

[SwiftUI](#) / Modal presentations

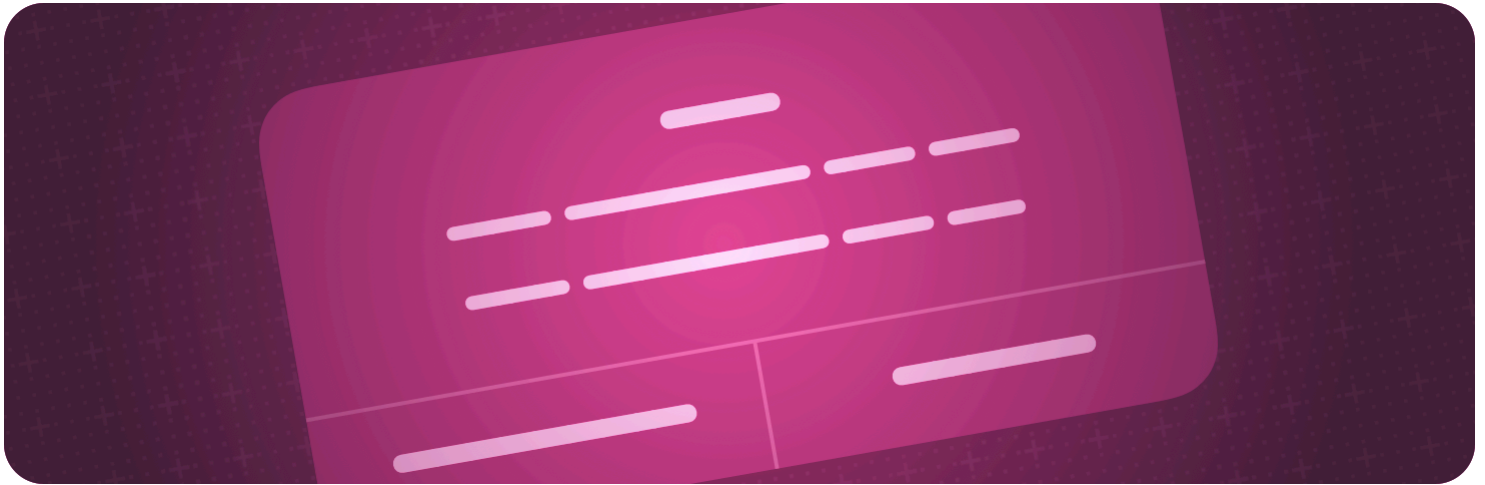
API Collection

# Modal presentations

Present content in a separate view that offers focused interaction.

## Overview

To draw attention to an important, narrowly scoped task, you display a modal presentation, like an alert, popover, sheet, or confirmation dialog.



In SwiftUI, you create a modal presentation using a view modifier that defines how the presentation looks and the condition under which SwiftUI presents it. SwiftUI detects when the condition changes and makes the presentation for you. Because you provide a [Binding](#) to the condition that initiates the presentation, SwiftUI can reset the underlying value when the user dismisses the presentation.

For design guidance, see [Modality](#) in the Human Interface Guidelines.

---

## Topics

## Configuring a dialog

```
struct DialogSeverity
```

The severity of an alert or confirmation dialog.

## Showing a sheet, cover, or popover

```
func sheet<Content>(isPresented: Binding<Bool>, onDismiss: (() -> Void)?, content: () -> Content) -> some View
```

Presents a sheet when a binding to a Boolean value that you provide is true.

```
func sheet<Item, Content>(item: Binding<Item?>, onDismiss: (() -> Void)?, content: (Item) -> Content) -> some View
```

Presents a sheet using the given item as a data source for the sheet's content.

```
func fullScreenCover<Content>(isPresented: Binding<Bool>, onDismiss: (() -> Void)?, content: () -> Content) -> some View
```

Presents a modal view that covers as much of the screen as possible when binding to a Boolean value you provide is true.

```
func fullScreenCover<Item, Content>(item: Binding<Item?>, onDismiss: (() -> Void)?, content: (Item) -> Content) -> some View
```

Presents a modal view that covers as much of the screen as possible using the binding you provide as a data source for the sheet's content.

```
func popover<Item, Content>(item: Binding<Item?>, attachmentAnchor: PopoverAttachmentAnchor, arrowEdge: Edge?, content: (Item) -> Content) -> some View
```

Presents a popover using the given item as a data source for the popover's content.

```
func popover<Content>(isPresented: Binding<Bool>, attachmentAnchor: PopoverAttachmentAnchor, arrowEdge: Edge?, content: () -> Content) -> some View
```

Presents a popover when a given condition is true.

```
enum PopoverAttachmentAnchor
```

An attachment anchor for a popover.

## Adapting a presentation size

```
func presentationCompactAdaptation(horizontal: PresentationAdaptation,
vertical: PresentationAdaptation) -> some View
```

Specifies how to adapt a presentation to horizontally and vertically compact size classes.

```
func presentationCompactAdaptation(PresentationAdaptation) -> some View
```

Specifies how to adapt a presentation to compact size classes.

```
struct PresentationAdaptation
```

Strategies for adapting a presentation to a different size class.

```
func presentationSizing(some PresentationSizing) -> some View
```

Sets the sizing of the containing presentation.

```
protocol PresentationSizing
```

A type that defines the size of the presentation content and how the presentation size adjusts to its content's size changing.

```
struct PresentationSizingRoot
```

A proxy to a view provided to the presentation with a defined presentation size.

```
struct PresentationSizingContext
```

Contextual information about a presentation.

## Configuring a sheet's height

```
func presentationDetents(Set<PresentationDetent>) -> some View
```

Sets the available detents for the enclosing sheet.

```
func presentationDetents(Set<PresentationDetent>, selection: Binding<
PresentationDetent>) -> some View
```

Sets the available detents for the enclosing sheet, giving you programmatic control of the currently selected detent.

```
func presentationContentInteraction(PresentationContentInteraction) ->
some View
```

Configures the behavior of swipe gestures on a presentation.

```
func presentationDragIndicator(Visibility) -> some View
```

Sets the visibility of the drag indicator on top of a sheet.

```
struct PresentationDetent
```

A type that represents a height where a sheet naturally rests.

`protocol CustomPresentationDetent`

The definition of a custom detent with a calculated height.

`struct PresentationContentInteraction`

A behavior that you can use to influence how a presentation responds to swipe gestures.

## Styling a sheet and its background

`func presentationCornerRadius(CGFloat?) -> some View`

Requests that the presentation have a specific corner radius.

`func presentationBackground<S>(S) -> some View`

Sets the presentation background of the enclosing sheet using a shape style.

`func presentationBackground<V>(alignment: Alignment, content: () -> V) -> some View`

Sets the presentation background of the enclosing sheet to a custom view.

`func presentationBackgroundInteraction(PresentationBackgroundInteraction) -> some View`

Controls whether people can interact with the view behind a presentation.

`struct PresentationBackgroundInteraction`

The kinds of interaction available to views behind a presentation.

## Presenting an alert

`struct AlertScene`

A scene that renders itself as a standalone alert dialog.

`func alert(_:isPresented:actions:)`

Presents an alert when a given condition is true, using a text view for the title.

`func alert(_:isPresented:presenting:actions:)`

Presents an alert using the given data to produce the alert's content and a text view as a title.

`func alert<E, A>(isPresented: Binding<Bool>, error: E?, actions: () -> A) -> some View`

Presents an alert when an error is present.

`func alert(_:isPresented:actions:message:)`

Presents an alert with a message when a given condition is true using a text view as a title.

```
func alert(_:isPresented:presenting:actions:message:)
```

Presents an alert with a message using the given data to produce the alert's content and a text view for a title.

```
func alert<E, A, M>(isPresented: Binding<Bool>, error: E?, actions: (E)
-> A, message: (E) -> M) -> some View
```

Presents an alert with a message when an error is present.

## Getting confirmation for an action

```
func confirmationDialog(_:isPresented:titleVisibility:actions:)
```

Presents a confirmation dialog when a given condition is true, using a text view for the title.

```
func confirmationDialog(_:isPresented:titleVisibility:presenting:
actions:)
```

Presents a confirmation dialog using data to produce the dialog's content and a text view for the title.

```
func dismissalConfirmationDialog(_:shouldPresent:actions:)
```

Presents a confirmation dialog when a dismiss action has been triggered.

## Showing a confirmation dialog with a message

```
func confirmationDialog(_:isPresented:titleVisibility:actions:message:)
```

Presents a confirmation dialog with a message when a given condition is true, using a text view for the title.

```
func confirmationDialog(_:isPresented:titleVisibility:presenting:
actions:message:)
```

Presents a confirmation dialog with a message using data to produce the dialog's content and a text view for the message.

```
func dismissalConfirmationDialog(_:shouldPresent:actions:message:)
```

Presents a confirmation dialog when a dismiss action has been triggered.

## Configuring a dialog

```
func dialogIcon(Image?) -> some View
```

Configures the icon used by dialogs within this view.

```
func dialogIcon(Image?) -> some Scene
```

Configures the icon used by alerts.

```
func dialogSeverity(DialogSeverity) -> some View
```

```
func dialogSeverity(DialogSeverity) -> some Scene
```

Sets the severity for alerts.

```
func dialogSuppressionToggle(isSuppressed: Binding<Bool>) -> some View
```

Enables user suppression of dialogs and alerts presented within `self`, with a default suppression message on macOS. Unused on other platforms.

```
func dialogSuppressionToggle(isSuppressed: Binding<Bool>) -> some Scene
```

Enables user suppression of an alert with a custom suppression message.

```
func dialogSuppressionToggle(_:isSuppressed:)
```

Enables user suppression of dialogs and alerts presented within `self`, with a custom suppression message on macOS. Unused on other platforms.

## Exporting to file

```
func fileExporter(isPresented:document:contentType:defaultFilename:onCompletion:)
```

Presents a system interface for exporting a document that's stored in a value type, like a structure, to a file on disk.

```
func fileExporter(isPresented:documents:contentType:onCompletion:)
```

Presents a system interface for exporting a collection of value type documents to files on disk.

```
func fileExporter(isPresented:document:contentTypes:defaultFilename:onCompletion:onCancellation:)
```

Presents a system interface for allowing the user to export a `FileDocument` to a file on disk.

```
func fileExporter(isPresented:documents:contentTypes:onCompletion:onCancellation:)
```

Presents a system dialog for allowing the user to export a collection of documents that conform to `FileDocument` to files on disk.

```
func fileExporter<T>(isPresented: Binding<Bool>, item: T?, contentTypes : [UTType], defaultFilename: String?, onCompletion: (Result<URL, any Error>) -> Void, onCancellation: () -> Void) -> some View
```

Presents a system interface allowing the user to export a Transferable item to file on disk.

```
func fileExporter<C, T>(isPresented: Binding<Bool>, items: C, content
Types: [UTType], onCompletion: (Result<[URL], any Error>) -> Void, on
Cancellation: () -> Void) -> some View
```

Presents a system interface allowing the user to export a collection of items to files on disk.

```
func fileExporterFilenameLabel(_:)
```

On macOS, configures the fileExporter with a label for the file name field.

## Importing from file

```
func fileImporter(isPresented: Binding<Bool>, allowedContentTypes: [
UTType], allowsMultipleSelection: Bool, onCompletion: (Result<[URL],
any Error>) -> Void) -> some View
```

Presents a system interface for allowing the user to import multiple files.

```
func fileImporter(isPresented: Binding<Bool>, allowedContentTypes: [
UTType], onCompletion: (Result<URL, any Error>) -> Void) -> some View
```

Presents a system interface for allowing the user to import an existing file.

```
func fileImporter(isPresented: Binding<Bool>, allowedContentTypes: [
UTType], allowsMultipleSelection: Bool, onCompletion: (Result<[URL],
any Error>) -> Void, onCancelled: () -> Void) -> some View
```

Presents a system dialog for allowing the user to import multiple files.

## Moving a file

```
func fileMover(isPresented: Binding<Bool>, file: URL?, onCompletion: (
Result<URL, any Error>) -> Void) -> some View
```

Presents a system interface for allowing the user to move an existing file to a new location.

```
func fileMover<C>(isPresented: Binding<Bool>, files: C, onCompletion: (
Result<[URL], any Error>) -> Void) -> some View
```

Presents a system interface for allowing the user to move a collection of existing files to a new location.

```
func fileMover(isPresented: Binding<Bool>, file: URL?, onCompletion: (
Result<URL, any Error>) -> Void, onCancelled: () -> Void) -> some
View
```

Presents a system dialog for allowing the user to move an existing file to a new location.

```
func fileMover<C>(isPresented: Binding<Bool>, files: C, onCompletion: (Result<[URL], any Error>) -> Void, onCancellation: () -> Void) -> some View
```

Presents a system dialog for allowing the user to move a collection of existing files to a new location.

## Configuring a file dialog

```
func fileDialogBrowserOptions(FileDialogBrowserOptions) -> some View
```

On macOS, configures the `fileExporter`, `fileImporter`, or `fileMover` to provide a refined URL search experience: include or exclude hidden files, allow searching by tag, etc.

```
func fileDialogConfirmationLabel(_:)
```

On macOS, configures the `fileExporter`, `fileImporter`, or `fileMover` with a custom confirmation button label.

```
func fileDialogCustomizationID(String) -> some View
```

On macOS, configures the `fileExporter`, `fileImporter`, or `fileMover` to persist and restore the file dialog configuration.

```
func fileDialogDefaultDirectory(URL?) -> some View
```

Configures the `fileExporter`, `fileImporter`, or `fileMover` to open with the specified default directory.

```
func fileDialogImportsUnresolvedAliases(Bool) -> some View
```

On macOS, configures the `fileExporter`, `fileImporter`, or `fileMover` behavior when a user chooses an alias.

```
func fileDialogMessage(_:)
```

On macOS, configures the `fileExporter`, `fileImporter`, or `fileMover` with a custom text that is presented to the user, similar to a title.

```
func fileDialogURLEnabled(Predicate<URL>) -> some View
```

On macOS, configures the `fileImporter` or `fileMover` to conditionally disable presented URLs.

```
struct FileDialogBrowserOptions
```

The way that file dialogs present the file system.

## Presenting an inspector

```
func inspector<V>(isPresented: Binding<Bool>, content: () -> V) -> some View
```

Inserts an inspector at the applied position in the view hierarchy.

```
func inspectorColumnWidth(CGFloat) -> some View
```

Sets a fixed, preferred width for the inspector containing this view when presented as a trailing column.

```
func inspectorColumnWidth(min: CGFloat?, ideal: CGFloat, max: CGFloat?) -> some View
```

Sets a flexible, preferred width for the inspector in a trailing-column presentation.

## Dismissing a presentation

```
var isPresented: Bool
```

A Boolean value that indicates whether the view associated with this environment is currently presented.

```
var dismiss: DismissAction
```

An action that dismisses the current presentation.

```
struct DismissAction
```

An action that dismisses a presentation.

```
func interactiveDismissDisabled(Bool) -> some View
```

Conditionally prevents interactive dismissal of presentations like popovers, sheets, and inspectors.

## Deprecated modal presentations

```
struct Alert
```

A representation of an alert presentation.

Deprecated

```
struct ActionSheet
```

A representation of an action sheet presentation.

Deprecated

---

# See Also

## App structure

### ☰ App organization

Define the entry point and top-level structure of your app.

### ☰ Scenes

Declare the user interface groupings that make up the parts of your app.

### ☰ Windows

Display user interface content in a window or a collection of windows.

### ☰ Immersive spaces

Display unbounded content in a person's surroundings.

### ☰ Documents

Enable people to open and manage documents.

### ☰ Navigation

Enable people to move between different parts of your app's view hierarchy within a scene.

### ☰ Toolbars

Provide immediate access to frequently used commands and controls.

### ☰ Search

Enable people to search for text or other content within your app.

### ☰ App extensions

Extend your app's basic functionality to other parts of the system, like by adding a Widget.