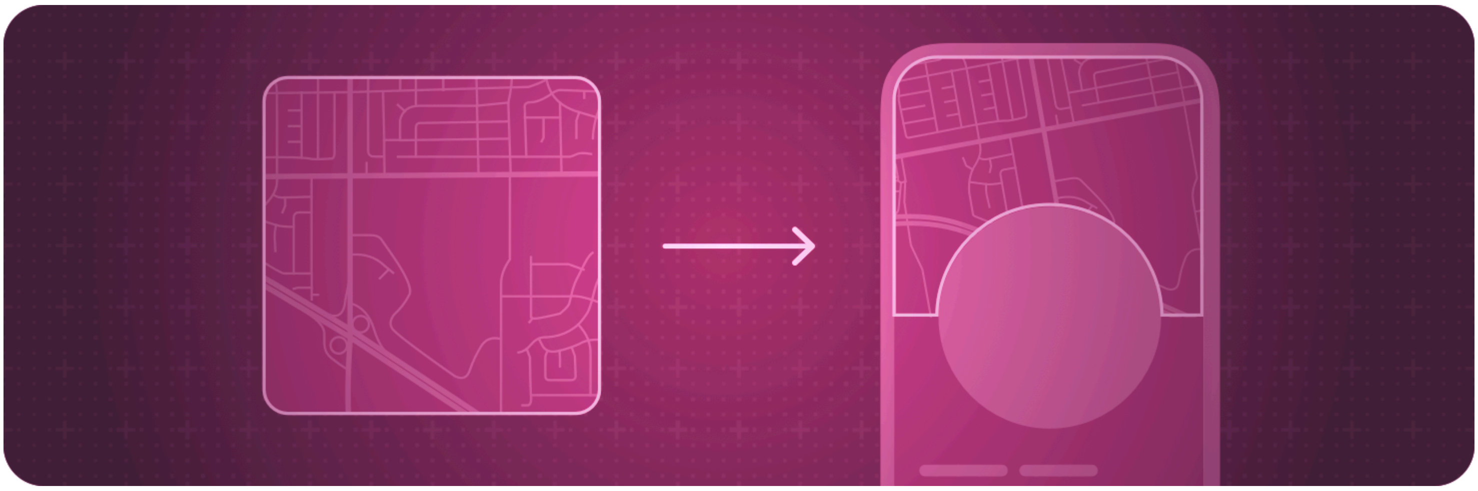
API Collection

Technology-specific views

Use SwiftUI views that other Apple frameworks provide.

Overview

To access SwiftUI views that another framework defines, import both SwiftUI and the other framework into the file where you use the view. You can find the framework to import by looking at the availability information on the view's documentation page.



For example, to use the Map view in your app, import both SwiftUI and MapKit.

```
import SwiftUI
import MapKit

struct MyMapView: View {
    // Center the map on Joshua Tree National Park.
    var region = MKCoordinateRegion(
            center: CLLocationCoordinate2D(latitude: 34.011_286, longitude: -116.16
            span: MKCoordinateSpan(latitudeDelta: 0.2, longitudeDelta: 0.2)
```

```
        )

    var body: some View {
        Map(initialPosition: .region(region))
    }
```

For design guidance, see Technologies in the Human Interface Guidelines.

Topics

Displaying web content

`@MainActor @preconcurrency struct WebView`

A view that displays some web content.

`@MainActor final class WebPage`

An object that controls and manages the behavior of interactive web content.

`func webViewBackForwardNavigationGestures(WebView.BackForwardNavigationGesturesBehavior) -> some View`

Determines whether horizontal swipe gestures trigger backward and forward page navigation.

`func webViewContentBackground(Visibility) -> some View`

Specifies the visibility of the webpage's natural background color within this view.

`func webViewContextMenu(menu: (WebView.ActivatedElementInfo) -> some View) -> some View`

Adds an item-based context menu to a WebView, replacing the default set of context menu items.

Beta

`func webViewElementFullscreenBehavior(WebView.ElementFullscreenBehavior) -> some View`

Determines whether a web view can display content full screen.

`func webViewLinkPreviews(WebView.LinkPreviewBehavior) -> some View`

Determines whether pressing a link displays a preview of the destination for the link.

`func webViewMagnificationGestures(WebView.MagnificationGesturesBehavior) -> some View`

Determines whether magnify gestures change the view's magnification.

`func webViewOnScrollGeometryChange<T>(for: T.Type, of: (ScrollGeometry) -> T, action: (T, T) -> Void) -> some View`

Adds an action to be performed when a value, created from a scroll geometry, changes.

`func webViewScrollInputBehavior(ScrollInputBehavior, for: ScrollInputKind) -> some View`

Enables or disables scrolling in web views when using particular inputs.

`func webViewScrollPosition(Binding<ScrollPosition>) -> some View`

Associates a binding to a scroll position with the web view.

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

Determines whether to allow people to select or otherwise interact with text.

Accessing Apple Pay and Wallet

`@MainActor @preconcurrency struct PayWithApplePayButton<Fallback> where Fallback : View`

A type that provides a button to pay with Apple pay.

`@MainActor @preconcurrency struct AddPassToWalletButton<Fallback> where Fallback : View`

A type that provides a button that enables people to add a new or existing pass to Apple Wallet.

`@MainActor @preconcurrency struct VerifyIdentityWithWalletButton<Fallback> where Fallback : View`

A type that displays a button to present the identity verification flow.

`func addOrderToWalletButtonStyle(AddOrderToWalletButtonStyle) -> some View`

Sets the button's style.

`func addPassToWalletButtonStyle(AddPassToWalletButtonStyle) -> some View`

Sets the style to be used by the button. (see `PKAddPassButtonStyle`).

```
func onApplePayCouponCodeChange(perform: (String) async -> PKPayment
RequestCouponCodeUpdate) -> some View
```

Called when a user has entered or updated a coupon code. This is required if the user is being asked to provide a coupon code.

```
func onApplePayPaymentMethodChange(perform: (PKPaymentMethod) async ->
PKPaymentRequestPaymentMethodUpdate) -> some View
```

Called when a payment method has changed and asks for an update payment request. If this modifier isn't provided Wallet will assume the payment method is valid.

```
func onApplePayShippingContactChange(perform: (PKContact) async ->
PKPaymentRequestShippingContactUpdate) -> some View
```

Called when a user selected a shipping address. This is required if the user is being asked to provide a shipping contact.

```
func onApplePayShippingMethodChange(perform: (PKShippingMethod) async -
> PKPaymentRequestShippingMethodUpdate) -> some View
```

Called when a user selected a shipping method. This is required if the user is being asked to provide a shipping method.

```
func payLaterViewAction(PayLaterViewAction) -> some View
```

Sets the action on the PayLaterView. See `PKPayLaterAction`.

```
func payLaterViewDisplayStyle(PayLaterViewDisplayStyle) -> some View
```

Sets the display style on the PayLaterView. See `PKPayLaterDisplayStyle`.

```
func payWithApplePayButtonStyle(PayWithApplePayButtonStyle) -> some
View
```

Sets the style to be used by the button. (see `PayWithApplePayButtonStyle`).

```
func verifyIdentityWithWalletButtonStyle(VerifyIdentityWithWalletButton
Style) -> some View
```

Sets the style to be used by the button. (see `PKIdentityButtonStyle`).

```
@MainActor @preconcurrency struct AsyncShareablePassConfiguration<
Content> where Content : View
```

```
func transactionTask(CredentialTransaction.Configuration?, action: (
CredentialTransaction) async -> Void) -> some View
```

Provides a task to perform before this view appears

Authorizing and authenticating

```
@MainActor @preconcurrency struct LocalAuthenticationView<Label> where
Label : View
```

A SwiftUI view that displays an authentication interface.

```
@MainActor @preconcurrency struct SignInWithAppleButton
```

A SwiftUI view that creates the Sign in with Apple button for display.

```
func signInWithAppleButtonStyle(SignInWithAppleButton.Style) -> some
View
```

Sets the style used for displaying the control (see `SignInWithAppleButton.Style`).

```
var authorizationController: AuthorizationController
```

A value provided in the SwiftUI environment that views can use to perform authorization requests.

```
var webAuthenticationSession: WebAuthenticationSession
```

A value provided in the SwiftUI environment that views can use to authenticate a user through a web service.

Configuring Family Sharing

```
@MainActor @preconcurrency struct FamilyActivityPicker
```

A view in which users specify applications, web domains, and categories without revealing their choices to the app.

```
func familyActivityPicker(isPresented: Binding<Bool>, selection:
Binding<FamilyActivitySelection>) -> some View
```

Presents an activity picker view as a sheet.

```
func familyActivityPicker(headerText: String?, footerText: String?, is
Presented: Binding<Bool>, selection: Binding<FamilyActivitySelection>)
-> some View
```

Presents an activity picker view as a sheet.

Reporting on device activity

```
@MainActor @preconcurrency struct DeviceActivityReport
```

A view that reports the user’s application, category, and web domain activity in a privacy-preserving way.

Working with managed devices

`func managedContentStyle(ManagedContentStyle) -> some View`

Applies a managed content style to the view.

`func automatedDeviceEnrollmentAddition(isPresented: Binding<Bool>) -> some View`

Presents a modal view that enables users to add devices to their organization.

Creating graphics

`@MainActor @preconcurrency struct Chart<Content> where Content : ChartContent`

A SwiftUI view that displays a chart.

~~`@MainActor @preconcurrency struct SceneView`~~

A SwiftUI view for displaying 3D SceneKit content.

Deprecated

`@MainActor @preconcurrency struct SpriteView`

A SwiftUI view that renders a SpriteKit scene.

Getting location information

`@MainActor @preconcurrency struct LocationButton`

A SwiftUI button that grants one-time location authorization.

`@MainActor @preconcurrency struct Map<Content> where Content : View`

A view that displays an embedded map interface.

`func mapStyle(MapStyle) -> some View`

Specifies the map style to be used.

`func mapScope(Namespace.ID) -> some View`

Creates a mapScope that SwiftUI uses to connect map controls to an associated map.

`func mapFeatureSelectionDisabled((MapFeature) -> Bool) -> some View`

Specifies which map features should have selection disabled.

`func mapFeatureSelectionAccessory(MapItemDetailSelectionAccessoryStyle?) -> some View`

Specifies the selection accessory to display for a `MapFeature`

`func mapFeatureSelectionContent(content: (MapFeature) -> some MapContent) -> some View`

Specifies a custom presentation for the currently selected feature.

`func mapControls(() -> some View) -> some View`

Configures all `Map` views in the associated environment to have standard size and position controls

`func mapControlVisibility(Visibility) -> some View`

Configures all Map controls in the environment to have the specified visibility

`func mapCameraKeyframeAnimator(trigger: some Equatable, keyframes: (MapCamera) -> some Keyframes<MapCamera>) -> some View`

Uses the given keyframes to animate the camera of a `Map` when the given trigger value changes.

`func lookAroundViewer(isPresented: Binding<Bool>, scene: Binding<MKLookAroundScene?>, allowsNavigation: Bool, showsRoadLabels: Bool, pointsOfInterest: PointOfInterestCategories, onDismiss: (() -> Void)?) -> some View`

`func lookAroundViewer(isPresented: Binding<Bool>, initialScene: MKLookAroundScene?, allowsNavigation: Bool, showsRoadLabels: Bool, pointsOfInterest: PointOfInterestCategories, onDismiss: (() -> Void)?) -> some View`

`func onMapCameraChange(frequency:_:)`

Performs an action when Map camera framing changes

`func mapItemDetailPopover(isPresented: Binding<Bool>, item: MKMapItem?, displaysMap: Bool, attachmentAnchor: PopoverAttachmentAnchor) -> some View`

Presents a map item detail popover.

`func mapItemDetailPopover(isPresented: Binding<Bool>, item: MKMapItem?, displaysMap: Bool, attachmentAnchor: PopoverAttachmentAnchor, arrowEdge: Edge) -> some View`

Presents a map item detail popover.

```
func mapItemDetailPopover(item: Binding<MKMapItem?>, displaysMap: Bool,
attachmentAnchor: PopoverAttachmentAnchor) -> some View
```

Presents a map item detail popover.

```
func mapItemDetailPopover(item: Binding<MKMapItem?>, displaysMap: Bool,
attachmentAnchor: PopoverAttachmentAnchor, arrowEdge: Edge) -> some
View
```

Presents a map item detail popover.

```
func mapItemDetailSheet(isPresented: Binding<Bool>, item: MKMapItem?,
displaysMap: Bool) -> some View
```

Presents a map item detail sheet.

```
func mapItemDetailSheet(item: Binding<MKMapItem?>, displaysMap: Bool) -
> some View
```

Presents a map item detail sheet.

Displaying media

```
@MainActor @preconcurrency struct CameraView
```

A SwiftUI view into which a video stream or an image snapshot is rendered.

```
@MainActor @preconcurrency struct NowPlayingView
```

A view that displays the system’s Now Playing interface so that the user can control audio.

```
@MainActor @preconcurrency struct VideoPlayer<VideoOverlay> where Video
Overlay : View
```

A view that displays content from a player and a native user interface to control playback.

```
func continuityDevicePicker(isPresented: Binding<Bool>, onDidConnect:
((AVContinuityDevice?) -> Void)?) -> some View
```

A `continuityDevicePicker` should be used to discover and connect nearby continuity device through a button interface or other form of activation. On tvOS, this presents a fullscreen continuity device picker experience when selected. The modal view covers as much the screen of `self` as possible when a given condition is true.

```
func cameraAnchor(isActive: Bool) -> some View
```

Specifies the view that should act as the virtual camera for Apple Vision Pro 2D Persona stream.

Selecting photos

```
@MainActor @preconcurrency struct PhotosPicker<Label> where Label : View
```

A view that displays a Photos picker for choosing assets from the photo library.

```
func photosPicker(isPresented: Binding<Bool>, selection: Binding<PhotosPickerItem?>, matching: PHPickerFilter?, preferredItemEncoding: PhotosPickerItem.EncodingDisambiguationPolicy) -> some View
```

Presents a Photos picker that selects a `PhotosPickerItem`.

```
func photosPicker(isPresented: Binding<Bool>, selection: Binding<PhotosPickerItem?>, matching: PHPickerFilter?, preferredItemEncoding: PhotosPickerItem.EncodingDisambiguationPolicy, photoLibrary: PHPhotoLibrary) -> some View
```

Presents a Photos picker that selects a `PhotosPickerItem` from a given photo library.

```
func photosPicker(isPresented: Binding<Bool>, selection: Binding<[PhotosPickerItem]>, maxSelectionCount: Int?, selectionBehavior: PhotosPickerSelectionBehavior, matching: PHPickerFilter?, preferredItemEncoding: PhotosPickerItem.EncodingDisambiguationPolicy) -> some View
```

Presents a Photos picker that selects a collection of `PhotosPickerItem`.

```
func photosPicker(isPresented: Binding<Bool>, selection: Binding<[PhotosPickerItem]>, maxSelectionCount: Int?, selectionBehavior: PhotosPickerSelectionBehavior, matching: PHPickerFilter?, preferredItemEncoding: PhotosPickerItem.EncodingDisambiguationPolicy, photoLibrary: PHPhotoLibrary) -> some View
```

Presents a Photos picker that selects a collection of `PhotosPickerItem` from a given photo library.

```
func photosPickerAccessoryVisibility(Visibility, edges: Edge.Set) -> some View
```

Sets the accessory visibility of the Photos picker. Accessories include anything between the content and the edge, like the navigation bar or the sidebar.

```
func photosPickerDisabledCapabilities(PHPickerCapabilities) -> some View
```

Disables capabilities of the Photos picker.

```
func photosPickerStyle(PhotosPickerStyle) -> some View
```

Sets the mode of the Photos picker.

Previewing content

```
func quickLookPreview(Binding<URL?>) -> some View
```

Presents a Quick Look preview of the contents of a single URL.

```
func quickLookPreview<Items>(Binding<Items.Element?>, in: Items) ->
some View
```

Presents a Quick Look preview of the URLs you provide.

Interacting with networked devices

```
@MainActor @preconcurrency struct DevicePicker<Label, Fallback> where
Label : View, Fallback : View
```

A SwiftUI view that displays other devices on the network, and creates an encrypted connection to a copy of your app running on that device.

```
var devicePickerSupports: DevicePickerSupportedAction
```

Checks for support to present a DevicePicker.

Configuring a Live Activity

```
func activitySystemActionForegroundColor(Color?) -> some View
```

The text color for the auxiliary action button that the system shows next to a Live Activity on the Lock Screen.

```
func activityBackgroundTint(Color?) -> some View
```

Sets the tint color for the background of a Live Activity that appears on the Lock Screen.

```
var isActivityFullscreen: Bool
```

A Boolean value that indicates whether the Live Activity appears in a full-screen presentation.

```
var activityFamily: ActivityFamily
```

The size family of the current Live Activity.

Interacting with the App Store and Apple Music

```
func appStoreOverlay(isPresented: Binding<Bool>, configuration: () ->
SKOverlay.Configuration) -> some View
```

Presents a StoreKit overlay when a given condition is true.

```
func manageSubscriptionsSheet(isPresented: Binding<Bool>) -> some View
func refundRequestSheet(for: Transaction.ID, isPresented: Binding<Bool
>, onDismiss: ((Result<Transaction.RefundRequestStatus, Transaction.
RefundRequestError>) -> ())?) -> some View
```

Display the refund request sheet for the given transaction.

```
func offerCodeRedemption(isPresented: Binding<Bool>, onCompletion: (
Result<Void, any Error>) -> Void) -> some View
```

Presents a sheet that enables customers to redeem offer codes that you configure in App Store Connect.

```
func musicSubscriptionOffer(isPresented: Binding<Bool>, options: Music
SubscriptionOffer.Options, onLoadCompletion: ((any Error)?) -> Void) ->
some View
```

Initiates the process of presenting a sheet with subscription offers for Apple Music when the `isPresented` binding is `true`.

```
func currentEntitlementTask(for: String, priority: TaskPriority, action
: (EntitlementTaskState<VerificationResult<Transaction>?>) async -> ())
-> some View
```

Declares the view as dependent on the entitlement of an In-App Purchase product, and returns a modified view.

```
func inAppPurchaseOptions(((Product) async -> Set<Product.Purchase
Option>)?) -> some View
```

Add a function to call before initiating a purchase from StoreKit view within this view, providing a set of options for the purchase.

```
func manageSubscriptionsSheet(isPresented: Binding<Bool>, subscription
GroupID: String) -> some View
```

```
func onInAppPurchaseCompletion(perform: ((Product, Result<Product.
PurchaseResult, any Error>) async -> ())?) -> some View
```

Add an action to perform when a purchase initiated from a StoreKit view within this view completes.

```
func onInAppPurchaseStart(perform: ((Product) async -> ())?) -> some
View
```

Add an action to perform when a user triggers the purchase button on a StoreKit view within this view.

```
func productIconBorder() -> some View
```

Adds a standard border to an in-app purchase product’s icon .

```
func productViewStyle(some ProductViewStyle) -> some View
```

Sets the style for In-App Purchase product views within a view.

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

Configure the visibility of labels displaying an in-app purchase product description within the view.

```
func storeButton(Visibility, for: StoreButtonKind...) -> some View
```

Specifies the visibility of auxiliary buttons that store view and subscription store view instances may use.

```
func storeProductTask(for: Product.ID, priority: TaskPriority, action:
(Product.TaskState) async -> ()) -> some View
```

Declares the view as dependent on an In-App Purchase product and returns a modified view.

```
func storeProductsTask(for: some Collection<String> & Equatable &
Sendable, priority: TaskPriority, action: (Product.CollectionTaskState)
async -> ()) -> some View
```

Declares the view as dependent on a collection of In-App Purchase products and returns a modified view.

```
func subscriptionStatusTask(for: String, priority: TaskPriority, action
: (EntitlementTaskState<[Product.SubscriptionInfo.Status]>) async ->
()) -> some View
```

Declares the view as dependent on the status of an auto-renewable subscription group, and returns a modified view.

```
func subscriptionStoreButtonLabel(SubscriptionStoreButtonLabel) -> some
 View
```

Configures subscription store view instances within a view to use the provided button label.

```
func subscriptionStoreControlIcon(icon: (Product, Product.Subscription
Info) -> some View) -> some View
```

Sets a view to use to decorate individual subscription options within a subscription store view.

```
func subscriptionStoreControlStyle(some SubscriptionStoreControlStyle)
-> some View
```

Sets the control style for subscription store views within a view.

```
func subscriptionStoreControlStyle<S>(S, placement: S.Placement) ->
some View
```

Sets the control style and control placement for subscription store views within a view.

```
func subscriptionStoreOptionGroupStyle(some SubscriptionOptionGroup
Style) -> some View
```

Sets the style subscription store views within this view use to display groups of subscription options.

```
func subscriptionStorePickerItemBackground(some ShapeStyle) -> some
View
```

Sets the background style for picker items of the subscription store view instances within a view.

```
func subscriptionStorePickerItemBackground(some ShapeStyle, in: some
Shape) -> some View
```

Sets the background shape and style for subscription store view picker items within a view.

```
func subscriptionStorePolicyDestination(for: SubscriptionStorePolicy
Kind, destination: () -> some View) -> some View
```

Configures a view as the destination for a policy button action in subscription store views.

```
func subscriptionStorePolicyDestination(url: URL, for: Subscription
StorePolicyKind) -> some View
```

Configures a URL as the destination for a policy button action in subscription store views.

```
func subscriptionStorePolicyForegroundStyle(some ShapeStyle) -> some
View
```

Sets the style for the terms of service and privacy policy buttons within a subscription store view.

```
func subscriptionStorePolicyForegroundStyle(some ShapeStyle, some Shape
Style) -> some View
```

Sets the primary and secondary style for the terms of service and privacy policy buttons within a subscription store view.

```
func subscriptionStoreSignInAction((() -> ())?) -> some View
```

Adds an action to perform when a person uses the sign-in button on a subscription store view within a view.

```
func subscriptionStoreControlBackground(_:)
```

Set a standard effect to use for the background of subscription store view controls within the view.

~~func subscriptionPromotionalOffer(offer: (Product, Product.SubscriptionInfo) -> Product.SubscriptionOffer?, signature: (Product, Product.SubscriptionInfo, Product.SubscriptionOffer) async throws -> Product.SubscriptionOffer.Signature) -> some View~~

Selects a promotional offer to apply to a purchase a customer makes from a subscription store view.

Deprecated

```
func preferredSubscriptionOffer((Product, Product.SubscriptionInfo, [Product.SubscriptionOffer]) -> Product.SubscriptionOffer?) -> some View
```

Selects a subscription offer to apply to a purchase that a customer makes from a subscription store view, a store view, or a product view.

Accessing health data

```
func healthDataAccessRequest(store: HKHealthStore, objectType: HKObjectType, predicate: NSPredicate?, trigger: some Equatable, completion: (Result<Bool, any Error>) -> Void) -> some View
```

Asynchronously requests permission to read a data type that requires per-object authorization (such as vision prescriptions).

```
func healthDataAccessRequest(store: HKHealthStore, readTypes: Set<HKObjectType>, trigger: some Equatable, completion: (Result<Bool, any Error>) -> Void) -> some View
```

Requests permission to read the specified HealthKit data types.

```
func healthDataAccessRequest(store: HKHealthStore, shareTypes: Set<HKSampleType>, readTypes: Set<HKObjectType>?, trigger: some Equatable, completion: (Result<Bool, any Error>) -> Void) -> some View
```

Requests permission to save and read the specified HealthKit data types.

```
func workoutPreview(WorkoutPlan, isPresented: Binding<Bool>) -> some View
```

Presents a preview of the workout contents as a modal sheet

Providing tips

```
func popoverTip((any Tip)?, arrowEdge: Edge?, action: (Tips.Action) -> Void) -> some View
```

Presents a popover tip on the modified view.

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

Sets the tip's view background to a style. Currently this only applies to inline tips, not popover tips.

```
func tipCornerRadius(CGFloat, antialiased: Bool) -> some View
```

Sets the corner radius for an inline tip view.

```
func tipImageSize(CGSize) -> some View
```

Sets the size for a tip's image.

```
func tipViewStyle(some TipViewStyle) -> some View
```

Sets the given style for TipView within the view hierarchy.

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

Sets the style for a tip's image.

```
func tipImageStyle<S1, S2>(S1, S2) -> some View
```

Sets the style for a tip's image.

```
func tipImageStyle<S1, S2, S3>(S1, S2, S3) -> some View
```

Sets the style for a tip's image.

Showing a translation

```
func translationPresentation(isPresented: Binding<Bool>, text: String,
attachmentAnchor: PopoverAttachmentAnchor, arrowEdge: Edge, replacement
Action: ((String) -> Void)?) -> some View
```

Presents a translation popover when a given condition is true.

```
func translationTask(TranslationSession.Configuration?, action: (
TranslationSession) async -> Void) -> some View
```

Adds a task to perform before this view appears or when the translation configuration changes.

```
func translationTask(source: Locale.Language?, target: Locale.Language
?, action: (TranslationSession) async -> Void) -> some View
```

Adds a task to perform before this view appears or when the specified source or target languages change.

Presenting journaling suggestions

```
func journalingSuggestionsPicker(isPresented: Binding<Bool>, on
Completion: (JournalingSuggestion) async -> Void) -> some View
```

Presents a visual picker interface that contains events and images that a person can select to retrieve more information.

Managing contact access

```
func contactAccessButtonCaption(ContactAccessButton.Caption) -> some
View
```

```
func contactAccessButtonStyle(ContactAccessButton.Style) -> some View
```

```
func contactAccessPicker(isPresented: Binding<Bool>, completionHandler:
([String]) -> ()) -> some View
```

Modally present UI which allows the user to select which contacts your app has access to.

Handling game controller events

```
func handlesGameControllerEvents(matching: GCUIEventTypes) -> some View
```

Specifies the game controllers events which should be delivered through the GameController framework when the view, or one of its descendants has focus.

Creating a tabletop game

```
func tabletopGame(TabletopGame, parent: Entity, automaticUpdate: Bool)
-> some View
```

Adds a tabletop game to a view.

```
func tabletopGame(TabletopGame, parent: Entity, automaticUpdate: Bool,
interaction: (TabletopInteraction.Value) -> any TabletopInteraction.
Delegate) -> some View
```

Supplies a closure which returns a new interaction whenever needed.

Configuring camera controls

```
var realityViewCameraControls: CameraControls
```

The camera controls for the reality view.

```
func realityViewCameraControls(CameraControls) -> some View
```

Adds gestures that control the position and direction of a virtual camera.

Interacting with transactions

```
func transactionPicker(isPresented: Binding<Bool>, selection: Binding<[
Transaction]>) -> some View
```

Presents a picker that selects a collection of transactions.

See Also

Framework integration

- AppKit integration

  Add AppKit views to your SwiftUI app, or use SwiftUI views in your AppKit app.

- UIKit integration

  Add UIKit views to your SwiftUI app, or use SwiftUI views in your UIKit app.

- WatchKit integration

  Add WatchKit views to your SwiftUI app, or use SwiftUI views in your WatchKit app.