```
import Accessibility
import AppKit.AppKitDefines
import AppKit.AppKitErrors
import AppKit.NSATSTypesetter
import AppKit.NSAccessibility
import AppKit.NSAccessibilityColor
import AppKit.NSAccessibilityConstants
import AppKit.NSAccessibilityCustomAction
import AppKit.NSAccessibilityCustomRotor
import AppKit.NSAccessibilityElement
import AppKit.NSAccessibilityProtocols
import AppKit.NSActionCell
import AppKit.NSAdaptiveImageGlyph
import AppKit.NSAffineTransform
import AppKit.NSAlert
import AppKit.NSAlignmentFeedbackFilter
import AppKit.NSAnimation
import AppKit.NSAnimationContext
import AppKit.NSAppearance
import AppKit.NSAppleScriptExtensions
import AppKit.NSApplication
import AppKit.NSApplicationScripting
import AppKit.NSArrayController
import AppKit.NSAttributedString
import AppKit.NSBezierPath
import AppKit.NSBitmapImageRep
import AppKit.NSBox
import AppKit.NSBrowser
import AppKit.NSBrowserCell
import AppKit.NSButton
import AppKit.NSButtonCell
import AppKit.NSButtonTouchBarItem
import AppKit.NSCIImageRep
import AppKit.NSCachedImageRep
import AppKit.NSCandidateListTouchBarItem
import AppKit.NSCell
import AppKit.NSClickGestureRecognizer
import AppKit.NSClipView
import AppKit.NSCollectionView
import
AppKit.NSCollectionViewCompositionalLayou
t
import AppKit.NSCollectionViewFlowLayout
import AppKit.NSCollectionViewGridLayout
import AppKit.NSCollectionViewLayout
import
AppKit.NSCollectionViewTransitionLayout
import AppKit.NSColor
import AppKit.NSColorList
import AppKit.NSColorPanel
import AppKit.NSColorPicker
import AppKit.NSColorPickerTouchBarItem
import AppKit.NSColorPicking
import AppKit.NSColorSampler
import AppKit.NSColorSpace
import AppKit.NSColorWell
import AppKit.NSComboBox
import AppKit.NSComboBoxCell
import AppKit.NSComboButton
import AppKit.NSControl
import AppKit.NSController
import AppKit.NSCursor
import AppKit.NSCustomImageRep
import AppKit.NSCustomTouchBarItem
import AppKit.NSDataAsset
import AppKit.NSDatePicker
import AppKit.NSDatePickerCell
import AppKit.NSDictionaryController
import AppKit.NSDiffableDataSource
import AppKit.NSDirection
import AppKit.NSDockTile
import AppKit.NSDocument
import AppKit.NSDocumentController
import AppKit.NSDocumentScripting
import AppKit.NSDragging
import AppKit.NSDraggingItem
import AppKit.NSDraggingSession
import AppKit.NSDrawer
import AppKit.NSEPSImageRep
import AppKit.NSErrors
import AppKit.NSEvent
import AppKit.NSFilePromiseProvider
import AppKit.NSFilePromiseReceiver
import AppKit.NSFileWrapperExtensions
import AppKit.NSFont
import AppKit.NSFontAssetRequest
import AppKit.NSFontCollection
import AppKit.NSFontDescriptor
import AppKit.NSFontManager
import AppKit.NSFontPanel
import AppKit.NSForm
import AppKit.NSFormCell
import AppKit.NSGestureRecognizer
import AppKit.NSGlyphGenerator
import AppKit.NSGlyphInfo
import AppKit.NSGradient
import AppKit.NSGraphics
import AppKit.NSGraphicsContext
import AppKit.NSGridView
import AppKit.NSGroupTouchBarItem
import AppKit.NSHapticFeedback
import AppKit.NSHelpManager
import AppKit.NSImage
import AppKit.NSImageCell
import AppKit.NSImageRep
import AppKit.NSImageView
import AppKit.NSInputManager
import AppKit.NSInputServer
import AppKit.NSInterfaceStyle
import AppKit.NSItemProvider
import AppKit.NSKeyValueBinding
import AppKit.NSLayoutAnchor
import AppKit.NSLayoutConstraint
import AppKit.NSLayoutGuide
import AppKit.NSLayoutManager
import AppKit.NSLevelIndicator
import AppKit.NSLevelIndicatorCell
import
AppKit.NSMagnificationGestureRecognizer
import AppKit.NSMatrix
import
AppKit.NSMediaLibraryBrowserController
import AppKit.NSMenu
import AppKit.NSMenuItem
import AppKit.NSMenuItemBadge
import AppKit.NSMenuItemCell
import AppKit.NSMenuToolbarItem
import AppKit.NSMovie
import AppKit.NSNib
import AppKit.NSNibConnector
import AppKit.NSNibControlConnector
import AppKit.NSNibDeclarations
import AppKit.NSNibLoading
import AppKit.NSNibOutletConnector
import AppKit.NSObjectController
import AppKit.NSOpenGL
import AppKit.NSOpenGLLayer
import AppKit.NSOpenGLView
import AppKit.NSOpenPanel
import AppKit.NSOutlineView
import AppKit.NSPDFImageRep
import AppKit.NSPDFInfo
import AppKit.NSPDFPanel
import AppKit.NSPICTImageRep
import AppKit.NSPageController
import AppKit.NSPageLayout
import AppKit.NSPanGestureRecognizer
import AppKit.NSPanel
import AppKit.NSParagraphStyle
import AppKit.NSPasteboard
import AppKit.NSPasteboardItem
import AppKit.NSPathCell
import AppKit.NSPathComponentCell
import AppKit.NSPathControl
import AppKit.NSPathControlItem
import AppKit.NSPersistentDocument
import AppKit.NSPickerTouchBarItem
import AppKit.NSPopUpButton
import AppKit.NSPopUpButtonCell
import AppKit.NSPopover
import AppKit.NSPopoverTouchBarItem
import AppKit.NSPredicateEditor
import
AppKit.NSPredicateEditorRowTemplate
import AppKit.NSPressGestureRecognizer
import AppKit.NSPressureConfiguration
import
AppKit.NSPreviewRepresentingActivityItem
import AppKit.NSPrintInfo
import AppKit.NSPrintOperation
import AppKit.NSPrintPanel
import AppKit.NSPrinter
import AppKit.NSProgressIndicator
import AppKit.NSResponder
import AppKit.NSRotationGestureRecognizer
import AppKit.NSRuleEditor
import AppKit.NSRulerMarker
import AppKit.NSRulerView
import AppKit.NSRunningApplication
import AppKit.NSSavePanel
import AppKit.NSScreen
import AppKit.NSScrollView
import AppKit.NSScroller
import AppKit.NSScrubber
import AppKit.NSScrubberItemView
import AppKit.NSScrubberLayout
import AppKit.NSSearchField
import AppKit.NSSearchFieldCell
import AppKit.NSSearchToolbarItem
import AppKit.NSSecureTextField
import AppKit.NSSegmentedCell
import AppKit.NSSegmentedControl
import AppKit.NSShadow
import
AppKit.NSSharingCollaborationModeRestrict
ion
import AppKit.NSSharingService
import
```

```
AppKit.NSSharingServicePickerToolbarItem
import
AppKit.NSSharingServicePickerTouchBarItem
import AppKit.NSSlider
import AppKit.NSSliderAccessory
import AppKit.NSSliderCell
import AppKit.NSSliderTouchBarItem
import AppKit.NSSound
import AppKit.NSSpeechRecognizer
import AppKit.NSSpeechSynthesizer
import AppKit.NSSpellChecker
import AppKit.NSSpellProtocol
import AppKit.NSSplitView
import AppKit.NSSplitViewController
import AppKit.NSSplitViewItem
import AppKit.NSStackView
import AppKit.NSStatusBar
import AppKit.NSStatusBarButton
import AppKit.NSStatusItem
import AppKit.NSStepper
import AppKit.NSStepperCell
import AppKit.NSStepperTouchBarItem
import AppKit.NSStoryboard
import AppKit.NSStoryboardSegue
import AppKit.NSStringDrawing
import AppKit.NSSwitch
import AppKit.NSTabView
import AppKit.NSTabViewController
import AppKit.NSTabViewItem
import AppKit.NSTableCellView
import AppKit.NSTableColumn
import AppKit.NSTableHeaderCell
import AppKit.NSTableHeaderView
import AppKit.NSTableRowView
import AppKit.NSTableView
import
AppKit.NSTableViewDiffableDataSource
import AppKit.NSTableViewRowAction
import AppKit.NSText
import AppKit.NSTextAlternatives
import AppKit.NSTextAttachment
import AppKit.NSTextAttachmentCell
import AppKit.NSTextCheckingClient
import AppKit.NSTextCheckingController
import AppKit.NSTextContainer
import AppKit.NSTextContent
import AppKit.NSTextContentManager
import AppKit.NSTextElement
import AppKit.NSTextField
import AppKit.NSTextFieldCell
import AppKit.NSTextFinder
import AppKit.NSTextInputClient
import AppKit.NSTextInputContext
import AppKit.NSTextInsertionIndicator
import AppKit.NSTextLayoutFragment
import AppKit.NSTextLayoutManager
import AppKit.NSTextLineFragment
import AppKit.NSTextList
import AppKit.NSTextListElement
import AppKit.NSTextRange
import AppKit.NSTextSelection
import AppKit.NSTextSelectionNavigation
import AppKit.NSTextStorage
import AppKit.NSTextStorageScripting
import AppKit.NSTextTable
import AppKit.NSTextView
```

```
import
AppKit.NSTextViewportLayoutController
import AppKit.NSTintConfiguration
import
AppKit.NSTitlebarAccessoryViewController
import AppKit.NSTokenField
import AppKit.NSTokenFieldCell
import AppKit.NSToolbar
import AppKit.NSToolbarItem
import AppKit.NSToolbarItemGroup
import AppKit.NSTouch
import AppKit.NSTouchBar
import AppKit.NSTouchBarItem
import AppKit.NSTrackingArea
import
AppKit.NSTrackingSeparatorToolbarItem
import AppKit.NSTreeController
import AppKit.NSTreeNode
import AppKit.NSTypesetter
import AppKit.NSUserActivity
import AppKit.NSUserDefaultsController
import AppKit.NSUserInterfaceCompression
import
AppKit.NSUserInterfaceItemIdentification
import
AppKit.NSUserInterfaceItemSearching
import AppKit.NSUserInterfaceLayout
import AppKit.NSUserInterfaceValidation
import AppKit.NSView
import AppKit.NSViewController
import AppKit.NSVisualEffectView
import AppKit.NSWindow
import AppKit.NSWindowController
import AppKit.NSWindowRestoration
import AppKit.NSWindowScripting
import AppKit.NSWindowTab
import AppKit.NSWindowTabGroup
import AppKit.NSWorkspace
import AppKit.NSWritingToolsCoordinator
import
AppKit.NSWritingToolsCoordinatorAnimation
Parameters
import
AppKit.NSWritingToolsCoordinatorContext
import CoreGraphics
import CoreText
import CoreTransferable
import DeveloperToolsSupport
import Foundation
import OSLog
import OpenGL
import Symbols
import UniformTypeIdentifiers
import _Concurrency
import _StringProcessing
import _SwiftConcurrencyShims

@available(macOS 10.9, *)
public func NSApplicationMain(_ argc:
Int32, _ argv:
UnsafeMutablePointer<UnsafeMutablePointer
<CChar>?>) -> Int32

@available(macOS 10.15.1, *)
open class
NSCollectionViewDiffableDataSource<Sectio
```

```
nIdentifierType, ItemIdentifierType> :
NSObject, NSCollectionViewDataSource
where SectionIdentifierType : Hashable,
ItemIdentifierType : Hashable {

    public typealias ItemProvider =
(NSCollectionView, IndexPath,
ItemIdentifierType) ->
NSCollectionViewItem?

    public typealias
SupplementaryViewProvider =
(NSCollectionView, String, IndexPath) ->
(any NSView & NSCollectionViewElement)?

    public var supplementaryViewProvider:
NSCollectionViewDiffableDataSource<Sectio
nIdentifierType,
ItemIdentifierType>.SupplementaryViewProv
ider?

    public init(collectionView:
NSCollectionView, itemProvider: @escaping
NSCollectionViewDiffableDataSource<Sectio
nIdentifierType,
ItemIdentifierType>.ItemProvider)

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

```
    open func snapshot() ->
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType>

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

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

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

    @MainActor @preconcurrency open func
collectionView(_ collectionView:
NSCollectionView, numberOfItemsInSection
section: Int) -> Int

    @MainActor @preconcurrency open func
collectionView(_ collectionView:
NSCollectionView,
itemForRepresentedObjectAt indexPath:
IndexPath) -> NSCollectionViewItem

    @MainActor @preconcurrency open func
collectionView(_ collectionView:
NSCollectionView,
viewForSupplementaryElementOfKind kind:
NSCollectionView.SupplementaryElementKind
, at indexPath: IndexPath) -> NSView
```

```
    public func description() -> String
}

@available(macOS 10.15.1, *)
public struct
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType> where
SectionIdentifierType : Hashable,
ItemIdentifierType : Hashable {

    public init()

    public var numberOfItems: Int { get }

    public var numberOfSections: Int {
get }

    public var sectionIdentifiers:
[SectionIdentifierType] { get }

    public var itemIdentifiers:
[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])

    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])
}

/// An absolute direction on the
horizontal axis.
@available(macCatalyst 18.0, macOS 15.0,
*)
@frozen public enum NSHorizontalDirection
: Int8, CaseIterable, Codable {

    /// The left direction.
    case left

    /// The right direction.
    case right

    /// An efficient set of horizontal
directions.
    @frozen public struct Set :
OptionSet, Equatable, Hashable {

        /// 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.
        public typealias Element =
NSHorizontalDirection.Set

        /// 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.
        public typealias RawValue = Int8

        /// 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:
NSHorizontalDirection.Set.RawValue

        /// 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:
NSHorizontalDirection.Set.RawValue)

        /// A set containing only the
left direction.
        public static let left:
NSHorizontalDirection.Set

        /// A set containing only the
right direction.
        public static let right:
NSHorizontalDirection.Set

        /// A set containing the all
horizontal directions (left and right).
        public static let all:
NSHorizontalDirection.Set

        /// Creates a set of horizontal
directions containing only the specified
direction.
        public init(_ direction:
NSHorizontalDirection)
```

```
        /// The type of the elements of
an array literal.
        @available(macOS 15.0,
macCatalyst 18.0, *)
        public typealias
ArrayLiteralElement =
NSHorizontalDirection.Set.Element
    }

    /// Creates a new instance with the
specified raw value.
    ///
    /// If there is no value of the type
that corresponds with the specified raw
    /// value, this initializer returns
`nil`. For example:
    ///
    ///     enum PaperSize: String {
    ///         case A4, A5, Letter,
Legal
    ///     }
    ///
    ///     print(PaperSize(rawValue:
"Legal"))
    ///     // Prints
"Optional("PaperSize.Legal")"
    ///
    ///     print(PaperSize(rawValue:
"Tabloid"))
    ///     // Prints "nil"
    ///
    /// - Parameter rawValue: The raw
value to use for the new instance.
```

```
    public init?(rawValue: Int8)

    /// A type that can represent a
collection of all values of this type.
    @available(macOS 15.0, macCatalyst
18.0, *)
    public typealias AllCases =
[NSHorizontalDirection]

    /// 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(macOS 15.0, macCatalyst
18.0, *)
    public typealias RawValue = Int8

    /// A collection of all values of
this type.
    nonisolated public static var
allCases: [NSHorizontalDirection] { get }

    /// 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 var rawValue: Int8 { get }
}

extension NSHorizontalDirection {

    /// Returns the leading direction in
the given user interface layout
direction.
    @available(macOS 15.0, *)
    public static func leading(relativeTo
layoutDirection:
NSUserInterfaceLayoutDirection) ->
NSHorizontalDirection

    /// Returns the trailing direction in
```

```
the given user interface layout
direction.
    @available(macOS 15.0, *)
    public static func
trailing(relativeTo layoutDirection:
NSUserInterfaceLayoutDirection) ->
NSHorizontalDirection
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSHorizontalDirection :
Equatable {
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSHorizontalDirection :
Hashable {
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSHorizontalDirection :
RawRepresentable {
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSHorizontalDirection :
Sendable {
}
```

```
@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSHorizontalDirection :
BitwiseCopyable {
}

extension NSHorizontalDirection.Set {

    @available(macCatalyst 18.0, macOS
15.0, *)
    public func contains(_ member:
NSHorizontalDirection) -> Bool

    @available(macCatalyst 18.0, macOS
15.0, *)
    @discardableResult
    public mutating func insert(_
newMember: NSHorizontalDirection) ->
(inserted: Bool, memberAfterInsert:
NSHorizontalDirection)

    @available(macCatalyst 18.0, macOS
15.0, *)
    @discardableResult
    public mutating func remove(_ member:
NSHorizontalDirection) ->
NSHorizontalDirection?

    @available(macCatalyst 18.0, macOS
15.0, *)
    @discardableResult
    public mutating func update(with
newMember: NSHorizontalDirection) ->
```

```
NSHorizontalDirection?
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSHorizontalDirection.Set :
Sendable {
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSHorizontalDirection.Set :
BitwiseCopyable {
}

/// The items that appear in suggestion
menus.
@available(macOS 15.0, *)
public struct
NSSuggestionItem<SuggestionItemType> {

    /// The value represented by the
receiver.
    /// - Note: Use this to associate an
underlying model object with a suggestion
item. This is useful to refer back to
later if/when the user selects this
suggestion item.
    public var representedValue:
SuggestionItemType

    /// A string to display for a
suggestion menu item. This value should
```

```
be localized.
    /// This value is a non-attributed
string representation of
`attributedTitle`.
    public var title: String

    /// An attributed string to display
for a suggestion menu item. This value
should be localized.
    /// This value is an attributed
string representation of `title`.
    public var attributedTitle:
AttributedString

    /// An optional second string to
display for a suggestion menu item. This
value should be localized.
    /// This value is a non-attributed
string representation of
`attributedSecondaryTitle`.
    public var secondaryTitle: String?

    /// An optional second attributed
string to display for a suggestion menu
item. This value should be localized.
    /// This value is an attributed
string representation of
`secondaryTitle`.
    public var attributedSecondaryTitle:
AttributedString?

    /// An optional tool tip to display
on hover for a suggestion menu item. This
```

```
value should be localized.
    public var toolTip: String?

    /// An optional image to display
before the title. This value should be
localized.
    public var image: NSImage?

    public init(representedValue:
SuggestionItemType, title: String)

    public init(representedValue:
SuggestionItemType, attributedTitle:
AttributedString)
}

@available(macOS 15.0, *)
extension NSSuggestionItem : Equatable
where SuggestionItemType : 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:
```

```
NSSuggestionItem<SuggestionItemType>,
rhs:
NSSuggestionItem<SuggestionItemType>) ->
Bool
}

@available(macOS 15.0, *)
extension NSSuggestionItem : Hashable
where SuggestionItemType : Hashable {

    /// 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 }
}

/// Describes the result of a batch of
suggestion items from a search
@available(macOS 15.0, *)
public struct
NSSuggestionItemResponse<SuggestionItemTy
pe> {

    /// A suggestion item with the same
suggestion item type as the response
    public typealias Item =
NSSuggestionItem<SuggestionItemType>
```

```
    /// A suggestion item section with
the same suggestion item type as the
response
    public typealias ItemSection =
NSSuggestionItemSection<SuggestionItemTyp
e>

    /// The items (organized in sections)
representing the results of the search
request
    public var itemSections:
[NSSuggestionItemResponse<SuggestionItemT
ype>.ItemSection]

    /// Describes the different possible
phases of results
    public enum Phase : Equatable,
Hashable {

        /// The collection of items
represent an intermediate (non-final) set
of results. The user can expect to
potentially see more results in a short
period of time.
        case intermediate

        /// The collection of items
represents a final set of results for the
request. The user can expect these
results to be stable until their search
request changes.
        case final
        /// 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:
NSSuggestionItemResponse<SuggestionItemTy
pe>.Phase, b:
NSSuggestionItemResponse<SuggestionItemTy
pe>.Phase) -> 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 }
    }

    /// Describes the phase of results.
In other words, whether this batch of
items represents an intermediate set of
results--and more are coming, or whether
these results are complete/final.
Defaults to `.final`.
    /// - Note: This controls whether or
not a indeterminate spinner appears by
the control and suggestions menu to
indicate to the user that there may be
any/more/different/updated suggestions
coming.
    /// - Note: Once a final set of
results have been provided, the control
will ignore subsequent provisions until
the search request changes.
    public var phase:
NSSuggestionItemResponse<SuggestionItemTy
pe>.Phase

    /// Describes the possible ways the
highlighted item may be impacted by these
results
    public enum Highlight : Equatable,
Hashable {

        /// The highlighted item is
managed automatically by the control
        /// This is useful in contexts
```

```
where the results, while still relevant,
aren’t an incredibly strong match, or
suggestions are a secondary means of
input, deferring to text the user has
entered themselves.
        case automatic

        /// The first selectable item (if
any) should be highlighted, indicating a
strong match
        /// This is useful in contexts
where suggestions are less for
convenience and are instead expected to
be the primary way for users to interact
with the control.
        case firstSelectableItem

        /// 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:
NSSuggestionItemResponse<SuggestionItemTy
pe>.Highlight, b:
NSSuggestionItemResponse<SuggestionItemTy
pe>.Highlight) -> 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 }
    }

    /// The preferred response that the
control should take when this batch of
results comes in (like whether or not to
highlight the first selectable item).
Defaults to `.automatic`.
    /// - Note: This value is just a hint
to the control on how to actually respond
and may not  affect the highlighted item.
In the case that the user has begun to
already make a selection by using the
up/down arrow keys, or using the mouse,
the actual highlighted item may not be
affected.
```

```
    public var preferredHighlight:
NSSuggestionItemResponse<SuggestionItemTy
pe>.Highlight

    public init(itemSections:
[NSSuggestionItemResponse<SuggestionItemT
ype>.ItemSection])

    public init(items:
[NSSuggestionItemResponse<SuggestionItemT
ype>.Item])

    public init()
}

@available(macOS 15.0, *)
extension NSSuggestionItemResponse :
Equatable where SuggestionItemType :
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 == (a:
```

```
NSSuggestionItemResponse<SuggestionItemTy
pe>, b:
NSSuggestionItemResponse<SuggestionItemTy
pe>) -> Bool
}

@available(macOS 15.0, *)
extension NSSuggestionItemResponse :
Hashable where SuggestionItemType :
Hashable {

    /// 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 }
}

/// Describes a section of suggestions
items in a suggestions menu
@available(macOS 15.0, *)
public struct
NSSuggestionItemSection<SuggestionItemTyp
e> {

    /// A suggestion item with the same
suggestion item type as the section
    public typealias Item =
NSSuggestionItem<SuggestionItemType>

    /// The title of this section of
items, or `nil` to have a untitled
section of items
    public var title: String?

    /// The items that appear in this
section
    public var items:
[NSSuggestionItemSection<SuggestionItemTy
pe>.Item]

    public init(title: String?, items:
[NSSuggestionItemSection<SuggestionItemTy
pe>.Item])

    public init(items:
[NSSuggestionItemSection<SuggestionItemTy
pe>.Item])
}

@available(macOS 15.0, *)
extension NSSuggestionItemSection :
Equatable where SuggestionItemType :
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 == (a:
NSSuggestionItemSection<SuggestionItemTyp
e>, b:
NSSuggestionItemSection<SuggestionItemTyp
e>) -> Bool
}

@available(macOS 15.0, *)
extension NSSuggestionItemSection :
Hashable where SuggestionItemType :
Hashable {

    /// 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(macOS 11.0, *)
open class
NSTableViewDiffableDataSource<SectionIden
```

```
tifierType, ItemIdentifierType> :
NSObject, NSTableViewDataSource where
SectionIdentifierType : Hashable,
ItemIdentifierType : Hashable {

    public typealias CellProvider = (_
tableView: NSTableView, _ tableColumn:
NSTableColumn, _ row: Int, _ identifier:
ItemIdentifierType) -> NSView

    public typealias
SectionHeaderViewProvider = (_ tableView:
NSTableView, _ row: Int, _ identifier:
SectionIdentifierType) -> NSView

    public typealias RowProvider = (_
tableView: NSTableView, _ row: Int, _
identifier: AnyHashable) ->
NSTableRowView

    public var rowViewProvider:
NSTableViewDiffableDataSource<SectionIden
tifierType,
ItemIdentifierType>.RowProvider?

    public var sectionHeaderViewProvider:
NSTableViewDiffableDataSource<SectionIden
tifierType,
ItemIdentifierType>.SectionHeaderViewProv
ider?

    public var defaultRowAnimation:
NSTableView.AnimationOptions
```

```
    public init(tableView: NSTableView,
cellProvider: @escaping
NSTableViewDiffableDataSource<SectionIden
tifierType,
ItemIdentifierType>.CellProvider)

    public func snapshot() ->
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType>

    public func apply(_ snapshot:
NSDiffableDataSourceSnapshot<SectionIdent
ifierType, ItemIdentifierType>,
animatingDifferences: Bool, completion:
(() -> Void)? = nil)

    public func itemIdentifier(forRow
row: Int) -> ItemIdentifierType?

    public func row(forItemIdentifier
identifier: ItemIdentifierType) -> Int?

    public func sectionIdentifier(forRow
row: Int) -> SectionIdentifierType?

    public func row(forSectionIdentifier
identifier: SectionIdentifierType) ->
Int?

    @MainActor @preconcurrency public
func numberOfRows(in tableView:
NSTableView) -> Int
```

```
}

/// A protocol for suggestion delegates
of text fields to conform to in order to
provide text suggestions in response to
the user typing.
@available(macOS 15.0, *)
@MainActor public protocol
NSTextSuggestionsDelegate<SuggestionItemT
ype> : AnyObject {

    /// The type of the
`representedValue` property of the
provided suggestion items
(`NSSuggestionItem`).
    associatedtype SuggestionItemType

    /// A suggestion item with the same
suggestion item type as the delegate.
    typealias Item =
NSSuggestionItem<Self.SuggestionItemType>

    /// A suggestion item section with
the same suggestion item type as the
controller.
    typealias ItemSection =
NSSuggestionItemSection<Self.SuggestionIt
emType>

    /// A suggestion item response with
the same generic type as the delegate.
    typealias ItemResponse =
NSSuggestionItemResponse<Self.SuggestionI
```

```
temType>

    /// Called when the text field’s text
(or tokens) have changed and when the
text field is going to display a new list
of suggestion items.
    ///
    /// Read the contents of `textField`
to determine what suggestions to provide.
For example:
    ///
    ///     func textField(
    ///         _ textField: NSTextField,
    ///         provideUpdatedSuggestions
responseHandler: @escaping
((ItemResponse) -> Void)
    ///     ) {
    ///         // 1. Get the results
(filter).
    ///         let searchString =
textField.stringValue
    ///         let matchedLocations =
self.favoriteLocations.filter({
    ///
$0.matches(searchString: searchString)
    ///         }).prefix(5)
    ///
    ///         // 2. Create item
representations for the results.
    ///         let items =
matchedLocations.map({
    ///
Item(representedValue: $0, title:
```

```
$0.title)
    ///         })
    ///
    ///         // 3. Provide them back
to the text field in a titled section.
    ///         let response =
ItemResponse(itemSections: [
    ///             ItemSection(
    ///                 title:
NSLocalizedString("Favorites", comment:
…),
    ///                 items: items
    ///             ),
    ///         ])
    ///         responseHandler(response)
    ///     }
    ///
    /// If some of the suggestions are
time-consuming to compute, or need to be
fetched from an asynchronous/remote
source, call `responseHandler` an
additional time when that data has been
fetched and processed.
    ///
    ///     func textField(
    ///         _ textField: NSTextField,
    ///         provideUpdatedSuggestions
responseHandler: @escaping
((ItemResponse) -> Void)
    ///     ) {
    ///         // …
    ///
    ///         var response =
ItemResponse(…)
    ///         response.phase
= .intermediate
    ///         responseHandler(response)
    ///
    ///         Task {
    ///             let
suggestedLocations = await
LocationSuggestionsProvider.shared.sugges
tedLocations(for: searchString)
    ///
    ///
response.itemSections.append(ItemSection(
    ///                 title:
NSLocalizedString("Suggested Locations",
comment: …),
    ///                 items:
suggestedLocations.map({
    ///
Item(representedValue: $0, title:
$0.title)
    ///                 })
    ///             ))
    ///             response.phase
= .final
    ///
responseHandler(response)
    ///         }
    ///     }
    ///
    /// - Parameters:
    ///   - textField: The text field
requesting updated suggestions.
```

```
    ///   - responseHandler: A closure to
call with an item response when results
have been gathered and are ready to
display.
    ///
    /// - Note: `responseHandler` must be
called on the main thread. This can be
(and should be for best user experience)
called once synchronously from this
function call to provide immediate
response to the user typing in text. But,
it can optionally be called additional
times later if more results can be
provided asynchronously (such as making a
network request). Note that this closure
is unique each time this function is
called. Invoking a previously-provided
closure will be ignored, therefore not
negatively impacting the results the user
sees by displaying results incongruous to
the text field’s contents. If possible,
it’s a good idea to cancel any long-
running or expensive operations called
for previous search requests when this
function is called again.
    ///
    /// - Note: This function is
automatically called when the text
field’s text or tokens change and the
system determines that new suggestions
are needed. This function may or may not
be called with a delay for debouncing
purposes.
```

```
    @MainActor func textField(_
textField: NSTextField,
provideUpdatedSuggestions
responseHandler: @escaping
((Self.ItemResponse) -> Void))

    /// Returns the full completion text
for a particular item to use when the
item is highlighted or selected.
    ///
    /// This function may be used by the
control to display a preview of the text
that would appear if the highlighted item
is selected. Additionally, the default
implementation of
`textField(_:didSelect:)` uses this
function to inform what changes are made
to the control’s text.
    ///
    /// For example, given a user
interface with a text field and
suggestions menu that looks like:
    ///
    ///
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
    ///     Recipe: ┃ apple│
┃
    ///
┗━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛
    ///
    ///                   │ Applesauce
    ///                   │ Apple juice
    ///                   │ Apple pie
    ///                 └────────────────┘
    ///
    /// where “|” denotes the text
insertion point.
    ///
    /// If this function returns
`"Applesauce"` for the first suggestion
item, when the first suggestion item is
highlighted, the user interface will look
like:
    ///
    ///
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
    ///     Recipe: ┃ apple|sauce|
┃
    ///
┗━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛
    ///
    ///                 ┌────────────────┐
    ///                 │|Applesauce    |│
    ///                 │ Apple juice    │
    ///                 │ Apple pie      │
    ///                 └────────────────┘
    ///
    /// where the text between “|” is
shown as a preview in the field
    ///
    /// Upon selection of that first
suggestion item:
    ///
    ///
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
    ///     Recipe: ┃ Applesauce
┃
```

```
    ///

    ///
    /// - Parameters:
    ///   - textField: The text field
whose suggestions item may be highlighted
or selected.
    ///   - item: The item that may be
highlighted or selected.
    ///
    /// - Returns: The full text to
insert, including the matched partial
word and its potential completion, or
`nil` if no text completion should be
used.
    ///
    /// - Note: This function may or may
not be called when an item is
highlighted, depending on a variety of
factors. Don’t depend upon the timing of
when this function is called. Solely use
it to return the full completion text for
a particular item.
    @MainActor func textField(_
textField: NSTextField, textCompletionFor
item: Self.Item) -> String?

    /// Called when an item in the
suggestions menu has been selected.
    ///
    /// The default implementation
inserts the item’s text completion
(`textField(_:textCompletionFor:)`) into
```

```
the control, replacing its existing text.
Overriding this method allows you to do a
custom behavior instead.
    ///
    /// - Parameters:
    ///   - textField: The text field
whose suggestions item was highlighted.
    ///   - item: The item that was
selected.
    @MainActor func textField(_
textField: NSTextField, didSelect item:
Self.Item)
}

extension NSTextSuggestionsDelegate {

    /// Returns the full completion text
for a particular item to use when the
item is highlighted or selected.
    ///
    /// This function may be used by the
control to display a preview of the text
that would appear if the highlighted item
is selected. Additionally, the default
implementation of
`textField(_:didSelect:)` uses this
function to inform what changes are made
to the control’s text.
    ///
    /// For example, given a user
interface with a text field and
suggestions menu that looks like:
    ///
    ///
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
    ///     Recipe: ┃ apple│
┃
    ///
┗━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛
    ///                 ┌─────────────────────┐
    ///                 │ Applesauce          │
    ///                 │ Apple juice         │
    ///                 │ Apple pie           │
    ///                 └─────────────────────┘
    ///
    /// where “│” denotes the text
insertion point.
    ///
    /// If this function returns
`"Applesauce"` for the first suggestion
item, when the first suggestion item is
highlighted, the user interface will look
like:
    ///
    ///
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
    ///     Recipe: ┃ apple│sauce│
┃
    ///
┗━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛
    ///                 ┌─────────────────────┐
    ///                 │|Applesauce      |   │
    ///                 │ Apple juice         │
    ///                 │ Apple pie           │
    ///                 └─────────────────────┘
    ///
    /// where the text between “|” is shown as a preview in the field
    ///
    /// Upon selection of that first suggestion item:
    ///
    ///
┌──────────────────────────────┐
    ///     Recipe: ┃ Applesauce
┃
    ///
└──────────────────────────────┘
    ///
    /// - Parameters:
    ///   - textField: The text field whose suggestions item may be highlighted or selected.
    ///   - item: The item that may be highlighted or selected.
    ///
    /// - Returns: The full text to insert, including the matched partial word and its potential completion, or `nil` if no text completion should be used.
    ///
    /// - Note: This function may or may not be called when an item is highlighted, depending on a variety of factors. Don’t depend upon the timing of when this function is called. Solely use it to return the full completion text for a particular item.
    @available(macOS 15.0, *)
    @MainActor public func textField(_
textField: NSTextField, textCompletionFor
item: Self.Item) -> String?

    /// Called when an item in the
suggestions menu has been selected.
    ///
    /// The default implementation
inserts the item’s text completion
(`textField(_:textCompletionFor:)`) into
the control, replacing its existing text.
Overriding this method allows you to do a
custom behavior instead.
    ///
    /// - Parameters:
    ///   - textField: The text field
whose suggestions item was highlighted.
    ///   - item: The item that was
selected.
    @available(macOS 15.0, *)
    @MainActor public func textField(_
textField: NSTextField, didSelect item:
Self.Item)
}

extension NSTextSuggestionsDelegate where
Self.SuggestionItemType : Hashable {

    /// Returns a new text suggestions
delegate of the same suggestion item type
with the items and behaviors of the
receiving delegate and `other`
```

```
concatenated.
    /// When the returned delegate is
connected to a text field, all suggestion
items provided from the first suggestions
delegate appear before all those from the
second suggestions delegate, visually
separated by a separator.
    /// - Note: The returned aggregate
text suggestions delegate strongly
retains the given text suggestions
delegate (`other`).
    @available(macOS 15.0, *)
    @MainActor public func appending(_
other: (some
NSTextSuggestionsDelegate<Self.Suggestion
ItemType>)) -> some
NSTextSuggestionsDelegate<Self.Suggestion
ItemType>


    /// Returns a new text suggestions
delegate of a different, but `Hashable`
suggestion item type with the items and
behaviors of the receiving delegate and
`other` concatenated.
    /// When the returned delegate is
connected to a text field, all suggestion
items provided from the first suggestions
delegate appear before all those from the
second suggestions delegate, visually
separated by a separator.
    /// - Note: The returned aggregate
text suggestions delegate strongly
```

```
retains the given text suggestions
delegate (`other`).
    @available(macOS 15.0, *)
    @MainActor public func appending<T>(_
other: (some NSTextSuggestionsDelegate))
-> some
NSTextSuggestionsDelegate<AnyHashable>
where T : Hashable

}

/// A direction on the vertical axis.
@available(macCatalyst 18.0, macOS 15.0,
*)
@frozen public enum NSVerticalDirection :
Int8, CaseIterable, Codable {

    /// The upwards direction.
    case up

    /// The downward direction.
    case down

    /// An efficient set of vertical
directions.
    @frozen public struct Set :
OptionSet, Equatable, Hashable {

        /// 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.
        public typealias Element =
NSVerticalDirection.Set

        /// 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.
        public typealias RawValue = Int8

        /// 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:
NSVerticalDirection.Set.RawValue

        /// 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:
NSVerticalDirection.Set.RawValue)

        /// A set containing only the up
direction.
        public static let up:
NSVerticalDirection.Set

        /// A set containing only the
down direction.
        public static let down:
NSVerticalDirection.Set

        /// A set containing all vertical
directions (up and down)
        public static let all:
NSVerticalDirection.Set

        /// Creates a set of directions
containing only the specified direction.
```

```
        public init(_ direction:
NSVerticalDirection)

        /// The type of the elements of
an array literal.
        @available(macOS 15.0,
macCatalyst 18.0, *)
        public typealias
ArrayLiteralElement =
NSVerticalDirection.Set.Element
    }

    /// Creates a new instance with the
specified raw value.
    ///
    /// If there is no value of the type
that corresponds with the specified raw
    /// value, this initializer returns
`nil`. For example:
    ///
    ///     enum PaperSize: String {
    ///         case A4, A5, Letter,
Legal
    ///     }
    ///
    ///     print(PaperSize(rawValue:
"Legal"))
    ///     // Prints
"Optional("PaperSize.Legal")"
    ///
    ///     print(PaperSize(rawValue:
"Tabloid"))
    ///     // Prints "nil"
    ///
    /// - Parameter rawValue: The raw
value to use for the new instance.
    public init?(rawValue: Int8)

    /// A type that can represent a
collection of all values of this type.
    @available(macOS 15.0, macCatalyst
18.0, *)
    public typealias AllCases =
[NSVerticalDirection]

    /// 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(macOS 15.0, macCatalyst
18.0, *)
    public typealias RawValue = Int8

    /// A collection of all values of
this type.
    nonisolated public static var
allCases: [NSVerticalDirection] { get }
```

```
    /// 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 var rawValue: Int8 { get }
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSVerticalDirection : Equatable
{
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSVerticalDirection : Hashable
```

```
{
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSVerticalDirection :
RawRepresentable {
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSVerticalDirection : Sendable
{
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSVerticalDirection :
BitwiseCopyable {
}

extension NSVerticalDirection.Set {

    @available(macCatalyst 18.0, macOS
15.0, *)
    public func contains(_ member:
NSVerticalDirection) -> Bool

    @available(macCatalyst 18.0, macOS
15.0, *)
    @discardableResult
    public mutating func insert(_
newMember: NSVerticalDirection) ->
```

```
(inserted: Bool, memberAfterInsert:
NSVerticalDirection)

    @available(macCatalyst 18.0, macOS
15.0, *)
    @discardableResult
    public mutating func remove(_ member:
NSVerticalDirection) ->
NSVerticalDirection?

    @available(macCatalyst 18.0, macOS
15.0, *)
    @discardableResult
    public mutating func update(with
newMember: NSVerticalDirection) ->
NSVerticalDirection?
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSVerticalDirection.Set :
Sendable {
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSVerticalDirection.Set :
BitwiseCopyable {
}

@available(swift 5.1)
@available(macOS 12, *)
public protocol NSViewInvalidating {
    func invalidate(view: NSView)
}

@available(swift 5.1)
@available(macOS 12, *)
extension NSViewInvalidating where Self
== NSView.Invalidations.Display {

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

@available(swift 5.1)
@available(macOS 12, *)
extension NSViewInvalidating where Self
== NSView.Invalidations.Layout {

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

@available(swift 5.1)
@available(macOS 12, *)
extension NSViewInvalidating where Self
== NSView.Invalidations.Constraints {

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

@available(swift 5.1)
@available(macOS 12, *)
```

```
extension NSViewInvalidating where Self
==
NSView.Invalidations.IntrinsicContentSize
{

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

@available(swift 5.1)
@available(macOS 12, *)
extension NSViewInvalidating where Self
== NSView.Invalidations.RestorableState {

    public static var restorableState:
NSView.Invalidations.RestorableState {
get }
}

/// Preview an NSView.
///
/// - Parameters:
///   - name: Optional display name for
the preview, which will appear in the
canvas.
///   - traits: Optional list of traits
customizing the appearance of the
preview.
///   - body: A closure producing an
NSView.
@available(macOS 14.0, *)
```

```
@freestanding(declaration) public macro
Preview(_ name: String? = nil, traits:
PreviewTrait<Preview.ViewTraits>...,
@PreviewMacroBodyBuilder<NSView> body:
@escaping @MainActor () -> NSView) =
#externalMacro(module: "PreviewsMacros",
type: "KitViewMacro")

/// Preview an NSViewController.
///
/// - Parameters:
///   - name: Optional display name for
the preview, which will appear in the
canvas.
///   - traits: Optional list of traits
customizing the appearance of the
preview.
///   - body: A closure producing an
NSViewController.
@available(macOS 14.0, *)
@freestanding(declaration) public macro
Preview(_ name: String? = nil, traits:
PreviewTrait<Preview.ViewTraits>...,
@PreviewMacroBodyBuilder<NSViewController
> body: @escaping @MainActor () ->
NSViewController) =
#externalMacro(module: "PreviewsMacros",
type: "KitViewMacro")

@available(macOS 14.0, *)
extension NSShadow : @unchecked Sendable
{
}
```

```
extension NSView {

    @available(swift 5.1)
    @available(macOS 12, *)
    @propertyWrapper public struct
Invalidating<Value, InvalidationType>
where Value : Equatable, InvalidationType
: NSViewInvalidating {

        public init(wrappedValue: Value,
_ invalidation: InvalidationType)

        public init<InvalidationType1,
InvalidationType2>(wrappedValue: Value, _
invalidation1: InvalidationType1, _
invalidation2: InvalidationType2) where
InvalidationType ==
NSView.Invalidations.Tuple<InvalidationTy
pe1, InvalidationType2>,
InvalidationType1 : NSViewInvalidating,
InvalidationType2 : NSViewInvalidating

        public init<InvalidationType1,
InvalidationType2,
InvalidationType3>(wrappedValue: Value, _
invalidation1: InvalidationType1, _
invalidation2: InvalidationType2, _
invalidation3: InvalidationType3) where
InvalidationType ==
NSView.Invalidations.Tuple<NSView.Invalid
ations.Tuple<InvalidationType1,
InvalidationType2>, InvalidationType3>,
InvalidationType1 : NSViewInvalidating,
InvalidationType2 : NSViewInvalidating,
InvalidationType3 : NSViewInvalidating

        public init<InvalidationType1,
InvalidationType2, InvalidationType3,
InvalidationType4>(wrappedValue: Value, _
invalidation1: InvalidationType1, _
invalidation2: InvalidationType2, _
invalidation3: InvalidationType3, _
invalidation4: InvalidationType4) where
InvalidationType ==
NSView.Invalidations.Tuple<NSView.Invalid
ations.Tuple<InvalidationType1,
InvalidationType2>,
NSView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
InvalidationType1 : NSViewInvalidating,
InvalidationType2 : NSViewInvalidating,
InvalidationType3 : NSViewInvalidating,
InvalidationType4 : NSViewInvalidating

        public init<InvalidationType1,
InvalidationType2, InvalidationType3,
InvalidationType4,
InvalidationType5>(wrappedValue: Value, _
invalidation1: InvalidationType1, _
invalidation2: InvalidationType2, _
invalidation3: InvalidationType3, _
invalidation4: InvalidationType4, _
invalidation5: InvalidationType5) where
InvalidationType ==
NSView.Invalidations.Tuple<NSView.Invalid
```

```
ations.Tuple<NSView.Invalidations.Tuple<I
nvalidationType1, InvalidationType2>,
NSView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
InvalidationType5>, InvalidationType1 :
NSViewInvalidating, InvalidationType2 :
NSViewInvalidating, InvalidationType3 :
NSViewInvalidating, InvalidationType4 :
NSViewInvalidating, InvalidationType5 :
NSViewInvalidating

        public init<InvalidationType1,
InvalidationType2, InvalidationType3,
InvalidationType4, InvalidationType5,
InvalidationType6>(wrappedValue: Value, _
invalidation1: InvalidationType1, _
invalidation2: InvalidationType2, _
invalidation3: InvalidationType3, _
invalidation4: InvalidationType4, _
invalidation5: InvalidationType5, _
invalidation6: InvalidationType6) where
InvalidationType ==
NSView.Invalidations.Tuple<NSView.Invalid
ations.Tuple<NSView.Invalidations.Tuple<I
nvalidationType1, InvalidationType2>,
NSView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
NSView.Invalidations.Tuple<InvalidationTy
pe5, InvalidationType6>>,
InvalidationType1 : NSViewInvalidating,
InvalidationType2 : NSViewInvalidating,
InvalidationType3 : NSViewInvalidating,
InvalidationType4 : NSViewInvalidating,
InvalidationType5 : NSViewInvalidating,
InvalidationType6 : NSViewInvalidating

        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 ==
NSView.Invalidations.Tuple<NSView.Invalid
ations.Tuple<NSView.Invalidations.Tuple<I
nvalidationType1, InvalidationType2>,
NSView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
NSView.Invalidations.Tuple<NSView.Invalid
ations.Tuple<InvalidationType5,
InvalidationType6>, InvalidationType7>>,
InvalidationType1 : NSViewInvalidating,
InvalidationType2 : NSViewInvalidating,
InvalidationType3 : NSViewInvalidating,
InvalidationType4 : NSViewInvalidating,
InvalidationType5 : NSViewInvalidating,
InvalidationType6 : NSViewInvalidating,
InvalidationType7 : NSViewInvalidating

        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 ==
NSView.Invalidations.Tuple<NSView.Invalid
ations.Tuple<NSView.Invalidations.Tuple<I
nvalidationType1, InvalidationType2>,
NSView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
NSView.Invalidations.Tuple<NSView.Invalid
ations.Tuple<InvalidationType5,
InvalidationType6>,
NSView.Invalidations.Tuple<InvalidationTy
pe7, InvalidationType8>>>,
InvalidationType1 : NSViewInvalidating,
InvalidationType2 : NSViewInvalidating,
InvalidationType3 : NSViewInvalidating,
InvalidationType4 : NSViewInvalidating,
InvalidationType5 : NSViewInvalidating,
InvalidationType6 : NSViewInvalidating,
InvalidationType7 : NSViewInvalidating,
InvalidationType8 : NSViewInvalidating

        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 ==
NSView.Invalidations.Tuple<NSView.Invalid
ations.Tuple<NSView.Invalidations.Tuple<N
SView.Invalidations.Tuple<InvalidationTyp
e1, InvalidationType2>,
NSView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
NSView.Invalidations.Tuple<NSView.Invalid
ations.Tuple<InvalidationType5,
InvalidationType6>,
NSView.Invalidations.Tuple<InvalidationTy
pe7, InvalidationType8>>>,
InvalidationType9>, InvalidationType1 :
NSViewInvalidating, InvalidationType2 :
NSViewInvalidating, InvalidationType3 :
NSViewInvalidating, InvalidationType4 :
NSViewInvalidating, InvalidationType5 :
NSViewInvalidating, InvalidationType6 :
NSViewInvalidating, InvalidationType7 :
```

```
NSViewInvalidating, InvalidationType8 :
NSViewInvalidating, InvalidationType9 :
NSViewInvalidating

        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 ==
NSView.Invalidations.Tuple<NSView.Invalid
ations.Tuple<NSView.Invalidations.Tuple<N
SView.Invalidations.Tuple<InvalidationTyp
e1, InvalidationType2>,
NSView.Invalidations.Tuple<InvalidationTy
pe3, InvalidationType4>>,
NSView.Invalidations.Tuple<NSView.Invalid
ations.Tuple<InvalidationType5,
InvalidationType6>,
NSView.Invalidations.Tuple<InvalidationTy
pe7, InvalidationType8>>>,
NSView.Invalidations.Tuple<InvalidationTy
```

```
pe9, InvalidationType10>>,
InvalidationType1 : NSViewInvalidating,
InvalidationType2 : NSViewInvalidating,
InvalidationType3 : NSViewInvalidating,
InvalidationType4 : NSViewInvalidating,
InvalidationType5 : NSViewInvalidating,
InvalidationType6 : NSViewInvalidating,
InvalidationType7 : NSViewInvalidating,
InvalidationType8 : NSViewInvalidating,
InvalidationType9 : NSViewInvalidating,
InvalidationType10 : NSViewInvalidating
    }
}

extension NSView {

    @available(swift 5.1)
    @available(macOS 12, *)
    public enum Invalidations {

        public struct Display :
NSViewInvalidating {

            public init()

            public func invalidate(view:
NSView)
        }

        public struct Layout :
NSViewInvalidating {

            public init()
            public func invalidate(view:
NSView)
        }

        public struct Constraints :
NSViewInvalidating {

            public init()

            public func invalidate(view:
NSView)
        }

        public struct
IntrinsicContentSize : NSViewInvalidating
{

            public init()

            public func invalidate(view:
NSView)
        }

        public struct RestorableState :
NSViewInvalidating {

            public init()

            public func invalidate(view:
NSView)
        }
        public struct
Tuple<Invalidation1, Invalidation2> :
NSViewInvalidating where Invalidation1 :
NSViewInvalidating, Invalidation2 :
NSViewInvalidating {

            public init(_ invalidation1:
Invalidation1, _ invalidation2:
Invalidation2)

            public func invalidate(view:
NSView)
        }
    }
}

extension NSPopUpButton {

    /// Creates a standard pull-down
button with a title, optional image, and
menu.
    ///
    /// Pull-down buttons created using
this method have the `usesItemFromMenu`
property set to `false`.
    ///
    /// - Parameters:
    ///   - title: The localized title
string that is displayed on the button.
    ///   - image: The icon that is
displayed on the button.
    ///   - pullDownMenu: The pull-down
menu to present when interacting with the
```

```
button.
    /// - Returns: An initialized pull-
down button object.
    @available(macOS 10.10, *)
    @backDeployed(before: macOS 15.0)
    @MainActor @preconcurrency public
convenience init(title: String, image:
NSImage? = nil, pullDownMenu: NSMenu)

    /// Creates a standard pull-down
button with a title, optional image, and
menu.
    ///
    /// Pull-down buttons created using
this method have the `usesItemFromMenu`
property set to `false`.
    ///
    /// - Parameters:
    ///   - image: The icon that is
displayed on the button.
    ///   - pullDownMenu: The pull-down
menu to present when interacting with the
button.
    /// - Returns: An initialized pull-
down button object.
    @available(macOS 10.10, *)
    @backDeployed(before: macOS 15.0)
    @MainActor @preconcurrency public
convenience init(image: NSImage,
pullDownMenu: NSMenu)

    /// Creates a standard pop-up button
with a menu, target, and action.
```

```
    ///
    /// If `menu` is non-empty, the pop-
up button uses the first item for its
initial selection.
    ///
    /// - Parameters:
    ///   - popUpMenu: A menu presented
by the pop-up button, containing items
that the user can choose between.
    ///   - target: The target object
that receives action messages from the
control.
    ///   - action: The action message
sent by the control.
    /// - Returns: An initialized pop-up
button object.
    @available(macOS 10.10, *)
    @backDeployed(before: macOS 15.0)
    @MainActor @preconcurrency public
convenience init(popUpMenu: NSMenu,
target: AnyObject?, action: Selector?)
}

extension NSApplicationDelegate {

    public static func main()
}

extension NSApplication {

    @available(swift 4)
    @available(macOS 10.9, *)
    @MainActor @preconcurrency public
```

```
static func loadApplication()
}

@available(macOS 10.9, *)
extension NSAppKitVersion {

    @available(*, deprecated, renamed:
"macOS10_14")
    public static var number10_14:
NSAppKitVersion { get }

    @available(*, deprecated, renamed:
"macOS10_14_1")
    public static var number10_14_1:
NSAppKitVersion { get }

    @available(*, deprecated, renamed:
"macOS10_14_2")
    public static var number10_14_2:
NSAppKitVersion { get }

    @available(*, deprecated, renamed:
"macOS10_14_3")
    public static var number10_14_3:
NSAppKitVersion { get }

    @available(*, deprecated, renamed:
"macOS10_14_4")
    public static var number10_14_4:
NSAppKitVersion { get }

    @available(*, deprecated, renamed:
"macOS10_14_5")
```

```
    public static var number10_14_5:
NSAppKitVersion { get }

    @available(*, deprecated, renamed:
"macOS10_15")
    public static var number10_15:
NSAppKitVersion { get }
}

@available(macOS 14.0, *)
extension NSImage {

    /// Initialize a `NSImage` with an
image resource.
    public convenience init(resource:
ImageResource)
}

@available(macOS 10.9, *)
extension NSImage {

    /// 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.
    @nonobjc required public convenience
init(imageLiteralResourceName name:
String)
}
```

```
@available(macOS 14.0, *)
extension NSMenuItemBadge {

    /// The string representation of the
badge as it would appear when
    /// the badge is displayed.
    @objc dynamic public var stringValue:
String? { get }
}

@available(macOS 15.0, *)
extension NSMenuItemBadge : 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.
    required public convenience 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(macOS 12, *)
extension AttributeScopes {

    public var appKit:
AttributeScopes.AppKitAttributes.Type {
get }

    public struct AppKitAttributes :
AttributeScope {

        public let font:
AttributeScopes.AppKitAttributes.FontAttr
ibute

        public let paragraphStyle:
AttributeScopes.AppKitAttributes.Paragrap
hStyleAttribute

        public let foregroundColor:
AttributeScopes.AppKitAttributes.Foregrou
ndColorAttribute

        public let backgroundColor:
```

```
AttributeScopes.AppKitAttributes.Backgrou
ndColorAttribute

        public let ligature:
AttributeScopes.AppKitAttributes.Ligature
Attribute

        public let kern:
AttributeScopes.AppKitAttributes.KernAttr
ibute

        public let tracking:
AttributeScopes.AppKitAttributes.Tracking
Attribute

        public let strikethroughStyle:
AttributeScopes.AppKitAttributes.Striketh
roughStyleAttribute

        public let underlineStyle:
AttributeScopes.AppKitAttributes.Underlin
eStyleAttribute

        public let strokeColor:
AttributeScopes.AppKitAttributes.StrokeCo
lorAttribute

        public let strokeWidth:
AttributeScopes.AppKitAttributes.StrokeWi
dthAttribute

        public let shadow:
AttributeScopes.AppKitAttributes.ShadowAt
```

```
tribute

        public let textEffect:
AttributeScopes.AppKitAttributes.TextEffe
ctAttribute

        public let attachment:
AttributeScopes.AppKitAttributes.Attachme
ntAttribute

        public let baselineOffset:
AttributeScopes.AppKitAttributes.Baseline
OffsetAttribute

        public let underlineColor:
AttributeScopes.AppKitAttributes.Underlin
eColorAttribute

        public let strikethroughColor:
AttributeScopes.AppKitAttributes.Striketh
roughColorAttribute

        @available(macOS, introduced:
12.0, deprecated: 100000.0, message:
"This attribute is not supported with
TextKit 2")
        public let obliqueness:
AttributeScopes.AppKitAttributes.Obliquen
essAttribute

        @available(macOS, introduced:
12.0, deprecated: 100000.0, message:
"This attribute is not supported with
TextKit 2")
        public let expansion:
AttributeScopes.AppKitAttributes.Expansio
nAttribute

        public let toolTip:
AttributeScopes.AppKitAttributes.ToolTipA
ttribute

        public let markedClauseSegment:
AttributeScopes.AppKitAttributes.MarkedCl
auseSegmentAttribute

        public let superscript:
AttributeScopes.AppKitAttributes.Superscr
iptAttribute

        public let textAlternatives:
AttributeScopes.AppKitAttributes.TextAlte
rnativesAttribute

        public let glyphInfo:
AttributeScopes.AppKitAttributes.GlyphInf
oAttribute

        public let cursor:
AttributeScopes.AppKitAttributes.CursorAt
tribute

        @available(macOS 15.0, iOS 18.0,
tvOS 18.0, watchOS 11.0, visionOS 2.0, *)
        public let adaptiveImageGlyph:
AttributeScopes.AppKitAttributes.Adaptive
```

```
ImageGlyphAttribute

        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(macOS 12, *)
extension AttributeDynamicLookup {

    public subscript<T>(dynamicMember
keyPath:
KeyPath<AttributeScopes.AppKitAttributes,
T>) -> T where T : AttributedStringKey {
get }
}

@available(macOS 12, *)
```

```
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(macOS 14.0, *)
extension NSColor {

    /// Initialize a `NSColor` with a
color resource.
    public convenience init(resource:
ColorResource)
}

@available(macOS 10.9, *)
extension IndexPath {
    /// Initialize for use with
`NSCollectionView`.
    public init(item: Int, section: Int)

    /// The item of this index path, when
used with `NSCollectionView`.
    ///
    /// - precondition: The index path
must have exactly two elements.
    public var item: Int

    /// The section of this index path,
when used with `NSCollectionView`.
    ///
    /// - precondition: The index path
must have exactly two elements.
    public var section: Int
}

@available(macOS 10.9, *)
extension URLResourceValues {

    /// All thumbnails as a single
`NSImage`
    @available(macOS, introduced: 10.10,
deprecated: 12.0, message: "Use the
QuickLookThumbnailing framework and
extension point instead")
    public var thumbnail: NSImage? {
get }

    /// The color of the assigned label
    public var labelColor: NSColor? { get
}

    /// The icon normally displayed for
the resource
    public var effectiveIcon: AnyObject?
{ get }

    /// The custom icon stored with the
resource, or `nil` if the resource has no
custom icon. Currently not implemented.
    public var customIcon: NSImage? { get
}

    /// A dictionary of `NSImage` objects
keyed by size
    @available(macOS, introduced: 10.10,
deprecated: 12.0, message: "Use the
QuickLookThumbnailing framework and
extension point instead")
    public var thumbnailDictionary:
[URLThumbnailDictionaryItem : NSImage]? {
get }
}

extension NSScreen {

    @available(macOS 12, *)
    public var auxiliaryTopLeftArea:
NSRect? { get }

    @available(macOS 12, *)
    public var auxiliaryTopRightArea:
NSRect? { get }
```

```
}

extension NSImageView {

    /// Adds a symbol effect to the image
view, with options and animation.
    @available(macOS 14.0, *)
    @MainActor @preconcurrency public
func addSymbolEffect(_ effect: some
DiscreteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true)

    /// Adds a symbol effect to the image
view, with options and animation.
    @available(macOS 14.0, *)
    @MainActor @preconcurrency public
func addSymbolEffect(_ effect: some
IndefiniteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true)

    /// Adds a symbol effect to the image
view, with options and animation.
    @available(macOS 14.0, *)
    @MainActor @preconcurrency public
func addSymbolEffect(_ effect: some
DiscreteSymbolEffect &
IndefiniteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true)

    /// Removes from the image view the
```

```
symbol effect matching the type of effect
passed in, with options and animation.
    @available(macOS 14.0, *)
    @MainActor @preconcurrency public
func removeSymbolEffect(ofType effect:
some DiscreteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true)

    /// Removes from the image view the
symbol effect matching the type of effect
passed in, with options and animation.
    @available(macOS 14.0, *)
    @MainActor @preconcurrency public
func removeSymbolEffect(ofType effect:
some IndefiniteSymbolEffect &
SymbolEffect, options:
SymbolEffectOptions = .default, animated:
Bool = true)

    /// Removes from the image view the
symbol effect matching the type of effect
passed in, with options and animation.
    @available(macOS 14.0, *)
    @MainActor @preconcurrency public
func removeSymbolEffect(ofType effect:
some DiscreteSymbolEffect &
IndefiniteSymbolEffect & SymbolEffect,
options: SymbolEffectOptions = .default,
animated: Bool = true)

    /// Removes all symbol effects from
the image view, with options and
```

```
animation.
    @available(macOS 14.0, *)
    @MainActor @preconcurrency public
func removeAllSymbolEffects(options:
SymbolEffectOptions = .default, animated:
Bool = true)

    /// Sets the symbol image on the
image view using a symbol content
transition effect and options.
    /// Passing in a non-symbol image
will result in undefined behavior.
    @available(macOS 14.0, *)
    @MainActor @preconcurrency public
func setSymbolImage(_ image: NSImage,
contentTransition: some
ContentTransitionSymbolEffect &
SymbolEffect, options:
SymbolEffectOptions = .default)
}

@available(macOS 14.0, *)
extension NSImage : Transferable {

    /// The representation used to import
and export the item.
    ///
    /// A ``transferRepresentation`` can
contain multiple representations
    /// for different content types.
    public static var
transferRepresentation: some
TransferRepresentation { get }
```

```
    /// The type of the representation
used to import and export the item.
    ///
    /// Swift infers this type from the
return value of the
    /// ``transferRepresentation``
property.
    @available(macOS 14.0, *)
    public typealias Representation =
some TransferRepresentation
}

@available(macOS 14.0, *)
extension NSSound : Transferable {

    /// The representation used to import
and export the item.
    ///
    /// A ``transferRepresentation`` can
contain multiple representations
    /// for different content types.
    public static var
transferRepresentation: some
TransferRepresentation { get }

    /// The type of the representation
used to import and export the item.
    ///
    /// Swift infers this type from the
return value of the
    /// ``transferRepresentation``
property.
```

```
    @available(macOS 14.0, *)
    public typealias Representation =
some TransferRepresentation
}

extension NSTextField {

    /// The delegate that provides text
suggestions for the receiving text field
and responds to the user highlighting and
selecting items.
    @available(macOS 15.0, *)
    @MainActor @preconcurrency weak
public var suggestionsDelegate: (any
NSTextSuggestionsDelegate)?
}

@available(macOS 10.9, *)
extension CGRect {

    /// Fills this rect in the current
NSGraphicsContext in the context's fill
    /// color.
    /// The compositing operation of the
fill defaults to the context's
    /// compositing operation, not
necessarily using `.copy` like
`NSRectFill()`.
    /// - precondition: There must be a
set current NSGraphicsContext.
    @available(swift 4)
    public func fill(using operation:
NSCompositingOperation =
NSGraphicsContext.current?.compositingOpe
ration ?? .sourceOver)

    /// Draws a frame around the inside
of this rect in the current
    /// NSGraphicsContext in the
context's fill color
    /// The compositing operation of the
fill defaults to the context's
    /// compositing operation, not
necessarily using `.copy` like
`NSFrameRect()`.
    /// - precondition: There must be a
set current NSGraphicsContext.
    @available(swift 4)
    public func frame(withWidth width:
CGFloat = 1.0, using operation:
NSCompositingOperation =
NSGraphicsContext.current?.compositingOpe
ration ?? .sourceOver)

    /// Modifies the current graphics
context clipping path by intersecting it
    /// with this rect.
    /// This permanently modifies the
graphics state, so the current state
should
    /// be saved beforehand and restored
afterwards.
    /// - precondition: There must be a
set current NSGraphicsContext.
    @available(swift 4)
    public func clip()
```

```
}

@available(macOS 10.9, *)
extension Sequence where Self.Element ==
CGRect {

    /// Fills this list of rects in the
current NSGraphicsContext in the
context's
    /// fill color.
    /// The compositing operation of the
fill defaults to the context's
    /// compositing operation, not
necessarily using `.copy` like
`NSRectFill()`.
    /// - precondition: There must be a
set current NSGraphicsContext.
    @available(swift 4)
    public func fill(using operation:
NSCompositingOperation =
NSGraphicsContext.current?.compositingOpe
ration ?? .sourceOver)

    /// Modifies the current graphics
context clipping path by intersecting it
    /// with the graphical union of this
list of rects
    /// This permanently modifies the
graphics state, so the current state
should
    /// be saved beforehand and restored
afterwards.
    /// - precondition: There must be a
```

```
set current NSGraphicsContext.
    @available(swift 4)
    public func clip()
}

@available(macOS 10.9, *)
extension Sequence where Self.Element ==
(CGRect, NSColor) {

    /// Fills this list of rects in the
current NSGraphicsContext with that
rect's
    /// associated color
    /// The compositing operation of the
fill defaults to the context's
    /// compositing operation, not
necessarily using `.copy` like
`NSRectFill()`.
    /// - precondition: There must be a
set current NSGraphicsContext.
    @available(swift 4)
    public func fill(using operation:
NSCompositingOperation =
NSGraphicsContext.current?.compositingOpe
ration ?? .sourceOver)
}

@available(macOS 10.9, *)
extension Sequence where Self.Element ==
(CGRect, gray: CGFloat) {

    /// Fills this list of rects in the
current NSGraphicsContext with that
```

```
rect's
    /// associated gray component value
in the DeviceGray color space.
    /// The compositing operation of the
fill defaults to the context's
    /// compositing operation, not
necessarily using `.copy` like
    /// `NSRectFillListWithGrays()`.
    /// - precondition: There must be a
set current NSGraphicsContext.
    @available(swift 4)
    public func fill(using operation:
NSCompositingOperation =
NSGraphicsContext.current?.compositingOpe
ration ?? .sourceOver)
}

@available(macOS 10.9, *)
extension NSWindow.Depth {

    @available(swift 4)
    public static func
bestDepth(colorSpaceName:
NSColorSpaceName, bitsPerSample: Int,
bitsPerPixel: Int, isPlanar: Bool) ->
(NSWindow.Depth, isExactMatch: Bool)

    @available(swift 4)
    public static var availableDepths:
[NSWindow.Depth] { get }
}

@available(macOS, introduced: 10.9,
deprecated: 14.0, message: "Use
NSCursor.disappearingItemCursor instead")
extension NSAnimationEffect {

    @available(swift 4)
    public func show(centeredAt
centerLocation: NSPoint, size: NSSize,
completionHandler: @escaping () -> Void =
{ })
}

@available(macOS 10.9, *)
extension NSSound {

    @available(swift 4)
    public static func beep()
}

@available(macOS 14.0, *)
extension Preview {

    /// Creates a preview of an NSView.
    ///
    /// The `#Preview` macro expands into
a declaration that calls this
initializer. To create a preview
    /// that appears in the canvas, you
must use the macro, not call this
initializer directly.
    @MainActor public init(_ name:
String? = nil, traits:
PreviewTrait<Preview.ViewTraits>...,
body: @escaping @MainActor () -> NSView)
```

```
    /// Creates a preview of an
NSViewController.
    ///
    /// The `#Preview` macro expands into
a declaration that calls this
initializer. To create a preview
    /// that appears in the canvas, you
must use the macro, not call this
initializer directly.
    @MainActor public init(_ name:
String? = nil, traits:
PreviewTrait<Preview.ViewTraits>...,
body: @escaping @MainActor () ->
NSViewController)
}

@available(macOS, introduced: 10.9,
deprecated: 10.14, message: "Please use
Metal or MetalKit.")
extension NSOpenGLGlobalOption {

    @available(swift 4)
    public var globalValue: GLint
}

@available(macOS, introduced: 10.9,
deprecated: 10.14, message: "Please use
Metal or MetalKit.")
extension NSOpenGLContext {

    @available(swift 4)
    public static var openGLVersion:
```

```
(major: GLint, minor: GLint) { get }
}

extension NSViewController {

    @available(swift 5.1)
    @available(macOS 13.3, *)
    @propertyWrapper public struct
ViewLoading<Value> {

        public init()

        public init(wrappedValue: Value)
    }
}

extension NSWindowController {

    @available(swift 5.1)
    @available(macOS 13.3, *)
    @propertyWrapper public struct
WindowLoading<Value> {

        public init()

        public init(wrappedValue: Value)
    }
}

@available(macOS 10.9, *)
extension NSGradient {

    public convenience init?
```

```
(colorsAndLocations objects: (NSColor,
CGFloat)...)
}

@available(macOS 10.10, *)
extension NSStoryboard {

    @available(macOS 10.15, *)
    public func
instantiateInitialController<Controller>(
creator: ((NSCoder) -> Controller?)? =
nil) -> Controller? where Controller :
NSViewController

    @available(macOS 10.15, *)
    public func
instantiateInitialController<Controller>(
creator: ((NSCoder) -> Controller?)? =
nil) -> Controller? where Controller :
NSWindowController

    @available(macOS 10.15, *)
    public func
instantiateController<Controller>(identif
ier: NSStoryboard.SceneIdentifier,
creator: ((NSCoder) -> Controller?)? =
nil) -> Controller where Controller :
NSViewController

    @available(macOS 10.15, *)
    public func
instantiateController<Controller>(identif
ier: NSStoryboard.SceneIdentifier,
creator: ((NSCoder) -> Controller?)? =
nil) -> Controller where Controller :
NSWindowController
}

@available(macCatalyst 18.0, macOS 15.0,
*)
extension NSCursor {

    /// Returns the cursor for resizing a
column (vertical divider) in the
specified direction.
    /// - Parameter directions: The
directions in which a column can be
resized. This must not be empty.
    public class func
columnResize(directions:
NSHorizontalDirection.Set) -> NSCursor

    /// Returns the cursor for resizing a
row (horizontal divider) in the specified
direction.
    /// - Parameter directions: The
directions in which a row can be resized.
This must not be empty.
    public class func
rowResize(directions:
NSVerticalDirection.Set) -> NSCursor

    /// The direction in which a
rectangular frame can be resized.
    public enum FrameResizeDirection :
Int8, CaseIterable {
        /// Indicates that the
rectangular frame can be resized inwards
to be smaller.
        case inward

        /// Indicates that the
rectangular frame can be resized outwards
to be larger.
        case outward

        /// An efficient set of frame
resize directions.
        public struct Set : OptionSet,
Equatable, Hashable {

            /// 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.
            public typealias Element =
NSCursor.FrameResizeDirection.Set

            /// 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.
            public typealias RawValue =
Int8

            /// 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:
NSCursor.FrameResizeDirection.Set.RawValu
e

            /// 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:
NSCursor.FrameResizeDirection.Set.RawValu
e)

            /// A set containing only the
inward resize direction.
            public static let inward:
NSCursor.FrameResizeDirection.Set

            /// A set containing only the
outward resize direction.
            public static let outward:
NSCursor.FrameResizeDirection.Set

            /// A set containing the
inward and outward resizing directions.
            public static let all:
NSCursor.FrameResizeDirection.Set

            /// Creates a set of
directions containing only the specified
direction.
            public init(_ direction:
NSCursor.FrameResizeDirection)
```

```
            /// The type of the elements
of an array literal.
            @available(macOS 15.0,
macCatalyst 18.0, *)
            public typealias
ArrayLiteralElement =
NSCursor.FrameResizeDirection.Set.Element
        }

        /// Creates a new instance with
the specified raw value.
        ///
        /// If there is no value of the
type that corresponds with the specified
raw
        /// value, this initializer
returns `nil`. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter,
Legal
        ///     }
        ///
        ///     print(PaperSize(rawValue:
"Legal"))
        ///     // Prints
"Optional("PaperSize.Legal")"
        ///
        ///     print(PaperSize(rawValue:
"Tabloid"))
        ///     // Prints "nil"
        ///
        /// - Parameter rawValue: The raw
```

```
value to use for the new instance.
        public init?(rawValue: Int8)

        /// A type that can represent a
collection of all values of this type.
        @available(macOS 15.0,
macCatalyst 18.0, *)
        public typealias AllCases =
[NSCursor.FrameResizeDirection]

        /// 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(macOS 15.0,
macCatalyst 18.0, *)
        public typealias RawValue = Int8

        /// A collection of all values of
this type.
        nonisolated public static var
allCases: [NSCursor.FrameResizeDirection]
{ get }

        /// 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 var rawValue: Int8 { get }
    }

    /// Returns the cursor for resizing a
rectangular frame from the specified edge
or corner.
    /// - Parameters:
    ///   - position: The position along
the perimeter of a rectangular frame (its
edges and corners) from which it’s
resized.
```

```
    ///   - directions: The directions in which a rectangular frame can be resized. This must not be empty.
    public class func frameResize(position: NSCursor.FrameResizePosition, directions: NSCursor.FrameResizeDirection.Set) -> NSCursor
}

@available(macCatalyst 18.0, macOS 15.0, *)
extension NSCursor.FrameResizePosition : CaseIterable {

    /// A collection of all values of this type.
    public static var allCases: [NSCursor.FrameResizePosition] { get }

    /// A type that can represent a collection of all values of this type.
    @available(macOS 15.0, macCatalyst 18.0, *)
    public typealias AllCases = [NSCursor.FrameResizePosition]
}

extension NSCursor.FrameResizePosition {

    /// The leading edge of the frame, in the given user interface layout direction.
    @available(macOS 15.0, *)
    public static func leading(relativeTo
layoutDirection:
NSUserInterfaceLayoutDirection) ->
NSCursor.FrameResizePosition

    /// The trailing edge of the frame,
in the given user interface layout
direction.
    @available(macOS 15.0, *)
    public static func
trailing(relativeTo layoutDirection:
NSUserInterfaceLayoutDirection) ->
NSCursor.FrameResizePosition

    /// The top leading corner of the
frame, in the given user interface layout
direction.
    @available(macOS 15.0, *)
    public static func
topLeading(relativeTo layoutDirection:
NSUserInterfaceLayoutDirection) ->
NSCursor.FrameResizePosition

    /// The top trailing corner of the
frame, in the given user interface layout
direction.
    @available(macOS 15.0, *)
    public static func
topTrailing(relativeTo layoutDirection:
NSUserInterfaceLayoutDirection) ->
NSCursor.FrameResizePosition
```

```
    /// The bottom leading corner of the
frame, in the given user interface layout
direction.
    @available(macOS 15.0, *)
    public static func
bottomLeading(relativeTo layoutDirection:
NSUserInterfaceLayoutDirection) ->
NSCursor.FrameResizePosition

    /// The bottom trailing corner of the
frame, in the given user interface layout
direction.
    @available(macOS 15.0, *)
    public static func
bottomTrailing(relativeTo
layoutDirection:
NSUserInterfaceLayoutDirection) ->
NSCursor.FrameResizePosition
}

@available(macOS 10.9, *)
extension NSEvent {

    public struct SpecialKey :
RawRepresentable, Equatable, Hashable {

        /// Creates a new instance with
the specified raw value.
        ///
        /// If there is no value of the
type that corresponds with the specified
raw
        /// value, this initializer
```

```
returns `nil`. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter,
Legal
        ///     }
        ///
        ///     print(PaperSize(rawValue:
"Legal"))
        ///     // Prints
"Optional("PaperSize.Legal")"
        ///
        ///     print(PaperSize(rawValue:
"Tabloid"))
        ///     // Prints "nil"
        ///
        /// - Parameter rawValue: The raw
value to use for the new instance.
        public init(rawValue: Int)

        /// 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

        public var unicodeScalar:
Unicode.Scalar { get }

        /// 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(macOS 10.9, *)
        public typealias RawValue = Int
    }

    /// Returns `nil` if the receiver is
not a “special” key event.
```

```
    public var specialKey:
NSEvent.SpecialKey? { get }
}

@available(macOS 10.9, *)
extension CocoaError.Code {

    public static var
textReadInapplicableDocumentType:
CocoaError.Code { get }

    public static var
textWriteInapplicableDocumentType:
CocoaError.Code { get }

    public static var
serviceApplicationNotFound:
CocoaError.Code { get }

    public static var
serviceApplicationLaunchFailed:
CocoaError.Code { get }

    public static var
serviceRequestTimedOut: CocoaError.Code {
get }

    public static var
serviceInvalidPasteboardData:
CocoaError.Code { get }

    public static var
serviceMalformedServiceDictionary:
```

```
CocoaError.Code { get }

    public static var
serviceMiscellaneousError:
CocoaError.Code { get }

    public static var
sharingServiceNotConfigured:
CocoaError.Code { get }

    @available(macOS 10.13, *)
    public static var
fontAssetDownloadError: CocoaError.Code {
get }
}

@available(macOS 10.9, *)
extension CocoaError.Code {

    @available(*, deprecated, renamed:
"textReadInapplicableDocumentType")
    public static var
textReadInapplicableDocumentTypeError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"textWriteInapplicableDocumentType")
    public static var
textWriteInapplicableDocumentTypeError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceApplicationNotFound")
```

```
    public static var
serviceApplicationNotFoundError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceApplicationLaunchFailed")
    public static var
serviceApplicationLaunchFailedError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceRequestTimedOut")
    public static var
serviceRequestTimedOutError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceInvalidPasteboardData")
    public static var
serviceInvalidPasteboardDataError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceMalformedServiceDictionary")
    public static var
serviceMalformedServiceDictionaryError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceMiscellaneousError")
    public static var
serviceMiscellaneous: CocoaError.Code {
get }
```

```
    @available(*, deprecated, renamed:
"sharingServiceNotConfigured")
    public static var
sharingServiceNotConfiguredError:
CocoaError.Code { get }
}

@available(macOS 10.9, *)
extension CocoaError {

    public static var
textReadInapplicableDocumentType:
CocoaError.Code { get }

    public static var
textWriteInapplicableDocumentType:
CocoaError.Code { get }

    public static var
serviceApplicationNotFound:
CocoaError.Code { get }

    public static var
serviceApplicationLaunchFailed:
CocoaError.Code { get }

    public static var
serviceRequestTimedOut: CocoaError.Code {
get }

    public static var
serviceInvalidPasteboardData:
```

```
CocoaError.Code { get }

    public static var
serviceMalformedServiceDictionary:
CocoaError.Code { get }

    public static var
serviceMiscellaneous: CocoaError.Code {
get }

    public static var
sharingServiceNotConfigured:
CocoaError.Code { get }

    @available(macOS 10.13, *)
    public static var
fontAssetDownloadError: CocoaError.Code {
get }
}

@available(macOS 10.9, *)
extension CocoaError {

    @available(*, deprecated, renamed:
"textReadInapplicableDocumentType")
    public static var
textReadInapplicableDocumentTypeError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"textWriteInapplicableDocumentType")
    public static var
textWriteInapplicableDocumentTypeError:
```

```
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceApplicationNotFound")
    public static var
serviceApplicationNotFoundError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceApplicationLaunchFailed")
    public static var
serviceApplicationLaunchFailedError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceRequestTimedOut")
    public static var
serviceRequestTimedOutError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceInvalidPasteboardData")
    public static var
serviceInvalidPasteboardDataError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceMalformedServiceDictionary")
    public static var
serviceMalformedServiceDictionaryError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
```

```
"serviceMiscellaneous")
    public static var
serviceMiscellaneousError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"sharingServiceNotConfigured")
    public static var
sharingServiceNotConfiguredError:
CocoaError.Code { get }
}

@available(macOS 10.9, *)
extension CocoaError {

    public var isServiceError: Bool { get
}

    public var isSharingServiceError:
Bool { get }

    public var isTextReadWriteError: Bool
{ get }

    @available(macOS 10.13, *)
    public var isFontError: Bool { get }
}

@available(macOS 10.9, *)
extension CocoaError {

    @available(*, deprecated, renamed:
"textReadInapplicableDocumentType")
```

```
    public static var
TextReadInapplicableDocumentTypeError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"textWriteInapplicableDocumentType")
    public static var
TextWriteInapplicableDocumentTypeError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceApplicationNotFound")
    public static var
ServiceApplicationNotFoundError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceApplicationLaunchFailed")
    public static var
ServiceApplicationLaunchFailedError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceRequestTimedOut")
    public static var
ServiceRequestTimedOutError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceInvalidPasteboardData")
    public static var
ServiceInvalidPasteboardDataError:
CocoaError.Code { get }
```

```
    @available(*, deprecated, renamed:
"serviceMalformedServiceDictionary")
    public static var
ServiceMalformedServiceDictionaryError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceMiscellaneous")
    public static var
ServiceMiscellaneousError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"sharingServiceNotConfigured")
    public static var
SharingServiceNotConfiguredError:
CocoaError.Code { get }
}

@available(macOS 10.9, *)
extension CocoaError.Code {

    @available(*, deprecated, renamed:
"textReadInapplicableDocumentType")
    public static var
TextReadInapplicableDocumentTypeError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"textWriteInapplicableDocumentType")
    public static var
TextWriteInapplicableDocumentTypeError:
```

```
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceApplicationNotFound")
    public static var
ServiceApplicationNotFoundError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceApplicationLaunchFailed")
    public static var
ServiceApplicationLaunchFailedError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceRequestTimedOut")
    public static var
ServiceRequestTimedOutError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceInvalidPasteboardData")
    public static var
ServiceInvalidPasteboardDataError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"serviceMalformedServiceDictionary")
    public static var
ServiceMalformedServiceDictionaryError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
```

```
"serviceMiscellaneous")
    public static var
ServiceMiscellaneousError:
CocoaError.Code { get }

    @available(*, deprecated, renamed:
"sharingServiceNotConfigured")
    public static var
SharingServiceNotConfiguredError:
CocoaError.Code { get }
}

@available(macOS 14.0, *)
extension NSMenuItem {

    /// Creates a menu item representing
a section header with the provided title.
    /// Section header items are used to
provide context to a grouping of menu
items.
    /// Items created using this method
are non-interactive and do not perform an
action.
    @available(*, deprecated, renamed:
"sectionHeader(title:)", message: "Use
sectionHeader(title:) instead.")
    public static func
sectionHeader(withTitle title: String) ->
NSMenuItem

    /// Creates a menu item representing
a section header with the provided title.
    /// Section header items are used to
```

```
provide context to a grouping of menu
items.
    /// Items created using this method
are non-interactive and do not perform an
action.
    public static func
sectionHeader(title: String) ->
NSMenuItem
}

@available(macOS 14.0, *)
extension NSMenu {

    /// Creates a palette menu displaying
user-selectable color tags
    /// using the provided template
image, tinted using the specified
    /// array of colors.
    ///
    /// Optionally allows observing
changes to the selection state in
    /// the palette menu. The closure is
invoked after the selection
    /// has been updated. Currently
selected items can be retrieved
    /// from the `selectedItems`
property.
    public static func palette(colors:
[NSColor], titles: [String] = [],
template: NSImage? = nil,
onSelectionChange: ((NSMenu) -> Void)? =
nil) -> NSMenu
}
```

```
@available(macOS 14.0, *)
extension NSColor : Transferable {

    /// The representation used to import
and export the item.
    ///
    /// A ``transferRepresentation`` can
contain multiple representations
    /// for different content types.
    public static var
transferRepresentation: some
TransferRepresentation { get }

    /// The type of the representation
used to import and export the item.
    ///
    /// Swift infers this type from the
return value of the
    /// ``transferRepresentation``
property.
    @available(macOS 14.0, *)
    public typealias Representation =
some TransferRepresentation
}
```