

[UIKit](#) /  / [UIContextMenuInteraction](#) / Adding context menus in your app

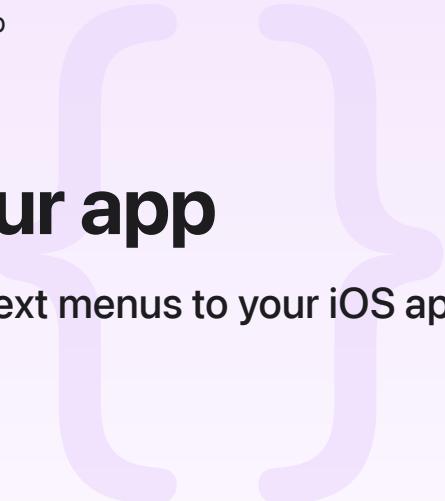
Sample Code

Adding context menus in your app

Provide quick access to useful actions by adding context menus to your iOS app.

[Download](#)

iOS 14.0+ | iPadOS 14.0+ | Xcode 12.0+



Overview

This sample project demonstrates how to add context menus to user-interface elements such as views, controls, table views, collection views, and web views. Apps enhance and extend context menus with actions, nested submenu actions, and custom previews. For more information on context menu design, see [Human Interface Guidelines](#).

Create a context menu

This sample shows two ways to create context menus:

1. For a [UIView](#), the sample creates a [UIContextMenuInteraction](#) object and attaches it to that view. The [UIContextMenuInteraction](#) object focuses the user's attention on a specific portion of the UI, and provides actions for the user to perform on that content. Then, the app adopts the [UIContextMenuInteractionDelegate](#) protocol and provides a [UIContextMenuConfiguration](#) object.
2. For more specific UI elements like tables, collections, and web views, the app adopts specific protocols for those elements that return a [UIContextMenuConfiguration](#).

Adopt [UIContextMenuInteractionDelegate](#) to manage the lifecycle of context menus. The app implements [contextMenuInteraction\(_:configurationForMenuAtLocation:\)](#) to return a [UIContextMenuConfiguration](#) object. This configuration object consists of an optional identifier, a previewProvider that returns a [UIViewController](#), and an actionProvider that returns a [UIContextMenu](#) with a set of [UIAction](#).

Context menus with rich content also provide a `UITargetedPreview`, an object that describes the source view when opening and animating the contextual menu. A `UITargetedPreview` specifies the view to use during animated transitions.

Add a context menu to a view

This sample adds a context menu to a `UIView` by calling `addInteraction(_ :)`.

```
let interaction = UIContextMenuInteraction(delegate: self)
imageView.addInteraction(interaction)
```

When the user touches and holds on that view, the view asks its delegate to provide the context menu by calling `contextMenuInteraction(_ :configurationForMenuAtLocation:)`.

```
func contextMenuInteraction(_ interaction: UIContextMenuInteraction,
                           configurationForMenuAtLocation location: CGPoint) -> UIC
return UIContextMenuConfiguration(identifier: nil,
                                  previewProvider: nil,
                                  actionProvider: {
    suggestedActions in
    let inspectAction =
        UIAction(title: NSLocalizedString("InspectTitle", comment: ""),
                 image: UIImage(systemName: "arrow.up.square")) { action in
            self.performInspect()
        }

    let duplicateAction =
        UIAction(title: NSLocalizedString("DuplicateTitle", comment: ""),
                 image: UIImage(systemName: "plus.square.on.square")) { action in
            self.performDuplicate()
        }

    let deleteAction =
        UIAction(title: NSLocalizedString("DeleteTitle", comment: ""),
                 image: UIImage(systemName: "trash"),
                 attributes: .destructive) { action in
            self.performDelete()
        }

    return UIMenu(title: "", children: [inspectAction, duplicateAction, deleteA
})
```

}

Add a context menu to a table view

This sample adds a context menu to a `UITableView` when the user touches and holds a `UITableViewCell`. UITableView asks its delegate to provide the context menu by calling `tableView(_:contextMenuConfigurationForRowAt:point:)`.

```
override func tableView(_ tableView: UITableView,
                      contextMenuConfigurationForRowAt indexPath: IndexPath,
                      point: CGPoint) -> UIContextMenuConfiguration? {
    return UIContextMenuConfiguration(identifier: nil,
                                      previewProvider: nil,
                                      actionProvider: {
        suggestedActions in
        let inspectAction =
            UIAction(title: NSLocalizedString("InspectTitle", comment: ""),
                     image: UIImage(systemName: "arrow.up.square")) { action in
                self.performInspect(indexPath)
            }
        let duplicateAction =
            UIAction(title: NSLocalizedString("DuplicateTitle", comment: ""),
                     image: UIImage(systemName: "plus.square.on.square")) { action in
                self.performDuplicate(indexPath)
            }
        let deleteAction =
            UIAction(title: NSLocalizedString("DeleteTitle", comment: ""),
                     image: UIImage(systemName: "trash"),
                     attributes: .destructive) { action in
                self.performDelete(indexPath)
            }
        return UIMenu(title: "", children: [inspectAction, duplicateAction, deleteAction])
    })
}
```

Add a context menu to a collection view

This sample adds a context menu to a `UICollectionView` when the user touches and holds a `UICollectionViewCell`. UICollectionView asks its delegate to provide the context menu by calling `collectionView(_:contextMenuConfigurationForItemAt:point:)`.

```
override func collectionView(_ collectionView: UICollectionView,
                           contextMenuConfigurationForItemAt indexPath: IndexPath,
                           point: CGPoint) -> UIContextMenuConfiguration? {
    return UIContextMenuConfiguration(identifier: nil, previewProvider: nil) { suggestedActions in
        let inspectAction = self.inspectAction(indexPath)
        let duplicateAction = self.duplicateAction(indexPath)
        let deleteAction = self.deleteAction(indexPath)
        return UIMenu(title: "", children: [inspectAction, duplicateAction, deleteAction])
    }
}
```

Add a context menu to a control

This sample adds a context menu to a [UIControl](#). UIKit attaches a context menu to a UIControl in two different ways:

- By setting the menu property with a UIMenu object

```
let inspectAction = self.inspectAction()
let duplicateAction = self.duplicateAction()
let deleteAction = self.deleteAction()
buttonMenuAsPrimary.menu = UIMenu(title: "", children: [inspectAction, duplicateAction, deleteAction])
buttonMenuAsPrimary.showsMenuAsPrimaryAction = true
```

- By adding a UIContextMenuInteraction object

```
let interaction = UIContextMenuInteraction(delegate: self)
buttonMenu.addInteraction(interaction)
```

When the user touches and holds on that control, the app asks its delegate to provide a context menu by calling [contextMenuInteraction\(_:configurationForMenuAtLocation:\)](#)

```
func contextMenuInteraction(_ interaction: UIContextMenuInteraction,
                           configurationForMenuAtLocation location: CGPoint) -> UIContextMenuConfiguration {
    return UIContextMenuConfiguration(identifier: nil,
                                      previewProvider: nil,
                                      actionProvider: {
        suggestedActions in
        // Use the ContextMenu protocol to produce the UIActions.
        let inspectAction = self.inspectAction()
```

```

        let duplicateAction = self.duplicateAction()
        let deleteAction = self.deleteAction()
        return UIMenu(title: "", children: [inspectAction, duplicateAction, deleteAction])
    }
}

```

Add a context menu to a web view

Select the Basic test case from the sample's Web Views outline item. When the user touches and holds on the link within the `WKWebView`, the app presents a `SFSafariViewController` to show that link content with a set of default UIAction items. When the user touches the view controller, the user moves out of the app and into Safari.

Select the Preview Provider test case from the sample's Web Views outline item. When the user touches and holds on the link within the `WKWebView`, the app presents a custom view controller.

This sample intercepts and adds to that context menu by adopting the `WKUIDelegate` protocol. `WKWebView` asks its delegate to provide the context menu by calling `webView(_:contextMenuConfigurationForElement:completionHandler:)`.

```

func webView(_ webView: WKWebView,
            contextMenuConfigurationForElement elementInfo: WKContextMenuElementInfo,
            completionHandler: @escaping (UIContextMenuConfiguration?) -> Void) {
    let configuration =
        UIContextMenuConfiguration(identifier: nil,
                                  previewProvider: { return SFSafariViewController(),
                                  actionProvider: { elements in
            guard elements.isEmpty == false else { return nil }

            // Add our custom action to the existing actions passed in.
            var elementsToUse = elements
            let inspectAction = self.extraAction(elementInfo.linkURL!)
            let editMenu = UIMenu(title: "", options: .displayInline, children: [inspectAction])
            elementsToUse.append(editMenu)

            let contextMenuTitle = elementInfo.linkURL?.lastPathComponent
            return UIMenu(title: contextMenuTitle!, image: nil, identifier: nil, options: .displayInline, children: [editMenu]))
        })
    completionHandler(configuration)
}

```

See Also

Creating a context menu interaction object

`init(delegate: any UIContextMenuInteractionDelegate)`

Creates a context menu interaction object with the specified delegate object.

{ } Adding menus and shortcuts to the menu bar and user interface

Provide quick access to useful actions by adding menus and keyboard shortcuts to your Mac app built with Mac Catalyst.