unavailable)
    @MainActor @preconcurrency weak
```

```
public var renameDelegate: (any
UINavigationItemRenameDelegate)?
}

@available(iOS 14.0, tvOS 14.0, *)
extension UICollectionViewCell {

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public var
contentConfiguration: (any
UIContentConfiguration)?
}

@available(iOS 14.0, tvOS 14.0, *)
extension UICollectionViewListCell {

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public
func defaultContentConfiguration() ->
UIListContentConfiguration
}

@available(iOS 14.0, tvOS 14.0, *)
extension UITableViewCell {

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public var
contentConfiguration: (any
UIContentConfiguration)?

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public
func defaultContentConfiguration() ->
```

```
UIListContentConfiguration
}

@available(iOS 14.0, tvOS 14.0, *)
extension UITableViewHeaderFooterView {

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public var
contentConfiguration: (any
UIContentConfiguration)?

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public
func defaultContentConfiguration() ->
UIListContentConfiguration
}

@available(iOS 14.0, tvOS 14.0, *)
extension UIListContentView :
UIContentView {

    @available(iOS 16.0, tvOS 16.0, *)
    @MainActor public func supports(_
configuration: any
UIContentConfiguration) -> Bool

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor public var configuration:
any UIContentConfiguration

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public
convenience init(configuration:
```

```
UIListContentConfiguration)
}

@available(iOS 18.0, *)
@available(visionOS, unavailable)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
@available(macCatalyst, unavailable)
extension UITextFormattingViewController
{

    public struct FormattingStyle :
Equatable {

        public let styleKey: String

        public let title:
LocalizedStringResource

        public let attributes:
AttributeContainer

        public init(styleKey: String,
title: LocalizedStringResource,
attributes: AttributeContainer)

        public init(styleKey: String,
title: LocalizedStringResource,
attributes: [NSAttributedString.Key :
Any])

        /// Returns a Boolean value
indicating whether two values are equal.
```

```
        ///
        /// Equality is the inverse of
inequality. For any values `a` and `b`,
        /// `a == b` implies that `a !=
b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to
compare.
        public static func == (a:
UITextFormattingViewController.Formatting
Style, b:
UITextFormattingViewController.Formatting
Style) -> Bool
    }
}

@available(iOS 18.0, *)
@available(visionOS, unavailable)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
@available(macCatalyst, unavailable)
extension
UITextFormattingViewController.Configurat
ion {

    public var formattingStyles:
[UITextFormattingViewController.Formattin
gStyle]?
}

@available(tvOS, unavailable)
```

```
@available(visionOS, unavailable)
extension
UISheetPresentationController.Detent {

    @available(iOS 16.0, *)
    @MainActor @preconcurrency public
static func custom(identifier:
UISheetPresentationController.Detent.Iden
tifier? = nil, resolver: @escaping (_
context: any
UISheetPresentationControllerDetentResolu
tionContext) -> CGFloat?) ->
UISheetPresentationController.Detent

    @available(iOS 16.0, *)
    @MainActor @preconcurrency public
func resolvedValue(in context: any
UISheetPresentationControllerDetentResolu
tionContext) -> CGFloat?
}

@available(tvOS, unavailable)
extension UISheetPresentationController {

    @available(iOS 15.0, *)
    @available(visionOS, unavailable)
    @MainActor @preconcurrency public var
preferredCornerRadius: CGFloat?
}

@available(iOS 7.0, watchOS 2.0, *)
extension IndexPath {
    public init(row: Int, section: Int)

    public init(item: Int, section: Int)

    public var section: Int

    public var row: Int

    public var item: Int
}

extension URLResourceValues {

    @available(iOS, introduced: 8.0,
deprecated: 15.0, message: "Use the
QuickLookThumbnailing framework and
extension point instead")
    @available(watchOS, introduced: 2.0,
deprecated: 8.0, message: "Use the
QuickLookThumbnailing framework and
extension point instead")
    @available(visionOS, introduced: 1.0,
deprecated: 1.0, message: "Use the
QuickLookThumbnailing framework and
extension point instead")
    public var thumbnailDictionary:
[URLThumbnailDictionaryItem : UIImage]? {
get }
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIImageView {
    @MainActor @preconcurrency public
func addSymbolEffect(_ effect: some
DiscreteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true, completion:
UISymbolEffectCompletion? = nil)

    @MainActor @preconcurrency public
func addSymbolEffect(_ effect: some
IndefiniteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true, completion:
UISymbolEffectCompletion? = nil)

    @MainActor @preconcurrency public
func addSymbolEffect(_ effect: some
DiscreteSymbolEffect &
IndefiniteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true, completion:
UISymbolEffectCompletion? = nil)

    @MainActor @preconcurrency public
func removeSymbolEffect(ofType effect:
some DiscreteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true, completion:
UISymbolEffectCompletion? = nil)

    @MainActor @preconcurrency public
func removeSymbolEffect(ofType effect:
some IndefiniteSymbolEffect &
SymbolEffect, options:
```

```
SymbolEffectOptions = .default, animated:
Bool = true, completion:
UISymbolEffectCompletion? = nil)

    @MainActor @preconcurrency public
func removeSymbolEffect(ofType effect:
some DiscreteSymbolEffect &
IndefiniteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true, completion:
UISymbolEffectCompletion? = nil)

    @MainActor @preconcurrency public
func removeAllSymbolEffects(options:
SymbolEffectOptions = .default, animated:
Bool = true)

    @MainActor @preconcurrency public
func setSymbolImage(_ image: UIImage,
contentTransition: some
ContentTransitionSymbolEffect &
SymbolEffect, options:
SymbolEffectOptions = .default,
completion: UISymbolEffectCompletion? =
nil)
}

@available(iOS 17.0, tvOS 17.0, *)
extension
UIWindowScene.ActivationRequestOptions {

    @MainActor @preconcurrency public var
placement: (any UIWindowScenePlacement)?
```

```
}

@available(iOS 16.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
extension UIEditMenuConfiguration {

    @MainActor @preconcurrency public var
identifier: AnyHashable { get }

    @MainActor @preconcurrency public
convenience init(identifier:
AnyHashable?, sourcePoint: CGPoint)
}

@available(iOS 14.0, tvOS 14.0, *)
extension UICollectionView {

    public struct CellRegistration<Cell,
Item> where Cell : UICollectionViewCell {

        public typealias Handler = (_
cell: Cell, _ indexPath: IndexPath, _
itemIdentifier: Item) -> Void

        public init(handler: @escaping
UICollectionView.CellRegistration<Cell,
Item>.Handler)

        @available(visionOS, introduced:
1.0, deprecated: 1.0, message: "Loading
Interface Builder products will not be
supported in a future version of
visionOS.")
        public init(cellNib: UINib,
handler: @escaping
UICollectionView.CellRegistration<Cell,
Item>.Handler)
    }

    public struct
SupplementaryRegistration<Supplementary>
where Supplementary :
UICollectionReusableView {

        public typealias Handler = (_
supplementaryView: Supplementary, _
elementKind: String, _ indexPath:
IndexPath) -> Void

        public init(elementKind: String,
handler: @escaping
UICollectionView.SupplementaryRegistratio
n<Supplementary>.Handler)

        @available(visionOS, introduced:
1.0, deprecated: 1.0, message: "Loading
Interface Builder products will not be
supported in a future version of
visionOS.")
        public init(supplementaryNib:
UINib, elementKind: String, handler:
@escaping
UICollectionView.SupplementaryRegistratio
n<Supplementary>.Handler)
    }
    @MainActor @preconcurrency public
func dequeueConfiguredReusableCell<Cell,
Item>(using registration:
UICollectionView.CellRegistration<Cell,
Item>, for indexPath: IndexPath, item:
Item?) -> Cell where Cell :
UICollectionViewCell

    @MainActor @preconcurrency public
func
dequeueConfiguredReusableSupplementary<Su
pplementary>(using registration:
UICollectionView.SupplementaryRegistratio
n<Supplementary>, for indexPath:
IndexPath) -> Supplementary where
Supplementary : UICollectionReusableView
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIApplication {

    @MainActor @preconcurrency public
func activateSceneSession(for request:
UISceneSessionActivationRequest,
errorHandler: ((any Error) -> Void)? =
nil)
}

extension UIPasteboard {

    @available(iOS 15.0, *)
    public struct DetectedValues {
        public var patterns:
Set<PartialKeyPath<UIPasteboard.DetectedV
alues>> { get }

        public var probableWebURL: String
{ get }

        public var probableWebSearch:
String { get }

        public var number: Double? {
get }

        public var links: [DDMatchLink] {
get }

        public var phoneNumbers:
[DDMatchPhoneNumber] { get }

        public var emailAddresses:
[DDMatchEmailAddress] { get }

        public var postalAddresses:
[DDMatchPostalAddress] { get }

        public var calendarEvents:
[DDMatchCalendarEvent] { get }

        public var
shipmentTrackingNumbers:
[DDMatchShipmentTrackingNumber] { get }
```

```
        public var flightNumbers:
[DDMatchFlightNumber] { get }

        public var moneyAmounts:
[DDMatchMoneyAmount] { get }
    }
}

extension UIPasteboard {

    @available(iOS, introduced: 14.0,
deprecated: 15.0)
    @available(visionOS, introduced: 1.0,
deprecated: 1.0)
    public func detectPatterns(for
patterns:
Set<UIPasteboard.DetectionPattern>,
completionHandler: @escaping
(Result<Set<UIPasteboard.DetectionPattern
>, any Error>) -> ())

    @available(iOS 15.0, *)
    public func detectPatterns(for
keyPaths:
Set<PartialKeyPath<UIPasteboard.DetectedV
alues>>, completionHandler: @escaping
(Result<Set<PartialKeyPath<UIPasteboard.D
etectedValues>>, any Error>) -> ())

    @available(iOS 15.0, *)
    public func detectedPatterns(for
keyPaths:
Set<PartialKeyPath<UIPasteboard.DetectedV
```

```
alues>>) async throws ->
Set<PartialKeyPath<UIPasteboard.DetectedV
alues>>

    @available(iOS, introduced: 14.0,
deprecated: 15.0)
    @available(visionOS, introduced: 1.0,
deprecated: 1.0)
    public func detectPatterns(for
patterns:
Set<UIPasteboard.DetectionPattern>,
inItemSet itemSet: IndexSet?,
completionHandler: @escaping
(Result<[Set<UIPasteboard.DetectionPatter
n>], any Error>) -> ())

    @available(iOS 15.0, *)
    public func detectPatterns(for
keyPaths:
Set<PartialKeyPath<UIPasteboard.DetectedV
alues>>, inItemSet itemSet: IndexSet?,
completionHandler: @escaping
(Result<[Set<PartialKeyPath<UIPasteboard.
DetectedValues>>], any Error>) -> ())

    @available(iOS 15.0, *)
    public func detectedPatterns(for
keyPaths:
Set<PartialKeyPath<UIPasteboard.DetectedV
alues>>, inItemSet itemSet: IndexSet?)
async throws ->
[Set<PartialKeyPath<UIPasteboard.Detected
Values>>]
```

```
    @available(iOS, introduced: 14.0,
deprecated: 15.0)
    @available(visionOS, introduced: 1.0,
deprecated: 1.0)
    public func detectValues(for
patterns:
Set<UIPasteboard.DetectionPattern>,
completionHandler: @escaping
(Result<[UIPasteboard.DetectionPattern :
Any], any Error>) -> ())

    @available(iOS 15.0, *)
    public func detectValues(for
keyPaths:
Set<PartialKeyPath<UIPasteboard.DetectedV
alues>>, completionHandler: @escaping
(Result<UIPasteboard.DetectedValues, any
Error>) -> ())

    @available(iOS 15.0, *)
    public func detectedValues(for
keyPaths:
Set<PartialKeyPath<UIPasteboard.DetectedV
alues>>) async throws ->
UIPasteboard.DetectedValues

    @available(iOS, introduced: 14.0,
deprecated: 15.0)
    @available(visionOS, introduced: 1.0,
deprecated: 1.0)
    public func detectValues(for
patterns:
```

```
Set<UIPasteboard.DetectionPattern>,
inItemSet itemSet: IndexSet?,
completionHandler: @escaping
(Result<[[UIPasteboard.DetectionPattern :
Any]], any Error>) -> ())

    @available(iOS 15.0, *)
    public func detectValues(for
keyPaths:
Set<PartialKeyPath<UIPasteboard.DetectedV
alues>>, inItemSet itemSet: IndexSet?,
completionHandler: @escaping
(Result<[UIPasteboard.DetectedValues],
any Error>) -> ())

    @available(iOS 15.0, *)
    public func detectedValues(for
keyPaths:
Set<PartialKeyPath<UIPasteboard.DetectedV
alues>>, inItemSet itemSet: IndexSet?)
async throws ->
[UIPasteboard.DetectedValues]
}

extension UIBarButtonItem {

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public
convenience init(systemItem:
UIBarButtonItem.SystemItem,
primaryAction: UIAction? = nil, menu:
UIMenu? = nil)
```

```
    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public
convenience init(title: String? = nil,
image: UIImage? = nil, primaryAction:
UIAction? = nil, menu: UIMenu? = nil)

    @available(iOS 16.0, tvOS 16.0, *)
    @MainActor @preconcurrency public
convenience init(title: String?, image:
UIImage?, target: AnyObject?, action:
Selector?, menu: UIMenu? = nil)
}

extension UIBarButtonItem {

    @available(iOS 16.0, *)
    @available(tvOS, unavailable)
    @available(watchOS, unavailable)
    @MainActor @preconcurrency public
func
creatingOptionalGroup(customizationIdenti
fier: String, isInDefaultCustomization:
Bool = true) -> UIBarButtonItemGroup
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIBarButtonItem {

    @MainActor @preconcurrency public
func addSymbolEffect(_ effect: some
DiscreteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true)
```

```
    @MainActor @preconcurrency public
func addSymbolEffect(_ effect: some
IndefiniteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true)

    @MainActor @preconcurrency public
func addSymbolEffect(_ effect: some
DiscreteSymbolEffect &
IndefiniteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true)

    @MainActor @preconcurrency public
func removeSymbolEffect(ofType effect:
some DiscreteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true)

    @MainActor @preconcurrency public
func removeSymbolEffect(ofType effect:
some IndefiniteSymbolEffect &
SymbolEffect, options:
SymbolEffectOptions = .default, animated:
Bool = true)

    @MainActor @preconcurrency public
func removeSymbolEffect(ofType effect:
some DiscreteSymbolEffect &
IndefiniteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true)
```

```
    @MainActor @preconcurrency public
func removeAllSymbolEffects(options:
SymbolEffectOptions = .default, animated:
Bool = true)

    @MainActor @preconcurrency public
func setSymbolImage(_ image: UIImage,
contentTransition: some
ContentTransitionSymbolEffect &
SymbolEffect, options:
SymbolEffectOptions = .default)
}

extension
UIFontPickerViewController.Configuration
{

    @available(iOS 18.0, visionOS 2.0, *)
    @available(tvOS, unavailable)
    @available(watchOS, unavailable)
    @MainActor @preconcurrency public var
languageFilter: Predicate<[String]>?
}

@available(iOS 15.0, *)
extension UIPointerAccessory {

    public struct Position : Sendable {

        public static let defaultOffset:
CGFloat
```

```
        public var offset: CGFloat

        public var angle: CGFloat

        public init(offset: CGFloat =
Position.defaultOffset, angle: CGFloat =
0)

        public static var top:
UIPointerAccessory.Position { get }

        public static var topRight:
UIPointerAccessory.Position { get }

        public static var right:
UIPointerAccessory.Position { get }

        public static var bottomRight:
UIPointerAccessory.Position { get }

        public static var bottom:
UIPointerAccessory.Position { get }

        public static var bottomLeft:
UIPointerAccessory.Position { get }

        public static var left:
UIPointerAccessory.Position { get }

        public static var topLeft:
UIPointerAccessory.Position { get }
    }
    @MainActor @preconcurrency public
convenience init(_ shape: UIPointerShape,
position: UIPointerAccessory.Position)

    @MainActor @preconcurrency public
class func arrow(_ position:
UIPointerAccessory.Position) -> Self

    @MainActor @preconcurrency public var
shape: UIPointerShape { get }
}

@available(iOS 13.4, *)
extension UIPointerStyle {

    @MainActor @preconcurrency public
convenience init(effect: UIPointerEffect,
shape: UIPointerShape? = nil)

    @MainActor @preconcurrency public
convenience init(shape: UIPointerShape,
constrainedAxes: UIAxis = [])
}

@available(iOS 13.4, *)
extension UIPointerRegion {

    @MainActor @preconcurrency public
convenience init(rect: CGRect,
identifier: AnyHashable? = nil)

    @MainActor @preconcurrency public var
identifier: AnyHashable? { get }
```

```
}

@available(iOS 13.4, *)
extension UIButton {

    public typealias PointerStyleProvider
= (_ button: UIButton, _ proposedEffect:
UIPointerEffect, _ proposedShape:
UIPointerShape) -> UIPointerStyle?

    @MainActor @preconcurrency public var
pointerStyleProvider:
UIButton.PointerStyleProvider?
}

@available(iOS 15.0, *)
extension UIBandSelectionInteraction {

    @MainActor @preconcurrency public var
selectionRect: CGRect? { get }
}

@available(iOS 15.0, *)
@available(tvOS, unavailable)
extension UIToolTipConfiguration {

    @MainActor @preconcurrency public var
sourceRect: CGRect? { get }
}

@available(iOS 18.0, *)
@available(visionOS, unavailable)
@available(tvOS, unavailable)
```

```
@available(watchOS, unavailable)
@available(macCatalyst, unavailable)
extension UITextFormattingViewController
{

    public enum ChangeValue {

        case undefined

        case bold(Bool)

        case italic(Bool)

        case underline(Bool)

        case strikethrough(Bool)

        case font(UIFont)

        case fontSize(Double)

        case increaseFontSize

        case decreaseFontSize

        case textColor(UIColor)

        case
textList(UITextFormattingViewController.T
extList?)

        case
textAlignment(UITextFormattingViewControl
ler.TextAlignment)

        case lineHeightPointSize(Double)

        case increaseIndentation

        case decreaseIndentation

        case
highlight(UITextFormattingViewController.
Highlight?)

        case formattingStyle(String)
    }
}

extension UIBarButtonItemGroup {

    @available(iOS 16.0, *)
    @available(tvOS, unavailable)
    @available(watchOS, unavailable)
    @MainActor @preconcurrency public
class func fixedGroup(representativeItem:
UIBarButtonItem? = nil, items:
[UIBarButtonItem]) ->
UIBarButtonItemGroup

    @available(iOS 16.0, *)
    @available(tvOS, unavailable)
    @available(watchOS, unavailable)
    @MainActor @preconcurrency public
class func
movableGroup(customizationIdentifier:
```

```
String, representativeItem:
UIBarButtonItem? = nil, items:
[UIBarButtonItem]) ->
UIBarButtonItemGroup

    @available(iOS 16.0, *)
    @available(tvOS, unavailable)
    @available(watchOS, unavailable)
    @MainActor @preconcurrency public
class func
optionalGroup(customizationIdentifier:
String, isInDefaultCustomization: Bool =
true, representativeItem:
UIBarButtonItem? = nil, items:
[UIBarButtonItem]) ->
UIBarButtonItemGroup
}

extension UIViewController {

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency public
func setContentScrollView(_ scrollView:
UIScrollView?)
}

@available(iOS 11.0, tvOS 11.0, *)
extension UIContentSizeCategory {

    public var isAccessibilityCategory:
Bool { get }

    /// Returns a Boolean value
```

```
indicating whether the value of the first
    /// argument is less than that of the
second argument.
    ///
    /// This function is the only
requirement of the `Comparable` protocol.
The
    /// remainder of the relational
operator functions are implemented by the
    /// standard library for any type
that conforms to `Comparable`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func < (left:
UIContentSizeCategory, right:
UIContentSizeCategory) -> Bool

    /// Returns a Boolean value
indicating whether the value of the first
    /// argument is less than or equal to
that of the second argument.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func <= (left:
UIContentSizeCategory, right:
UIContentSizeCategory) -> Bool
```

```
    /// Returns a Boolean value
indicating whether the value of the first
    /// argument is greater than that of
the second argument.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func > (left:
UIContentSizeCategory, right:
UIContentSizeCategory) -> Bool

    /// Returns a Boolean value
indicating whether the value of the first
    /// argument is greater than or equal
to that of the second argument.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func >= (left:
UIContentSizeCategory, right:
UIContentSizeCategory) -> Bool
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIContentSizeCategory :
Comparable {
}

@available(iOS 16.0, *)
```

```
extension UITextSearchingFindSession {

    @MainActor @preconcurrency public
convenience
init<SearchableObject>(searchableObject:
SearchableObject) where
SearchableObject : UITextSearching
}

@available(iOS 16.0, *)
extension UITextView : UITextSearching {

    @MainActor @preconcurrency public
func compare(_ foundRange: UITextRange,
toRange: UITextRange, document:
AnyHashable??) -> ComparisonResult

    @MainActor @preconcurrency public
func performTextSearch(queryString:
String, options: UITextSearchOptions,
resultAggregator:
UITextSearchAggregator<AnyHashable?>)

    @MainActor @preconcurrency public
func decorate(foundTextRange:
UITextRange, document: AnyHashable??,
usingStyle: UITextSearchFoundTextStyle)

    @MainActor @preconcurrency public
func shouldReplace(foundTextRange:
UITextRange, document:
UITextView.DocumentIdentifier?, withText
text: String) -> Bool
```

```
    @MainActor @preconcurrency public
func replace(foundTextRange: UITextRange,
document: UITextView.DocumentIdentifier?,
withText text: String)

    @MainActor @preconcurrency public
func replaceAll(queryString: String,
options: UITextSearchOptions, withText
text: String)

    @MainActor @preconcurrency public
func willHighlight(foundTextRange:
UITextRange, document: AnyHashable??)

    @MainActor @preconcurrency public
func scrollRangeToVisible(_ range:
UITextRange, inDocument: AnyHashable??)

    @available(iOS 16.0, *)
    public typealias DocumentIdentifier =
AnyHashable?
}

@available(iOS 17.0, macOS 14.0, tvOS
17.0, *)
extension Preview {

    @MainActor public init(_ name:
String? = nil, traits:
PreviewTrait<Preview.ViewTraits>...,
body: @escaping @MainActor () -> UIView)
```

```
    @MainActor public init(_ name:
String? = nil, traits:
PreviewTrait<Preview.ViewTraits>...,
body: @escaping @MainActor () ->
UIViewController)
}

extension UIView {

    @available(swift 5.1)
    @available(iOS 15, tvOS 15, *)
    @propertyWrapper public struct
Invalidating<Value, InvalidationType>
where Value : Equatable, InvalidationType
: UIViewInvalidating {

        public init(wrappedValue: Value,
_ invalidation: InvalidationType)

        public init<InvalidationType1,
InvalidationType2>(wrappedValue: Value, _
invalidation1: InvalidationType1, _
invalidation2: InvalidationType2) where
InvalidationType ==
UIView.Invalidations.Tuple<InvalidationTy
pe1, InvalidationType2>,
InvalidationType1 : UIViewInvalidating,
InvalidationType2 : UIViewInvalidating

        public init<InvalidationType1,
InvalidationType2,
InvalidationType3>(wrappedValue: Value, _
invalidation1: InvalidationType1, _
```

```
invalidation2: InvalidationType2, _
invalidation3: InvalidationType3) where
InvalidationType ==
UIView.Invalidations.Tuple<UIView.Invalid
ations.Tuple<InvalidationType1,
InvalidationType2>, InvalidationType3>,
InvalidationType1 : UIViewInvalidating,
InvalidationType2 : UIViewInvalidating,
InvalidationType3 : UIViewInvalidating

        public init<InvalidationType1,
InvalidationType2, InvalidationType3,
InvalidationType4>(wrappedValue: Value, _
invalidation1: InvalidationType1, _
invalidation2: InvalidationType2, _
invalidation3: InvalidationType3, _
invalidation4: InvalidationType4) where
InvalidationType ==
UIView.Invalidations.Tuple<UIView.Invalid
ations.Tuple<InvalidationType1,
InvalidationType2>,
UIView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
InvalidationType1 : UIViewInvalidating,
InvalidationType2 : UIViewInvalidating,
InvalidationType3 : UIViewInvalidating,
InvalidationType4 : UIViewInvalidating

        public init<InvalidationType1,
InvalidationType2, InvalidationType3,
InvalidationType4,
InvalidationType5>(wrappedValue: Value, _
invalidation1: InvalidationType1, _
```

```
invalidation2: InvalidationType2, _
invalidation3: InvalidationType3, _
invalidation4: InvalidationType4, _
invalidation5: InvalidationType5) where
InvalidationType ==
UIView.Invalidations.Tuple<UIView.Invalid
ations.Tuple<UIView.Invalidations.Tuple<I
nvalidationType1, InvalidationType2>,
UIView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
InvalidationType5>, InvalidationType1 :
UIViewInvalidating, InvalidationType2 :
UIViewInvalidating, InvalidationType3 :
UIViewInvalidating, InvalidationType4 :
UIViewInvalidating, InvalidationType5 :
UIViewInvalidating

        public init<InvalidationType1,
InvalidationType2, InvalidationType3,
InvalidationType4, InvalidationType5,
InvalidationType6>(wrappedValue: Value, _
invalidation1: InvalidationType1, _
invalidation2: InvalidationType2, _
invalidation3: InvalidationType3, _
invalidation4: InvalidationType4, _
invalidation5: InvalidationType5, _
invalidation6: InvalidationType6) where
InvalidationType ==
UIView.Invalidations.Tuple<UIView.Invalid
ations.Tuple<UIView.Invalidations.Tuple<I
nvalidationType1, InvalidationType2>,
UIView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
UIView.Invalidations.Tuple<InvalidationTy
pe5, InvalidationType6>>,
InvalidationType1 : UIViewInvalidating,
InvalidationType2 : UIViewInvalidating,
InvalidationType3 : UIViewInvalidating,
InvalidationType4 : UIViewInvalidating,
InvalidationType5 : UIViewInvalidating,
InvalidationType6 : UIViewInvalidating

        public init<InvalidationType1,
InvalidationType2, InvalidationType3,
InvalidationType4, InvalidationType5,
InvalidationType6,
InvalidationType7>(wrappedValue: Value, _
invalidation1: InvalidationType1, _
invalidation2: InvalidationType2, _
invalidation3: InvalidationType3, _
invalidation4: InvalidationType4, _
invalidation5: InvalidationType5, _
invalidation6: InvalidationType6, _
invalidation7: InvalidationType7) where
InvalidationType ==
UIView.Invalidations.Tuple<UIView.Invalid
ations.Tuple<UIView.Invalidations.Tuple<I
nvalidationType1, InvalidationType2>,
UIView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
UIView.Invalidations.Tuple<UIView.Invalid
ations.Tuple<InvalidationType5,
InvalidationType6>, InvalidationType7>>,
InvalidationType1 : UIViewInvalidating,
InvalidationType2 : UIViewInvalidating,
InvalidationType3 : UIViewInvalidating,
InvalidationType4 : UIViewInvalidating,
InvalidationType5 : UIViewInvalidating,
InvalidationType6 : UIViewInvalidating,
InvalidationType7 : UIViewInvalidating

        public init<InvalidationType1,
InvalidationType2, InvalidationType3,
InvalidationType4, InvalidationType5,
InvalidationType6, InvalidationType7,
InvalidationType8>(wrappedValue: Value, _
invalidation1: InvalidationType1, _
invalidation2: InvalidationType2, _
invalidation3: InvalidationType3, _
invalidation4: InvalidationType4, _
invalidation5: InvalidationType5, _
invalidation6: InvalidationType6, _
invalidation7: InvalidationType7, _
invalidation8: InvalidationType8) where
InvalidationType ==
UIView.Invalidations.Tuple<UIView.Invalid
ations.Tuple<UIView.Invalidations.Tuple<I
nvalidationType1, InvalidationType2>,
UIView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
UIView.Invalidations.Tuple<UIView.Invalid
ations.Tuple<InvalidationType5,
InvalidationType6>,
UIView.Invalidations.Tuple<InvalidationTy
pe7, InvalidationType8>>>,
InvalidationType1 : UIViewInvalidating,
InvalidationType2 : UIViewInvalidating,
InvalidationType3 : UIViewInvalidating,
InvalidationType4 : UIViewInvalidating,
InvalidationType5 : UIViewInvalidating,
InvalidationType6 : UIViewInvalidating,
InvalidationType7 : UIViewInvalidating,
InvalidationType8 : UIViewInvalidating

        public init<InvalidationType1,
InvalidationType2, InvalidationType3,
InvalidationType4, InvalidationType5,
InvalidationType6, InvalidationType7,
InvalidationType8,
InvalidationType9>(wrappedValue: Value, _
invalidation1: InvalidationType1, _
invalidation2: InvalidationType2, _
invalidation3: InvalidationType3, _
invalidation4: InvalidationType4, _
invalidation5: InvalidationType5, _
invalidation6: InvalidationType6, _
invalidation7: InvalidationType7, _
invalidation8: InvalidationType8, _
invalidation9: InvalidationType9) where
InvalidationType ==
UIView.Invalidations.Tuple<UIView.Invalid
ations.Tuple<UIView.Invalidations.Tuple<U
IView.Invalidations.Tuple<InvalidationTyp
e1, InvalidationType2>,
UIView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
UIView.Invalidations.Tuple<UIView.Invalid
ations.Tuple<InvalidationType5,
InvalidationType6>,
UIView.Invalidations.Tuple<InvalidationTy
pe7, InvalidationType8>>>,
InvalidationType9>, InvalidationType1 :
```

```
UIViewInvalidating, InvalidationType2 :
UIViewInvalidating, InvalidationType3 :
UIViewInvalidating, InvalidationType4 :
UIViewInvalidating, InvalidationType5 :
UIViewInvalidating, InvalidationType6 :
UIViewInvalidating, InvalidationType7 :
UIViewInvalidating, InvalidationType8 :
UIViewInvalidating, InvalidationType9 :
UIViewInvalidating

        public init<InvalidationType1,
InvalidationType2, InvalidationType3,
InvalidationType4, InvalidationType5,
InvalidationType6, InvalidationType7,
InvalidationType8, InvalidationType9,
InvalidationType10>(wrappedValue: Value,
_ invalidation1: InvalidationType1, _
invalidation2: InvalidationType2, _
invalidation3: InvalidationType3, _
invalidation4: InvalidationType4, _
invalidation5: InvalidationType5, _
invalidation6: InvalidationType6, _
invalidation7: InvalidationType7, _
invalidation8: InvalidationType8, _
invalidation9: InvalidationType9, _
invalidation10: InvalidationType10) where
InvalidationType ==
UIView.Invalidations.Tuple<UIView.Invalid
ations.Tuple<UIView.Invalidations.Tuple<U
IView.Invalidations.Tuple<InvalidationTyp
e1, InvalidationType2>,
UIView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
UIView.Invalidations.Tuple<UIView.Invalid
ations.Tuple<InvalidationType5,
InvalidationType6>,
UIView.Invalidations.Tuple<InvalidationTy
pe7, InvalidationType8>>>,
UIView.Invalidations.Tuple<InvalidationTy
pe9, InvalidationType10>>,
InvalidationType1 : UIViewInvalidating,
InvalidationType2 : UIViewInvalidating,
InvalidationType3 : UIViewInvalidating,
InvalidationType4 : UIViewInvalidating,
InvalidationType5 : UIViewInvalidating,
InvalidationType6 : UIViewInvalidating,
InvalidationType7 : UIViewInvalidating,
InvalidationType8 : UIViewInvalidating,
InvalidationType9 : UIViewInvalidating,
InvalidationType10 : UIViewInvalidating
    }
}

extension UIView {

    @available(swift 5.1)
    @available(iOS 15, tvOS 15, *)
    public enum Invalidations {

        public struct Display :
UIViewInvalidating {

            public init()

            public func invalidate(view:
UIView)
```

```
        }

        public struct Layout :
UIViewInvalidating {

            public init()

            public func invalidate(view:
UIView)
        }

        public struct Constraints :
UIViewInvalidating {

            public init()

            public func invalidate(view:
UIView)
        }

        public struct
IntrinsicContentSize : UIViewInvalidating
{

            public init()

            public func invalidate(view:
UIView)
        }

        public struct Configuration :
UIViewInvalidating {
```

```
            public init()

            public func invalidate(view:
UIView)
        }

        public struct
Tuple<Invalidation1, Invalidation2> :
UIViewInvalidating where Invalidation1 :
UIViewInvalidating, Invalidation2 :
UIViewInvalidating {

            public init(_ invalidation1:
Invalidation1, _ invalidation2:
Invalidation2)

            public func invalidate(view:
UIView)
        }
    }
}

@available(iOS 18.0, *)
@available(visionOS, unavailable)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
@available(macCatalyst, unavailable)
extension UITextFormattingViewController
{

    @MainActor @preconcurrency public var
formattingDescriptor:
UITextFormattingViewController.Formatting
```

```
Descriptor?
}

@available(iOS 18.0, *)
@available(visionOS, unavailable)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
@available(macCatalyst, unavailable)
extension UITextFormattingViewController
{

    public struct FormattingDescriptor {

        public var fonts: [UIFont]

        public var textColors: [UIColor]

        public var lineHeight: Double?

        public var underlinePresent: Bool

        public var strikethroughPresent:
Bool

        public var textAlignments:
Set<UITextFormattingViewController.TextAl
ignment>

        public var textLists:
Set<UITextFormattingViewController.TextLi
st>

        public var highlights:
```

```
Set<UITextFormattingViewController.Highli
ght>

        public var formattingStyleKey:
String?

        public init()

        public init(string:
NSAttributedString, range: NSRange)

        public init(attributes:
[NSAttributedString.Key : Any])

        public init(string: some
AttributedStringProtocol)

        public init(attributes:
AttributeContainer)
    }
}

@available(iOS 18.0, *)
@available(visionOS, unavailable)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
@available(macCatalyst, unavailable)
extension
UITextFormattingViewController.TextAlignm
ent {

    public init(nsTextAlignment:
NSTextAlignment)
```

```
    public var nsTextAlignment:
NSTextAlignment { get }
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIViewController {

    @available(iOS 17.0, tvOS 17.0, *)
    @MainActor @preconcurrency public var
contentUnavailableConfiguration: (any
UIContentConfiguration)?

    @available(iOS 17.0, tvOS 17.0, *)
    @MainActor
@objc(_bridgedContentUnavailableConfigura
tionState) @preconcurrency dynamic open
var contentUnavailableConfigurationState:
UIContentUnavailableConfigurationState {
get }

    @available(iOS 17.0, tvOS 17.0, *)
    @MainActor
@objc(_bridgedUpdateContentUnavailableCon
figurationUsingState:) @preconcurrency
dynamic open func
updateContentUnavailableConfiguration(usi
ng state:
UIContentUnavailableConfigurationState)
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIContentUnavailableView :
```

```
UIContentView {

    @available(iOS 17.0, tvOS 17.0, *)
    @MainActor public func supports(_
configuration: any
UIContentConfiguration) -> Bool

    @available(iOS 17.0, tvOS 17.0, *)
    @MainActor public var configuration:
any UIContentConfiguration

    @available(iOS 17.0, tvOS 17.0, *)
    @MainActor @preconcurrency public
convenience init(configuration:
UIContentUnavailableConfiguration)
}

@available(iOS 16.0, *)
@available(tvOS, unavailable)
extension UICalendarView.Decoration {

    @MainActor @preconcurrency public
static func `default`(color: UIColor? =
nil, size: UICalendarView.DecorationSize
= .medium) -> UICalendarView.Decoration

    @MainActor @preconcurrency public
static func image(_ image: UIImage?,
color: UIColor? = nil, size:
UICalendarView.DecorationSize = .medium)
-> UICalendarView.Decoration
}
```

```
@available(iOS 18.0, *)
@available(visionOS, unavailable)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
@available(macCatalyst, unavailable)
extension
UITextFormattingViewController.Component
{

    public static func component(_
componentKey:
UITextFormattingViewController.ComponentK
ey, _ preferredSize:
UITextFormattingViewController.ComponentS
ize) ->
UITextFormattingViewController.Component
}

@available(iOS 18.0, *)
@available(visionOS, unavailable)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
@available(macCatalyst, unavailable)
extension
UITextFormattingViewController.ComponentG
roup {

    public static func group(_
components:
[UITextFormattingViewController.Component
]) ->
UITextFormattingViewController.ComponentG
roup
```

```
}

extension UIView {

    @available(iOS 17.0, macOS 14.0, tvOS
17.0, watchOS 10.0, *)
    @MainActor @preconcurrency public
class func animate(springDuration
duration: TimeInterval = 0.5, bounce:
CGFloat = 0.0, initialSpringVelocity:
CGFloat = 0.0, delay: TimeInterval = 0.0,
options: UIView.AnimationOptions = [],
animations: () -> Void, completion:
((Bool) -> Void)? = nil)
}

@available(iOS 14.0, tvOS 14.0, *)
extension
UICollectionViewCompositionalLayout {

    @MainActor @preconcurrency public
static func list(using configuration:
UICollectionLayoutListConfiguration) ->
UICollectionViewCompositionalLayout
}

@available(iOS 14.0, tvOS 14.0, *)
extension NSCollectionLayoutSection {

    @MainActor @preconcurrency public
static func list(using configuration:
UICollectionLayoutListConfiguration,
layoutEnvironment: any
NSCollectionLayoutEnvironment) ->
NSCollectionLayoutSection
}

@available(iOS 15.0, tvOS 15.0, *)
extension UIFocusSystem {

    @MainActor @preconcurrency public
class func focusSystem(for environment:
any UIFocusEnvironment) -> UIFocusSystem?
}

extension NSCollectionLayoutGroup {

    @available(iOS, introduced: 16.0,
deprecated: 16.0, renamed:
"horizontal(layoutSize:repeatingSubitem:c
ount:)")
    @MainActor @preconcurrency public
class func horizontalGroup(with size:
NSCollectionLayoutSize, repeatingSubitem
subitem: NSCollectionLayoutItem, count:
Int) -> NSCollectionLayoutGroup

    @available(iOS, introduced: 16.0,
deprecated: 16.0, renamed:
"vertical(layoutSize:repeatingSubitem:cou
nt:)")
    @MainActor @preconcurrency public
class func verticalGroup(with size:
NSCollectionLayoutSize, repeatingSubitem
subitem: NSCollectionLayoutItem, count:
Int) -> NSCollectionLayoutGroup
```

```swift
}

extension UIApplication {

    @available(iOS 16.0, *)
    nonisolated public static let
openNotificationSettingsURLString: String

    @available(iOS 18.2, *)
    @available(visionOS, unavailable)
    @available(macCatalyst, unavailable)
    @available(tvOS, unavailable)
    @available(watchOS, unavailable)
    nonisolated public func isDefault(_
category: UIApplication.Category) throws
-> Bool
}

extension
UIApplication.CategoryDefaultError {

    @available(iOS 18.2, *)
    @available(visionOS, unavailable)
    @available(macCatalyst, unavailable)
    @available(tvOS, unavailable)
    @available(watchOS, unavailable)
    nonisolated public static let
statusLastProvidedDateErrorKey: String

    @available(iOS 18.2, *)
    @available(visionOS, unavailable)
    @available(macCatalyst, unavailable)
    @available(tvOS, unavailable)
    @available(watchOS, unavailable)
    nonisolated public static let
retryAvailableDateErrorKey: String
}

@available(iOS 18.0, *)
extension UIViewController.Transition {

    @available(iOS 18.0, *)
    public static func zoom(options:
UIViewController.Transition.ZoomOptions?
= nil, sourceViewProvider: @escaping (_
context:
UIViewController.Transition.ZoomSourceVie
wProviderContext) -> UIView?) -> Self

    @available(iOS 18.0, *)
    public static var coverVertical: Self
{ get }

    @available(iOS 18.0, *)
    public static var flipHorizontal:
Self { get }

    @available(iOS 18.0, *)
    public static var crossDissolve: Self
{ get }

    @available(iOS 18.0, *)
    public static var partialCurl: Self {
get }
}
```

```
@available(iOS 18.0, *)
extension
UIViewController.Transition.ZoomOptions {

    @available(iOS 18.0, *)
    public var alignmentRectProvider: ((_
context:
UIViewController.Transition.ZoomOptions.A
lignmentRectContext) -> CGRect?)?
}

@available(iOS 18.0, visionOS 2.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
extension UITabBarController.Sidebar {

    @available(iOS 18.0, visionOS 2.0, *)
    @available(tvOS, unavailable)
    @available(watchOS, unavailable)
    public enum ScrollTarget {

        case header

        case footer

        case tab(UITab)
    }

    @MainActor @preconcurrency public var
headerContentConfiguration: (any
UIContentConfiguration)?

    @MainActor @preconcurrency public var
```

```
footerContentConfiguration: (any
UIContentConfiguration)?

    @MainActor @preconcurrency public
func scroll(to target:
UITabBarController.Sidebar.ScrollTarget,
animated: Bool)
}

@available(iOS 18.0, visionOS 2.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
extension UITabSidebarItem {

    @available(iOS 18.0, visionOS 2.0, *)
    @available(tvOS, unavailable)
    @available(watchOS, unavailable)
    public enum Content {

        case tab(UITab)

        case action(UIAction)
    }

    @MainActor @preconcurrency public var
content: UITabSidebarItem.Content { get }

    @MainActor @preconcurrency public var
configurationState:
UICellConfigurationState { get }

    @MainActor @preconcurrency public var
contentConfiguration: any
```

```
UIContentConfiguration

    @MainActor @preconcurrency public var
backgroundConfiguration:
UIBackgroundConfiguration

    @MainActor @preconcurrency public var
accessories: [UICellAccessory]

    @MainActor @preconcurrency public
func defaultContentConfiguration() ->
UIListContentConfiguration

    @MainActor @preconcurrency public
func defaultBackgroundConfiguration() ->
UIBackgroundConfiguration
}

@available(iOS 18.0, visionOS 2.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
extension UITabSidebarItem.Request {

    @MainActor @preconcurrency public var
content: UITabSidebarItem.Content { get }
}

@available(iOS 17.0, *)
@available(tvOS, unavailable)
extension
UIPopoverPresentationControllerSourceItem
{
    public func frame(in referenceView:
UIView) -> CGRect?
}

@available(iOS 15, tvOS 15, watchOS 8, *)
extension AttributeScopes {

    public var uiKit:
AttributeScopes.UIKitAttributes.Type {
get }

    public struct UIKitAttributes :
AttributeScope {

        public let font:
AttributeScopes.UIKitAttributes.FontAttri
bute

        public let paragraphStyle:
AttributeScopes.UIKitAttributes.Paragraph
StyleAttribute

        public let foregroundColor:
AttributeScopes.UIKitAttributes.Foregroun
dColorAttribute

        public let backgroundColor:
AttributeScopes.UIKitAttributes.Backgroun
dColorAttribute

        public let ligature:
AttributeScopes.UIKitAttributes.LigatureA
ttribute
```

```
        public let kern:
AttributeScopes.UIKitAttributes.KernAttri
bute

        public let tracking:
AttributeScopes.UIKitAttributes.TrackingA
ttribute

        public let strikethroughStyle:
AttributeScopes.UIKitAttributes.Strikethr
oughStyleAttribute

        public let underlineStyle:
AttributeScopes.UIKitAttributes.Underline
StyleAttribute

        public let strokeColor:
AttributeScopes.UIKitAttributes.StrokeCol
orAttribute

        public let strokeWidth:
AttributeScopes.UIKitAttributes.StrokeWid
thAttribute

        public let shadow:
AttributeScopes.UIKitAttributes.ShadowAtt
ribute

        public let textEffect:
AttributeScopes.UIKitAttributes.TextEffec
tAttribute
```

```
        public let baselineOffset:
AttributeScopes.UIKitAttributes.BaselineO
ffsetAttribute

        public let underlineColor:
AttributeScopes.UIKitAttributes.Underline
ColorAttribute

        public let strikethroughColor:
AttributeScopes.UIKitAttributes.Strikethr
oughColorAttribute

        @available(watchOS, unavailable)
        public let attachment:
AttributeScopes.UIKitAttributes.Attachmen
tAttribute

        @available(iOS, introduced: 15.0,
deprecated: 100000.0, message: "This
attribute is not supported with TextKit
2")
        public let obliqueness:
AttributeScopes.UIKitAttributes.Obliquene
ssAttribute

        @available(iOS, introduced: 15.0,
deprecated: 100000.0, message: "This
attribute is not supported with TextKit
2")
        public let expansion:
AttributeScopes.UIKitAttributes.Expansion
Attribute
```

```
        @available(iOS 17.0, *)
        @available(tvOS, unavailable)
        public let textItemTag:
AttributeScopes.UIKitAttributes.TextItemT
agAttribute

        @available(macOS 15.0, iOS 18.0,
tvOS 18.0, watchOS 11.0, visionOS 2.0, *)
        public let adaptiveImageGlyph:
AttributeScopes.UIKitAttributes.AdaptiveI
mageGlyphAttribute

        public let accessibility:
AttributeScopes.AccessibilityAttributes

        public let foundation:
AttributeScopes.FoundationAttributes

        @available(iOS 15, tvOS 15,
watchOS 8, macOS 12, *)
        public typealias
DecodingConfiguration =
AttributeScopeCodableConfiguration

        @available(iOS 15, tvOS 15,
watchOS 8, macOS 12, *)
        public typealias
EncodingConfiguration =
AttributeScopeCodableConfiguration
    }
}

@available(iOS 15, tvOS 15, watchOS 8, *)
```

```
extension AttributeDynamicLookup {

    public subscript<T>(dynamicMember
keyPath:
KeyPath<AttributeScopes.UIKitAttributes,
T>) -> T where T : AttributedStringKey {
get }
}

@available(iOS 15, tvOS 15, watchOS 8, *)
extension NSUnderlineStyle : Hashable {
}

@available(macOS 15.0, iOS 18.0, tvOS
18.0, watchOS 11.0, visionOS 2.0, *)
extension
AttributedString.AdaptiveImageGlyph {

    public init(_ nsAdaptiveImageGlyph:
NSAdaptiveImageGlyph)
}

@available(macOS 15.0, iOS 18.0, tvOS
18.0, watchOS 11.0, visionOS 2.0, *)
extension NSAdaptiveImageGlyph {

    public convenience init(_
adaptiveImageGlypth:
AttributedString.AdaptiveImageGlyph)
}

@available(iOS 17.0, visionOS 1.0, *)
@available(tvOS, unavailable)
```

```
@available(watchOS, unavailable)
extension UIHoverStyle {

    @MainActor @preconcurrency public var
effect: any UIHoverEffect

    @MainActor @preconcurrency public var
shape: UIShape?

    @MainActor @preconcurrency public
convenience init(effect: some
UIHoverEffect, shape: UIShape? = nil)

    @MainActor @preconcurrency public
convenience init(shape: UIShape? = nil)
}

@available(iOS 16.0, *)
extension UIPasteControl.Configuration {

    @MainActor @preconcurrency public var
cornerStyle:
UIButton.Configuration.CornerStyle
}

@available(iOS 7.0, *)
extension UIEdgeInsets : Equatable {

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (lhs:
UIEdgeInsets, rhs: UIEdgeInsets) -> Bool
}

@available(iOS 11.0, tvOS 11.0, watchOS
4.0, *)
extension NSDirectionalEdgeInsets :
Equatable {

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (lhs:
NSDirectionalEdgeInsets, rhs:
NSDirectionalEdgeInsets) -> Bool
}
```

```
@available(iOS 7.0, *)
extension UIOffset : Equatable {

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (lhs: UIOffset,
rhs: UIOffset) -> Bool
}

@available(iOS 9.0, *)
extension UIFloatRange : Equatable {

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
```

```
compare.
    public static func == (lhs: 
UIFloatRange, rhs: UIFloatRange) -> Bool
}

@available(iOS 7.0, *)
extension UIEdgeInsets : Codable {

    /// Creates a new instance by 
decoding from the given decoder.
    ///
    /// This initializer throws an error 
if reading from the decoder fails, or
    /// if the data read is corrupted or 
otherwise invalid.
    ///
    /// - Parameter decoder: The decoder 
to read data from.
    public init(from decoder: any 
Decoder) throws

    /// Encodes this value into the given 
encoder.
    ///
    /// If the value fails to encode 
anything, `encoder` will encode an empty
    /// keyed container in its place.
    ///
    /// This function throws an error if 
any values are invalid for the given
    /// encoder's format.
    ///
    /// - Parameter encoder: The encoder
```

```
to write data to.
    public func encode(to encoder: any
Encoder) throws
}

@available(iOS 11.0, tvOS 11.0, watchOS
4.0, *)
extension NSDirectionalEdgeInsets :
Codable {

    /// Creates a new instance by
decoding from the given decoder.
    ///
    /// This initializer throws an error
if reading from the decoder fails, or
    /// if the data read is corrupted or
otherwise invalid.
    ///
    /// - Parameter decoder: The decoder
to read data from.
    public init(from decoder: any
Decoder) throws

    /// Encodes this value into the given
encoder.
    ///
    /// If the value fails to encode
anything, `encoder` will encode an empty
    /// keyed container in its place.
    ///
    /// This function throws an error if
any values are invalid for the given
    /// encoder's format.
    ///
    /// - Parameter encoder: The encoder
to write data to.
    public func encode(to encoder: any
Encoder) throws
}

@available(iOS 7.0, *)
extension UIOffset : Codable {

    /// Creates a new instance by
decoding from the given decoder.
    ///
    /// This initializer throws an error
if reading from the decoder fails, or
    /// if the data read is corrupted or
otherwise invalid.
    ///
    /// - Parameter decoder: The decoder
to read data from.
    public init(from decoder: any
Decoder) throws

    /// Encodes this value into the given
encoder.
    ///
    /// If the value fails to encode
anything, `encoder` will encode an empty
    /// keyed container in its place.
    ///
    /// This function throws an error if
any values are invalid for the given
    /// encoder's format.
    ///
    /// - Parameter encoder: The encoder
to write data to.
    public func encode(to encoder: any
Encoder) throws
}

@available(iOS 7.0, *)
extension UIFloatRange : Codable {

    /// Creates a new instance by
decoding from the given decoder.
    ///
    /// This initializer throws an error
if reading from the decoder fails, or
    /// if the data read is corrupted or
otherwise invalid.
    ///
    /// - Parameter decoder: The decoder
to read data from.
    public init(from decoder: any
Decoder) throws

    /// Encodes this value into the given
encoder.
    ///
    /// If the value fails to encode
anything, `encoder` will encode an empty
    /// keyed container in its place.
    ///
    /// This function throws an error if
any values are invalid for the given
    /// encoder's format.
    ///
    /// - Parameter encoder: The encoder
to write data to.
    public func encode(to encoder: any
Encoder) throws
}

@available(iOS, introduced: 2.0,
deprecated: 8.3, message: "UIActionSheet
is deprecated. Use UIAlertController with
a preferredStyle of
UIAlertControllerStyleActionSheet
instead")
@available(visionOS, unavailable)
extension UIActionSheet {

    @MainActor @preconcurrency public
convenience init(title: String?,
delegate: (any UIActionSheetDelegate)?,
cancelButtonTitle: String?,
destructiveButtonTitle: String?,
otherButtonTitles firstButtonTitle:
String, _ moreButtonTitles: String...)
}

@available(iOS, introduced: 2.0,
deprecated: 9.0, message: "UIAlertView is
deprecated. Use UIAlertController with a
preferredStyle of
UIAlertControllerStyleAlert instead")
@available(visionOS, unavailable)
extension UIAlertView {
    @MainActor @preconcurrency public
convenience init(title: String, message:
String, delegate: (any
UIAlertViewDelegate)?, cancelButtonTitle:
String?, otherButtonTitles
firstButtonTitle: String, _
moreButtonTitles: String...)
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIColor {

    public convenience init(resource:
ColorResource)
}

@available(iOS 2.0, *)
extension UIImage {

    /// Creates an instance initialized
with the given resource name.
    ///
    /// Do not call this initializer
directly. Instead, initialize a variable
or
    /// constant using an image literal.
    required public convenience
init(imageLiteralResourceName name:
String)
}

@available(iOS 17.0, tvOS 17.0, *)
extension UIImage {
    public convenience init(resource:
ImageResource)
}

extension UIFont.TextStyle {

    @available(iOS 11.0, watchOS 4.0,
tvOS 11.0, *)
    public var metrics: UIFontMetrics {
get }
}

@available(iOS 11.0, tvOS 11.0, *)
extension UIFocusEnvironment {

    @MainActor @preconcurrency public
func contains(_ environment: any
UIFocusEnvironment) -> Bool
}

@available(iOS 11.0, tvOS 11.0, *)
extension UIFocusItem {

    @MainActor @preconcurrency public var
isFocused: Bool { get }
}

@available(iOS 11.0, *)
extension UIDragDropSession {

    @available(iOS 11.0, *)
    @MainActor @preconcurrency public
```

```
func canLoadObjects<T>(ofClass: T.Type)
-> Bool where T : _ObjectiveCBridgeable,
T._ObjectiveCType : NSItemProviderReading
}

@available(iOS 11.0, *)
extension UIDropSession {

    @available(iOS 11.0, *)
    @MainActor @preconcurrency public
func loadObjects<T>(ofClass: T.Type,
completion: @escaping ([T]) -> Void) ->
Progress where T : _ObjectiveCBridgeable,
T._ObjectiveCType : NSItemProviderReading
}

@available(iOS 11.0, *)
extension UIPasteConfiguration {

    @available(iOS 11.0, *)
    @MainActor @preconcurrency public
convenience init<T>(forAccepting _:
T.Type) where T : _ObjectiveCBridgeable,
T._ObjectiveCType : NSItemProviderReading

    @available(iOS 11.0, *)
    @MainActor @preconcurrency public
func addTypeIdentifiers<T>(forAccepting
aClass: T.Type) where T :
_ObjectiveCBridgeable,
T._ObjectiveCType : NSItemProviderReading
}
```

```
extension UIPasteboard {

    @available(iOS 11.0, *)
    public func setObjects<T>(_ objects:
[T]) where T : _ObjectiveCBridgeable,
T._ObjectiveCType : NSItemProviderWriting

    @available(iOS 11.0, *)
    public func setObjects<T>(_ objects:
[T], localOnly: Bool, expirationDate:
Date?) where T : _ObjectiveCBridgeable,
T._ObjectiveCType : NSItemProviderWriting
}

extension UIApplicationDelegate {

    @MainActor @preconcurrency public
static func main()
}

extension UIStoryboard {

    @available(iOS 13.0, tvOS 13.0, *)
    @MainActor @preconcurrency public
func
instantiateInitialViewController<ViewCont
roller>(creator: ((NSCoder) ->
ViewController?)? = nil) ->
ViewController? where ViewController :
UIViewController

    @available(iOS 13.0, tvOS 13.0, *)
    @MainActor @preconcurrency public
```

```
func
instantiateViewController<ViewController>
(identifier: String, creator: ((NSCoder)
-> ViewController?)? = nil) ->
ViewController where ViewController :
UIViewController
}

@available(iOS 7.0, *)
extension UIAccessibilityTraits :
OptionSet {

    /// The type of the elements of an
array literal.
    @available(iOS 7.0, *)
    public typealias ArrayLiteralElement
= UIAccessibilityTraits

    /// The element type of the option
set.
    ///
    /// To inherit all the default
implementations from the `OptionSet`
protocol,
    /// the `Element` type must be
`Self`, the default.
    @available(iOS 7.0, *)
    public typealias Element =
UIAccessibilityTraits
}

@available(iOS 7.0, *)
extension UITextDirection {
    public static func storage(_
direction: UITextStorageDirection) ->
UITextDirection

    public static func layout(_
direction: UITextLayoutDirection) ->
UITextDirection
}

@available(iOS 13.0, *)
extension UICommand {

    @MainActor @preconcurrency public
convenience init(title: String = "",
image: UIImage? = nil, action: Selector,
propertyList: Any? = nil, alternates:
[UICommandAlternate] = [],
discoverabilityTitle: String? = nil,
attributes: UIMenuElement.Attributes =
[], state: UIMenuElement.State = .off)

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency public
convenience init(title: String = "",
subtitle: String? = nil, image: UIImage?
= nil, action: Selector, propertyList:
Any? = nil, alternates:
[UICommandAlternate] = [],
discoverabilityTitle: String? = nil,
attributes: UIMenuElement.Attributes =
[], state: UIMenuElement.State = .off)
}
```

```
@available(iOS 17.0, tvOS 17.0, *)
extension UICommand {

    @MainActor @preconcurrency public
convenience init(title: String = "",
subtitle: String? = nil, image: UIImage?
= nil, selectedImage: UIImage? = nil,
action: Selector, propertyList: Any? =
nil, alternates: [UICommandAlternate] =
[], discoverabilityTitle: String? = nil,
attributes: UIMenuElement.Attributes =
[], state: UIMenuElement.State = .off)
}

@available(iOS 13.0, *)
extension UIKeyCommand {

    @MainActor @preconcurrency public
convenience init(title: String = "",
image: UIImage? = nil, action: Selector,
input: String, modifierFlags:
UIKeyModifierFlags = [], propertyList:
Any? = nil, alternates:
[UICommandAlternate] = [],
discoverabilityTitle: String? = nil,
attributes: UIMenuElement.Attributes =
[], state: UIMenuElement.State = .off)
}

@available(iOS 13.0, tvOS 14.0, *)
extension UIAction {
    @MainActor @preconcurrency public
convenience init(title: String = "",
image: UIImage? = nil, identifier:
UIAction.Identifier? = nil,
discoverabilityTitle: String? = nil,
attributes: UIMenuElement.Attributes =
[], state: UIMenuElement.State = .off,
handler: @escaping UIActionHandler)

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency public
convenience init(title: String = "",
subtitle: String? = nil, image: UIImage?
= nil, identifier: UIAction.Identifier? =
nil, discoverabilityTitle: String? = nil,
attributes: UIMenuElement.Attributes =
[], state: UIMenuElement.State = .off,
handler: @escaping UIActionHandler)

    @available(iOS 17.0, tvOS 17.0, *)
    @MainActor @preconcurrency public
convenience init(title: String = "",
subtitle: String? = nil, image: UIImage?
= nil, selectedImage: UIImage? = nil,
identifier: UIAction.Identifier? = nil,
discoverabilityTitle: String? = nil,
attributes: UIMenuElement.Attributes =
[], state: UIMenuElement.State = .off,
handler: @escaping UIActionHandler)
}

@available(iOS 13.0, tvOS 14.0, *)
extension UIMenu {
    @MainActor @preconcurrency public
convenience init(title: String = "",
image: UIImage? = nil, identifier:
UIMenu.Identifier? = nil, options:
UIMenu.Options = [], children:
[UIMenuElement] = [])

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency public
convenience init(title: String = "",
subtitle: String? = nil, image: UIImage?
= nil, identifier: UIMenu.Identifier? =
nil, options: UIMenu.Options = [],
children: [UIMenuElement] = [])

    @available(iOS 16.0, tvOS 16.0, *)
    @MainActor @preconcurrency public
convenience init(title: String = "",
subtitle: String? = nil, image: UIImage?
= nil, identifier: UIMenu.Identifier? =
nil, options: UIMenu.Options = [],
preferredElementSize: UIMenu.ElementSize
= { if #available(iOS 17.0, tvOS 17.0, *)
{ .automatic } else { .large } }(),
children: [UIMenuElement] = [])
}

@available(iOS 13.0, tvOS 14.0, *)
extension UIMenuBuilder {

    @MainActor @preconcurrency public
func command(for action: Selector,
propertyList: Any? = nil) -> UICommand?
}

@available(iOS 13.0, tvOS 13.0, watchOS
6.0, *)
extension UIImage {

    public var baselineOffsetFromBottom:
CGFloat? { get }
}

@available(iOS 16.0, tvOS 16.0, watchOS
9.0, *)
extension UIImage {

    public convenience init?(systemName
name: String, variableValue: Double,
configuration: UIImage.Configuration? =
nil)

    public convenience init?(named name:
String, in bundle: Bundle? = nil,
variableValue: Double, configuration:
UIImage.Configuration? = nil)
}

@available(iOS 15.0, *)
extension
UIWindowScene.ActivationConfiguration {

    public convenience init(userActivity:
NSUserActivity, options:
UIWindowScene.ActivationRequestOptions? =
nil, preview: UITargetedPreview? = nil)
}

@available(iOS 15.0, *)
extension UIWindowScene.ActivationAction
{

    @MainActor @preconcurrency public
convenience init(title: String? = nil,
subtitle: String? = nil, image: UIImage?
= nil, identifier: UIAction.Identifier? =
nil, discoverabilityTitle: String? = nil,
attributes: UIMenuElement.Attributes =
[], alternate: UIAction? = nil, _
configuration: @escaping
UIWindowScene.ActivationAction.Configurat
ionProvider)
}

@available(iOS 13.0, tvOS 17.0, *)
extension UIContextMenuConfiguration {

    @MainActor @preconcurrency public
convenience init(identifier: (any
NSCopying)? = nil, previewProvider:
UIContextMenuContentPreviewProvider? =
nil, actionProvider:
UIContextMenuActionProvider? = nil)
}

@available(iOS 14.0, tvOS 14.0, *)
extension UICollectionViewCell {
    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor
@objc(_bridgedConfigurationState)
@preconcurrency dynamic open var
configurationState:
UICellConfigurationState { get }

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor
@objc(_bridgedUpdateConfigurationUsingSta
te:) @preconcurrency dynamic open func
updateConfiguration(using state:
UICellConfigurationState)

    @available(iOS 15.0, tvOS 15.0, *)
    public typealias
ConfigurationUpdateHandler = (_ cell:
UICollectionViewCell, _ state:
UICellConfigurationState) -> Void

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency public var
configurationUpdateHandler:
UICollectionViewCell.ConfigurationUpdateH
andler?
}

@available(iOS 14.0, tvOS 14.0, *)
extension UITableViewCell {

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor
@objc(_bridgedConfigurationState)
```

```
@preconcurrency dynamic open var
configurationState:
UICellConfigurationState { get }

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor
@objc(_bridgedUpdateConfigurationUsingSta
te:) @preconcurrency dynamic open func
updateConfiguration(using state:
UICellConfigurationState)

    @available(iOS 15.0, tvOS 15.0, *)
    public typealias
ConfigurationUpdateHandler = (_ cell:
UITableViewCell, _ state:
UICellConfigurationState) -> Void

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency public var
configurationUpdateHandler:
UITableViewCell.ConfigurationUpdateHandle
r?
}

@available(iOS 14.0, tvOS 14.0, *)
extension UITableViewHeaderFooterView {

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor
@objc(_bridgedConfigurationState)
@preconcurrency dynamic open var
configurationState:
UIViewConfigurationState { get }
```

```
    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor
@objc(_bridgedUpdateConfigurationUsingSta
te:) @preconcurrency dynamic open func
updateConfiguration(using state:
UIViewConfigurationState)

    @available(iOS 15.0, tvOS 15.0, *)
    public typealias
ConfigurationUpdateHandler = (_
headerFooterView:
UITableViewHeaderFooterView, _ state:
UIViewConfigurationState) -> Void

    @available(iOS 15.0, tvOS 15.0, *)
    @MainActor @preconcurrency public var
configurationUpdateHandler:
UITableViewHeaderFooterView.Configuration
UpdateHandler?
}

@available(iOS 18.0, *)
@available(visionOS, unavailable)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
@available(macCatalyst, unavailable)
extension UITextFormattingViewController
{

    @MainActor public protocol Delegate :
AnyObject {
        @MainActor func
textFormattingViewController(_
viewController:
UITextFormattingViewController,
didChangeValue changeValue:
UITextFormattingViewController.ChangeValu
e)

        @MainActor func
textFormattingViewController(_
viewController:
UITextFormattingViewController,
shouldPresentFontPicker fontPicker:
UIFontPickerViewController) -> Bool

        @MainActor func
textFormattingViewController(_
viewController:
UITextFormattingViewController,
shouldPresentColorPicker colorPicker:
UIColorPickerViewController) -> Bool

        @MainActor func
textFormattingDidFinish(_ viewController:
UITextFormattingViewController)
    }
}

@available(iOS 18.0, *)
@available(visionOS, unavailable)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
@available(macCatalyst, unavailable)
```

```
extension UITextFormattingViewController
{

    @MainActor @preconcurrency weak
public var delegate: (any
UITextFormattingViewController.Delegate)?
}

@available(iOS 17.0, *)
@available(tvOS, unavailable)
extension UITextItem {

    public enum Content {

        case link(URL)

        case
textAttachment(NSTextAttachment)

        case tag(String)
    }

    @MainActor @preconcurrency public var
content: UITextItem.Content { get }
}

@available(iOS 17.0, *)
@available(tvOS, unavailable)
extension UITextItem.MenuConfiguration {

    public enum Preview {

        case `default`
        case view(UIView)
    }

    @MainActor @preconcurrency public
convenience init(preview:
UITextItem.MenuConfiguration.Preview?
= .default, menu: UIMenu)
}

@available(iOS 14.0, tvOS 14.0, *)
extension UICollectionViewCell {

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public var
backgroundConfiguration:
UIBackgroundConfiguration?

    @available(iOS 16.0, tvOS 16.0, *)
    @MainActor @preconcurrency public
func defaultBackgroundConfiguration() ->
UIBackgroundConfiguration
}

@available(iOS 14.0, tvOS 14.0, *)
extension UITableViewCell {

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public var
backgroundConfiguration:
UIBackgroundConfiguration?

    @available(iOS 16.0, tvOS 16.0, *)
    @MainActor @preconcurrency public
func defaultBackgroundConfiguration() ->
UIBackgroundConfiguration
}

@available(iOS 14.0, tvOS 14.0, *)
extension UITableViewHeaderFooterView {

    @available(iOS 14.0, tvOS 14.0, *)
    @MainActor @preconcurrency public var
backgroundConfiguration:
UIBackgroundConfiguration?

    @available(iOS 16.0, tvOS 16.0, *)
    @MainActor @preconcurrency public
func defaultBackgroundConfiguration() ->
UIBackgroundConfiguration
}
```