Instance Method

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

Presents a confirmation dialog using data to produce the dialog's content and a localized string key for the title.

iOS 15.0+ | iPadOS 15.0+ | Mac Catalyst 15.0+ | macOS 12.0+ | tvOS 15.0+ | visionOS 1.0+ | watchOS 8.0+

```
nonisolated
func confirmationDialog<A, T>(
    _ titleKey: LocalizedStringKey,
    isPresented: Binding<Bool>,
    titleVisibility: Visibility = .automatic,
    presenting data: T?,
    @ViewBuilder actions: (T) -> A
) -> some View where A : View
```

Show all declarations ⊕

Parameters

`titleKey`
The key for the localized string that describes the title of the dialog.

`isPresented`
A binding to a Boolean value that determines whether to present the dialog. When the user presses or taps the dialog's default action button, the system sets this value to `false`, dismissing the dialog.

`titleVisibility`
The visibility of the dialog's title. The default value is `Visibility.automatic`.

`data`
An optional source of truth for the confirmation dialog. The system passes the contents to the modifier's closures. You use this data to populate the fields of a confirmation dialog that you create that the system displays to the user.

`actions`
A view builder returning the dialog's actions given the currently available data.

Discussion

In order for the interface to appear, both `isPresented` must be `true` and `data` must not be `nil`. `data` should not change after the presentation occurs. Any changes which occur after the presentation occurs will be ignored.

Use this method when you need to populate the fields of a confirmation dialog with content from a data source. The example below shows a custom data source, `FileDetails`, that provides data to populate the dialog:

```
struct FileDetails: Identifiable {
    var id: String { name }
    let name: String
    let fileType: UTType
}
struct ConfirmFileImport: View {
    @State private var isConfirming = false
    @State private var dialogDetail: FileDetails?
    var body: some View {
        Button("Import File") {
            dialogDetail = FileDetails(
                name: "MyImageFile.png", fileType: .png)
            isConfirming = true
        }
        .confirmationDialog(
            "Are you sure you want to import this file?",
            isPresented: $isConfirming, presenting: dialogDetail
        ) { detail in
            Button {
                // Handle import action.
            } label: {
                Text("""
                Import \(detail.name)
                File Type: \(detail.fileType.description)
                """)
            }
            Button("Cancel", role: .cancel) {
                dialogDetail = nil
            }
        }
    }
}
```

This modifier creates a `Text` view for the title on your behalf, and treats the localized key similar to `init(_:tableName:bundle:comment:)`. See `Text` for more information about localizing strings.

All actions in a confirmation dialog will dismiss the dialog after the action runs. The default button will be shown with greater prominence. You can influence the default button by assigning it the `defaultAction` keyboard shortcut.

The system may reorder the buttons based on their role and prominence.

Dialogs include a standard dismiss action by default. If you provide a button with a role of `cancel`, that button takes the place of the default dismiss action. You don't have to dismiss the presentation with the cancel button's action.

Note

In regular size classes in iOS, the system renders confirmation dialogs as a popover that the user dismisses by tapping anywhere outside the popover, rather than displaying the standard dismiss action.

On iOS, tvOS, and watchOS, confirmation dialogs only support controls with labels that are `Text`. Passing any other type of view results in the content being omitted.