Protocol

View

A type that represents part of your app's user interface and provides modifiers that you use to configure views.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | tvOS 13.0+ | visionOS 1.0+ | watchOS 6.0+

```
@MainActor @preconcurrency
protocol View
```

Mentioned in

- Declaring a custom view
- Configuring views
- Reducing view modifier maintenance
- Displaying data in lists
- Migrating to the SwiftUI life cycle

Overview

You create custom views by declaring types that conform to the `View` protocol. Implement the required `body` computed property to provide the content for your custom view.

```
struct MyView: View {
    var body: some View {
        Text("Hello, World!")
    }
}
```

Assemble the view's body by combining one or more of the built-in views provided by SwiftUI, like the `Text` instance in the example above, plus other custom views that you define, into a hierarchy of views. For more information about creating custom views, see Declaring a custom view.

The `View` protocol provides a set of modifiers — protocol methods with default implementations — that you use to configure views in the layout of your app. Modifiers work by wrapping the view instance on which you call them in another view with the specified characteristics, as described in Configuring views. For example, adding the `opacity(_:)` modifier to a text view returns a new view with some amount of transparency:

```
Text("Hello, World!")
    .opacity(0.5) // Display partially transparent text.
```

The complete list of default modifiers provides a large set of controls for managing views. For example, you can fine tune Layout modifiers, add Accessibility modifiers information, and respond to Input and event modifiers. You can also collect groups of default modifiers into new, custom view modifiers for easy reuse.

A type conforming to this protocol inherits `@preconcurrency @MainActor` isolation from the protocol if the conformance is declared in its original declaration. Isolation to the main actor is the default, but it's not required. Declare the conformance in an extension to opt-out the isolation.

Topics

Implementing a custom view

`var body: Self.Body`

The content and behavior of the view.

Required Default implementations provided.

`associatedtype Body : View`

The type of view representing the body of this view.

Required

`func modifier<T>(T) -> ModifiedContent<Self, T>`

Applies a modifier to a view and returns a new view.

Previews in Xcode

Generate dynamic, interactive previews of your custom views.

Configuring view elements

- Accessibility modifiers

  Make your SwiftUI apps accessible to everyone, including people with disabilities.

- Appearance modifiers

  Configure a view's foreground and background styles, controls, and visibility.

- Text and symbol modifiers

  Manage the rendering, selection, and entry of text in your view.

- Auxiliary view modifiers

  Add and configure supporting views, like toolbars and context menus.

- Chart view modifiers

  Configure charts that you declare with Swift Charts.

Drawing views

- Style modifiers

  Apply built-in styles to different types of views.

- Layout modifiers

  Tell a view how to arrange itself within a view hierarchy by adjusting its size, position, alignment, padding, and so on.

- Graphics and rendering modifiers

  Affect the way the system draws a view, for example by scaling or masking a view, or by applying graphical effects.

Providing interactivity

- Input and event modifiers

  Supply actions for a view to perform in response to user input and system events.

- Search modifiers

  Enable people to search for content in your app.

- Presentation modifiers

  Define additional views for the view to present under specified conditions.

- State modifiers

  Access storage and provide child views with configuration data.

Deprecated modifiers

Deprecated modifiers

Review unsupported modifiers and their replacements.

Instance Methods

`func accessibilityActions<Content>(category: AccessibilityActionCategory, () -> Content) -> some View`

Adds multiple accessibility actions to the view with a specific category. Actions allow assistive technologies, such as VoiceOver, to interact with the view by invoking the action and are grouped by their category. When multiple action modifiers with an equal category are applied to the view, the actions are combined together.

`func accessibilityDefaultFocus<Value>(AccessibilityFocusState<Value>.Binding, Value) -> some View`

Defines a region in which default accessibility focus is evaluated by assigning a value to a given accessibility focus state binding.

`func accessibilityScrollStatus(_:isEnabled:)`

Changes the announcement provided by accessibility technologies when a user scrolls a scroll view within this view.

`func accessoryWidgetGroupStyle(AccessoryWidgetGroupStyle) -> some View`

The view modifier that can be applied to `AccessoryWidgetGroup` to specify the shape the three content views will be masked with. The value of `style` is set to `.automatic`, which is `.circular` by default.

`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 allowsWindowActivationEvents() -> some View`

Configures gestures in this view hierarchy to handle events that activate the containing window.

```
func appStoreMerchandising(isPresented: Binding<Bool>, kind: AppStore
MerchandisingKind, onDismiss: ((Result<AppStoreMerchandisingKind.
PresentationResult, any Error>) async -> ())?) -> some View
```

```
func aspectRatio3D(Size3D?, contentMode: ContentMode) -> some View
```

Constrains this view's dimensions to the specified 3D aspect ratio.

```
func assistiveAccessNavigationIcon(Image) -> some View
```

Configures the view's icon for purposes of navigation.

```
func assistiveAccessNavigationIcon(systemImage: String) -> some View
```

Configures the view's icon for purposes of navigation.

```
func attributedTextFormattingDefinition(_:)
```

Apply a text formatting definition to nested views.

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

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

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

Adds the background extension effect to the view. The view will be duplicated into mirrored copies which will be placed around the view on any edge with available safe area. Additionally, a blur effect will be applied on top to blur out the copies.

```
func backgroundExtensionEffect(isEnabled: Bool) -> some View
```

Adds the background extension effect to the view. The view will be duplicated into mirrored copies which will be placed around the view on any edge with available safe area. Additionally, a blur effect will be applied on top to blur out the copies.

```
func breakthroughEffect(BreakthroughEffect) -> some View
```

Ensures that the view is always visible to the user, even when other content is occluding it, like 3D models.

```
func buttonSizing(ButtonSizing) -> some View
```

The preferred sizing behavior of buttons in the view hierarchy.

```
func certificateSheet(trust: Binding<SecTrust?>, title: String?,
message: String?, help: URL?) -> some View
```

Displays a certificate sheet using the provided certificate trust.

```
func chart3DCameraProjection(Chart3DCameraProjection) -> some View
```

```
func chart3DPose(_:)
```

Associates a binding to be updated when the 3D chart's pose is changed by an interaction.

```
func chart3DRenderingStyle(Chart3DRenderingStyle) -> some View
```

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

Sets the visibility of the z axis.

```
func chartZAxis<Content>(content: () -> Content) -> some View
```

Configures the z-axis for 3D charts in the view.

```
func chartZAxisLabel(_:position:alignment:spacing:)
```

Adds z axis label for charts in the view. It effects 3D charts only.

```
func chartZScale<Domain, Range>(domain: Domain, range: Range, type:
ScaleType?) -> some View
```

Configures the z scale for 3D charts.

```
func chartZScale<Domain>(domain: Domain, type: ScaleType?) -> some View
```

Configures the z scale for 3D charts.

```
func chartZScale<Range>(range: Range, type: ScaleType?) -> some View
```

Configures the z scale for 3D charts.

```
func chartZSelection<P>(range: Binding<ClosedRange<P>?>) -> some View
```

```
func chartZSelection<P>(value: Binding<P?>) -> some View
```

```
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.

```
func containerCornerOffset(Edge.Set, sizeToFit: Bool) -> some View
```

Adjusts the view's layout to avoid the container view's corner insets for the specified edges.

```
func containerValue<V>(WritableKeyPath<ContainerValues, V>, V) -> some
View
```

Sets a particular container value of a view.

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

`func contentToolbar(for:content:)`

Populates the toolbar of the specified content view type with the views you provide.

`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 controlWidgetActionHint(_:)`

The action hint of the control described by the modified label.

`func controlWidgetStatus(_:)`

The status of the control described by the modified label.

`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 dialogPreventsAppTermination(Bool?) -> some View`

Whether the alert or confirmation dialog prevents the app from being quit/terminated by the system or app termination menu item.

`func documentBrowserContextMenu(([URL]?) -> some View) -> some View`

Adds to a `DocumentLaunchView` actions that accept a list of selected files as their parameter.

`func dragConfiguration(DragConfiguration) -> some View`

Configures a drag session.

`func dragContainer(for:in:_:)`

A container with draggable views where the drag payload is based on multiple identifiers of dragged items.

`func dragContainer(for:itemID:in:_:)`

A container with draggable views.

`func dragContainerSelection<ItemID>(@autoclosure () -> Array<ItemID>, containerNamespace: Namespace.ID?) -> some View`

Provides multiple item selection support for drag containers.

```
func dragPreviewsFormation(DragDropPreviewsFormation) -> some View
```

Describes the way dragged previews are visually composed.

```
func draggable<Item>(Item.Type, containerNamespace: Namespace.ID?, () -> Item?) -> some View
```

Activates this view as the source of a drag and drop operation, allowing to provide optional identifiable payload and specify the namespace of the drag container this view belongs to.

```
func draggable<Item, ItemID>(Item.Type, id: KeyPath<Item, ItemID>, containerNamespace: Namespace.ID?, () -> Item?) -> some View
```

Activates this view as the source of a drag and drop operation, allowing to provide optional payload and specify the namespace of the drag container this view belongs to.

```
func draggable<Item, ItemID>(Item.Type, id: KeyPath<Item, ItemID>, item: @autoclosure () -> Item?, containerNamespace: Namespace.ID?) -> some View
```

Activates this view as the source of a drag and drop operation, allowing to provide optional payload and specify the namespace of the drag container this view belongs to.

```
func draggable<Item>(Item.Type, item: @autoclosure () -> Item?, containerNamespace: Namespace.ID?) -> some View
```

Activates this view as the source of a drag and drop operation, allowing to provide optional identifiable payload and specify the namespace of the drag container this view belongs to.

```
func draggable<ItemID>(containerItemID: ItemID, containerNamespace: Namespace.ID?) -> some View
```

Inside a drag container, activates this view as the source of a drag and drop operation. Supports lazy drag containers.

```
func dropConfiguration((DropSession) -> DropConfiguration) -> some View
```

Configures a drop session.

```
func dropDestination<T>(for: T.Type, isEnabled: Bool, action: ([T], DropSession) -> Void) -> some View
```

Defines the destination of a drag and drop operation that provides a drop operation proposal and handles the dropped content with a closure that you specify.

```
func dropPreviewsFormation(DragDropPreviewsFormation) -> some View
```

Describes the way previews for a drop are composed.

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

Sets the style for forms in a view hierarchy.

```
func gameSaveSyncingAlert(directory: Binding<GameSaveSyncedDirectory?>, finishedLoading: () -> Void) -> some View
```

Presents a modal view while the game synced directory loads.

```
func glassBackgroundEffect<S>(S, displayMode: GlassBackgroundDisplayMode) -> some View
```

Fills the view's background with a custom glass background effect and container-relative rounded rectangle shape.

```
func glassBackgroundEffect<T, S>(S, in: T, displayMode: GlassBackgroundDisplayMode) -> some View
```

Fills the view's background with a custom glass background effect and a shape that you specify.

```
func glassEffect(Glass, in: some Shape) -> some View
```

Applies the Liquid Glass effect to a view.

```
func glassEffectID((some Hashable & Sendable)?, in: Namespace.ID) -> some View
```

Associates an identity value to Liquid Glass effects defined within this view.

```
func glassEffectTransition(GlassEffectTransition) -> some View
```

Associates a glass effect transition with any glass effects defined within this view.

```
func glassEffectUnion(id: (some Hashable & Sendable)?, namespace: Namespace.ID) -> some View
```

Associates any Liquid Glass effects defined within this view to a union with the provided identifier.

```
func groupActivityAssociation(GroupActivityAssociationKind?) -> some View
```

Specifies how a view should be associated with the current SharePlay group activity.

```
func handGestureShortcut(HandGestureShortcut, isEnabled: Bool) -> some View
```

Assigns a hand gesture shortcut to the modified control.

```
func handPointerBehavior(HandPointerBehavior?) -> some View
```

Sets the behavior of the hand pointer while the user is interacting with the view.

```
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.

```
func handlesGameControllerEvents(matching: GCUIEventTypes, withOptions: GameControllerEventHandlingOptions?) -> 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.

```
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 imagePlaygroundGenerationStyle(ImagePlaygroundStyle, in: [ImagePlaygroundStyle]) -> some View
```

Sets the generation style for an image playground.

```
func imagePlaygroundPersonalizationPolicy(ImagePlaygroundPersonalizationPolicy) -> some View
```

Policy determining whether to support the usage of people in the playground or not.

```
func imagePlaygroundSheet(isPresented: Binding<Bool>, concept: String, sourceImage: Image?, onCompletion: (URL) -> Void, onCancellation: (() -> Void)?) -> some View
```

Presents the system sheet to create images from the specified input.

```
func imagePlaygroundSheet(isPresented: Binding<Bool>, concept: String, sourceImageURL: URL, onCompletion: (URL) -> Void, onCancellation: (() -> Void)?) -> some View
```

Presents the system sheet to create images from the specified input.

```
func imagePlaygroundSheet(isPresented: Binding<Bool>, concepts: [Image
PlaygroundConcept], sourceImage: Image?, onCompletion: (URL) -> Void,
onCancellation: (() -> Void)?) -> some View
```

Presents the system sheet to create images from the specified input.

```
func imagePlaygroundSheet(isPresented: Binding<Bool>, concepts: [Image
PlaygroundConcept], sourceImageURL: URL, onCompletion: (URL) -> Void,
onCancellation: (() -> Void)?) -> some View
```

Presents the system sheet to create images from the specified input.

```
func immersiveEnvironmentPicker<Content>(content: () -> Content) ->
some View
```

Add menu items to open immersive spaces from a media player's environment picker.

```
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 journalingSuggestionsPicker(isPresented: Binding<Bool>, journaling
SuggestionToken: JournalingSuggestionPresentationToken?, onCompletion:
(JournalingSuggestion) async -> Void) -> some View
```

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

```
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.

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

Set the spacing between the icon and title in labels.

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

Set the width reserved for icons in labels.

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

Sets a style for labeled content.

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

Controls the visibility of labels of any controls contained within this view.

```
func lineHeight(AttributedString.LineHeight?) -> some View
```

A modifier for the default line height in the view hierarchy.

```
func listRowInsets(Edge.Set, CGFloat?) -> some View
```

Sets the insets of rows in a list on the specified edges.

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

Changes the visibility of the list section index.

```
func listSectionMargins(Edge.Set, CGFloat?) -> some View
```

Set the section margins for the specific edges.

```
func lookAroundViewer(isPresented: Binding<Bool>, initialScene: MKLook
AroundScene?, allowsNavigation: Bool, showsRoadLabels: Bool, pointsOf
Interest: PointOfInterestCategories, onDismiss: (() -> Void)?) -> some
View
```

```
func lookAroundViewer(isPresented: Binding<Bool>, scene: Binding<MKLook
AroundScene?>, allowsNavigation: Bool, showsRoadLabels: Bool, pointsOf
Interest: PointOfInterestCategories, onDismiss: (() -> Void)?) -> some
View
```

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

```
func managedContentStyle(ManagedContentStyle) -> some View
```

Applies a managed content style to the view.

```
func manipulable(coordinateSpace: some CoordinateSpaceProtocol,
operations: Manipulable.Operation.Set, inertia: Manipulable.Inertia, is
Enabled: Bool, onChanged: ((Manipulable.Event) -> Void)?) -> some View
```

Allows this view to be manipulated using common hand gestures.

```
func manipulable(transform: Binding<AffineTransform3D>, coordinateSpace
: some CoordinateSpaceProtocol, operations: Manipulable.Operation.Set,
inertia: Manipulable.Inertia, isEnabled: Bool, onChanged: ((Manipulable
.Event) -> Void)?) -> some View
```

Applies the given 3D affine transform to the view and allows it to be manipulated using common hand gestures.

```
func manipulable(using: Manipulable.GestureState) -> some View
```

Allows the view to be manipulated using a manipulation gesture attached to a different view.

```
func manipulationGesture(updating: Binding<Manipulable.GestureState>, coordinateSpace: some CoordinateSpaceProtocol, operations: Manipulable.Operation.Set, inertia: Manipulable.Inertia, isEnabled: Bool, onChanged: ((Manipulable.Event) -> Void)?) -> some View
```

Adds a manipulation gesture to this view without allowing this view to be manipulable itself.

```
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 mapControlVisibility(Visibility) -> some View
```

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

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

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

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

Specifies which map features should have selection disabled.

```
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.

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

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

```
func mapStyle(MapStyle) -> some View
```

Specifies the map style to be used.

```
func matchedTransitionSource(id: some Hashable, in: Namespace.ID) -> some View
```

Identifies this view as the source of a navigation transition, such as a zoom transition.

```
func matchedTransitionSource(id: some Hashable, in: Namespace.ID, configuration: (EmptyMatchedTransitionSourceConfiguration) -> some MatchedTransitionSourceConfiguration) -> some View
```

Identifies this view as the source of a navigation transition, such as a zoom transition.

```
func materialActiveAppearance(MaterialActiveAppearance) -> some View
```

Sets an explicit active appearance for materials in this view.

```
func multilineTextAlignment(strategy: Text.AlignmentStrategy) -> some View
```

A modifier for the default text alignment strategy in the view hierarchy.

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

Configures whether navigation links show a disclosure indicator.

```
func navigationTransition(some NavigationTransition) -> some View
```

Sets the navigation transition style for this view.

```
func onAppIntentExecution<I>(I.Type, perform: (I) -> Void) -> some View
```

Registers a handler to invoke in response to the specified app intent that your app receives.

`func onApplePayCouponCodeChange(perform: (String) async -> PKPaymentRequestCouponCodeUpdate) -> 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 onCameraCaptureEvent(isEnabled:defaultSoundDisabled:action:)`

Used to register an action triggered by system capture events.

`func onCameraCaptureEvent(isEnabled:defaultSoundDisabled:primaryAction:secondaryAction:)`

Used to register actions triggered by system capture events.

`func onDragSessionUpdated((DragSession) -> Void) -> some View`

Specifies an action to perform on each update of an ongoing dragging operation activated by `draggable(_:)` or anther drag modifiers.

`func onDropSessionUpdated((DropSession) -> Void) -> some View`

Specifies an action to perform on each update of an ongoing drop operation activated by `dropDestination(_:)` or other drop modifiers.

`func onGeometryChange3D(for:of:action:)`

Returns a new view that arranges to call `action(value)` whenever the value computed by `transform(proxy)` changes, where `proxy` provides access to the view's 3D geometry properties.

```
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 onInteractiveResizeChange((Bool) -> Void) -> some View
```

Adds an action to perform when the enclosing window is being interactively resized.

```
func onMapCameraChange(frequency:_:)
```

Performs an action when Map camera framing changes

```
func onOpenURL(prefersInApp: Bool) -> some View
```

Sets an `OpenURLAction` that prefers opening URL with an in-app browser. It's equivalent to calling `.onOpenURL(_:)`

```
func onWorldRecenter(action:)
```

Adds an action to perform when recentering the view with the digital crown.

```
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 payWithApplePayButtonDisableCardArt() -> some View
```

Sets the features that should be allowed to show on the payment buttons.

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

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

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

Presents a popover tip on the modified view.

```
func popoverTip((any Tip)?, isPresented: Binding<Bool>?, attachment
Anchor: PopoverAttachmentAnchor, arrowEdge: Edge?, action: (Tips.Action
) -> Void) -> some View
```

Presents a popover tip on the modified view.

```
func popoverTip((any Tip)?, isPresented: Binding<Bool>?, attachment
Anchor: PopoverAttachmentAnchor, arrowEdges: Edge.Set, action: (Tips.
Action) -> Void) -> some View
```

Presents a popover tip on the modified view.

```
func postToPhotosSharedAlbumSheet(isPresented:items:photoLibrary:
defaultAlbumIdentifier:completion:)
```

Presents an “Add to Shared Album” sheet that allows the user to post the given items to a shared album.

```
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.

```
func preferredWindowClippingMargins(_:_:)
```

Requests additional margins for drawing beyond the bounds of the window.

```
func presentationBreakthroughEffect(BreakthroughEffect) -> some View
```

Changes the way the enclosing presentation breaks through content occluding it.

```
func presentationPreventsAppTermination(Bool?) -> some View
```

Whether a presentation prevents the app from being terminated/quit by the system or app termination menu item.

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

Configure the visibility of labels displaying an in-app purchase product description within the 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 realityViewCameraControls(CameraControls) -> some View
```

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

`func realityViewLayoutBehavior(RealityViewLayoutOption) -> some View`

A view modifier that controls the frame sizing and content alignment behavior for `RealityView`

`func rotation3DLayout(Rotation3D) -> some View`

Rotates a view with impacts to its frame in a containing layout

`func rotation3DLayout(_:axis:)`

Rotates a view with impacts to its frame in a containing layout

`func safeAreaBar(edge:alignment:spacing:content:)`

Shows the specified content as a custom bar beside the modified view.

`func scaledToFill3D() -> some View`

Scales this view to fill its parent.

`func scaledToFit3D() -> some View`

Scales this view to fit its parent.

`func scrollEdgeEffectHidden(Bool, for: Edge.Set) -> some View`

Hides any scroll edge effects for scroll views within this hierarchy.

`func scrollEdgeEffectStyle(ScrollEdgeEffectStyle?, for: Edge.Set) -> some View`

Configures the scroll edge effect style for scroll views within this hierarchy.

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

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

`func searchSelection(Binding<TextSelection?>) -> some View`

Binds the selection of the search field associated with the nearest searchable modifier to the given `TextSelection` value.

`func searchToolbarBehavior(SearchToolbarBehavior) -> some View`

Configures the behavior for search in the toolbar.

`func sectionIndexLabel(_:)`

Sets the label that is used in a section index to point to this section, typically only a single character long.

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

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

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

Sets the thumb visibility for `Sliders` within this view.

```
func spatialOverlay<V>(alignment: Alignment3D, content: () -> V) -> some View
```

Adds secondary views within the 3D bounds of this view.

```
func spatialOverlayPreferenceValue<K, V>(K.Type, alignment: Alignment3D, (K.Value) -> V) -> some View
```

Uses the specified preference value from the view to produce another view occupying the same 3D space of the first 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 subscriptionIntroductoryOffer(applyOffer: (Product, Product.SubscriptionInfo) -> Bool, compactJWS: (Product, Product.SubscriptionInfo) async throws -> String) -> some View
```

Selects the introductory offer eligibility preference to apply to a purchase a customer makes from a subscription store view.

```
func subscriptionOfferViewButtonVisibility(Visibility, for: SubscriptionOfferViewButtonKind...) -> some View
```

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

```
func subscriptionOfferViewStyle(some SubscriptionOfferViewStyle) -> some View
```

```
func subscriptionPromotionalOffer(offer: (Product, Product.Subscription
Info) -> Product.SubscriptionOffer?, compactJWS: (Product, Product.
SubscriptionInfo, Product.SubscriptionOffer) async throws -> String) ->
some View
```

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

~~`func subscriptionPromotionalOffer(offer: (Product, Product.Subscription Info) -> 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 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 subscriptionStoreControlBackground(_:)
```

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

```
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 SubscriptionOptionGroupStyle) -> 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: SubscriptionStorePolicyKind, 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: SubscriptionStorePolicyKind) -> 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 ShapeStyle) -> 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 symbolColorRenderingMode(SymbolColorRenderingMode?) -> some View`

Sets the color rendering mode for symbol images.

`func symbolVariableValueMode(SymbolVariableValueMode?) -> some View`

Sets the variable value mode mode for symbol images within this view.

```
func tabBarMinimizeBehavior(TabBarMinimizeBehavior) -> some View
```

Sets the behavior for tab bar minimization.

```
func tabViewBottomAccessory<Content>(content: () -> Content) -> some View
```

Places a view as the bottom accessory of the tab view.

```
func tabViewBottomAccessory<Content>(isEnabled: Bool, content: () -> Content) -> some View
```

Places a view as the bottom accessory of the tab view. Use this modifier to dynamically show and hide the accessory view.

```
func tabViewSearchActivation(TabSearchActivation) -> some View
```

Configures the activation and deactivation behavior of search in the search tab.

```
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.

```
func task<T>(id: T, name: String?, executorPreference: any TaskExecutor, priority: TaskPriority, file: String, line: Int, sending () async -> Void) -> some View
```

Adds a task to perform before this view appears or when a specified value changes.

Beta

```
func textContentType(_:)
```

Sets the text content type for this view, which the system uses to offer suggestions while the user enters text on macOS.

```
func textInputFormattingControlVisibility(Visibility, for: TextInputFormattingControlPlacement.Set) -> some View
```

Define which system text formatting controls are available.

```
func textRenderer<T>(T) -> some View
```

Returns a new view such that any text views within it will use `renderer` to draw themselves.

```
func textSelectionAffinity(TextSelectionAffinity) -> some View
```

Sets the direction of a selection or cursor relative to a text character.

`func tipAnchor<AnchorID>(AnchorID) -> some View`

Sets a value for the specified tip anchor to be used to anchor a tip view to the `.bounds` of the 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 tipBackgroundInteraction(PresentationBackgroundInteraction) -> some View`

Controls whether people can interact with the view behind a presented tip.

`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 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.

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

Sets the given style for TipView within the view hierarchy.

`func toolbarItemHidden(Bool) -> some View`

Hides an individual view within a control group toolbar item.

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

Presents a picker that selects a collection of transactions.

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

Provides a task to perform before this view appears

```
func translationPresentation(isPresented: Binding<Bool>, text: String, attachmentAnchor: PopoverAttachmentAnchor, arrowEdge: Edge, replacementAction: ((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.

```
func verifyIdentityWithWalletButtonStyle(VerifyIdentityWithWalletButtonStyle) -> some View
```

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

```
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.

`func windowResizeAnchor(UnitPoint?) -> some View`

Sets the window anchor point used when the size of the view changes such that the window must resize.

`func windowToolbarFullScreenVisibility(WindowToolbarFullScreenVisibility) -> some View`

Configures the visibility of the window toolbar when the window enters full screen mode.

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

Presents a preview of the workout contents as a modal sheet

`func writingDirection(strategy: Text.WritingDirectionStrategy) -> some View`

A modifier for the default text writing direction strategy in the view hierarchy.

`func writingToolsAffordanceVisibility(Visibility) -> some View`

Specifies whether the system should show the Writing Tools affordance for text input views affected by the environment.

`func writingToolsBehavior(WritingToolsBehavior) -> some View`

Specifies the Writing Tools behavior for text and text input in the environment.

Relationships

Inherited By

DynamicViewContent
InsettableShape
NSViewControllerRepresentable
NSViewRepresentable
RoundedRectangularShape
Shape
ShapeView
UIViewControllerRepresentable
UIViewRepresentable
WKInterfaceObjectRepresentable

Conforming Types

AngularGradient
AnyShape
AnyView
AsyncImage
Button
ButtonBorderShape
ButtonStyleConfiguration.Label
Canvas

Conforms when `Symbols` conforms to `View`.

Capsule
Circle
Color
ColorPicker
ConcentricRectangle
ContainerRelativeShape
ContentUnavailableView
ControlGroup
ControlGroupStyleConfiguration.Content
ControlGroupStyleConfiguration.Label
DatePicker
DatePickerStyleConfiguration.Label
DebugReplaceableView
DefaultButtonLabel
DefaultDateProgressLabel
DefaultDocumentGroupLaunchActions
DefaultGlassEffectShape

DefaultSettingsLinkLabel
DefaultShareLinkLabel
DefaultTabLabel
DefaultWindowVisibilityToggleLabel
DisclosureGroup
DisclosureGroupStyleConfiguration.Content
DisclosureGroupStyleConfiguration.Label
Divider
DocumentLaunchView
EditButton
EditableCollectionContent

Conforms when `Content` conforms to `View`, `Data` conforms to `Copyable`, and `Data` conforms to `Escapable`.

Ellipse
EllipticalGradient
EmptyView
EquatableView
FillShapeView
ForEach

Conforms when `Data` conforms to `RandomAccessCollection`, `ID` conforms to `Hashable`, and `Content` conforms to `View`.

Form
FormStyleConfiguration.Content
Gauge
GaugeStyleConfiguration.CurrentValueLabel
GaugeStyleConfiguration.Label
GaugeStyleConfiguration.MarkedValueLabel
GaugeStyleConfiguration.MaximumValueLabel
GaugeStyleConfiguration.MinimumValueLabel
GeometryReader
GeometryReader3D
GlassBackgroundEffectConfiguration.Content
GlassEffectContainer
Grid

Conforms when `Content` conforms to `View`.

GridRow

Conforms when `Content` conforms to `View`.

Group

Conforms when `Content` conforms to `View`.

GroupBox
GroupBoxStyleConfiguration.Content
GroupBoxStyleConfiguration.Label

GroupElementsOfContent
GroupSectionsOfContent
HSplitView
HStack
HelpLink
Image
KeyframeAnimator
Label
LabelStyleConfiguration.Icon
LabelStyleConfiguration.Title
LabeledContent

Conforms when `Label` conforms to `View` and `Content` conforms to `View`.

LabeledContentStyleConfiguration.Content
LabeledContentStyleConfiguration.Label
LabeledControlGroupContent
LabeledToolbarItemGroupContent
LazyHGrid
LazyHStack
LazyVGrid
LazyVStack
LinearGradient
Link
List
Menu
MenuButton
MenuStyleConfiguration.Content
MenuStyleConfiguration.Label
MeshGradient
ModifiedContent

Conforms when `Content` conforms to `View` and `Modifier` conforms to `ViewModifier`.

MultiDatePicker
NavigationLink
NavigationSplitView
NavigationStack
NavigationView
NewDocumentButton
OffsetShape
OutlineGroup

Conforms when `Data` conforms to `RandomAccessCollection`, `ID` conforms to `Hashable`, `Parent` conforms to `View`, `Leaf` conforms to `View`, and `Subgroup` conforms to `View`.

OutlineSubgroupChildren
PasteButton

Path
PhaseAnimator
Picker
PlaceholderContentView
PresentedWindowContent
PreviewModifierContent
PrimitiveButtonStyleConfiguration.Label
ProgressView
ProgressViewStyleConfiguration.CurrentValueLabel
ProgressViewStyleConfiguration.Label
RadialGradient
Rectangle
RenameButton
RotatedShape
RoundedRectangle
ScaledShape
ScrollView
ScrollViewReader
SearchUnavailableContent.Actions
SearchUnavailableContent.Description
SearchUnavailableContent.Label
Section

Conforms when `Parent` conforms to `View`, `Content` conforms to `View`, and `Footer` conforms to `View`.

SectionConfiguration.Actions
SecureField
SettingsLink
ShareLink
Slider
Spacer
Stepper
StrokeBorderShapeView
StrokeShapeView
SubscriptionView
Subview
SubviewsCollection
SubviewsCollectionSlice
TabContentBuilder.Content
TabView
Table
Text
TextEditor
TextField
TextFieldLink

`TimelineView`

Conforms when `Schedule` conforms to `TimelineSchedule` and `Content` conforms to `View`.

`Toggle`
`ToggleStyleConfiguration.Label`
`TransformedShape`
`TupleView`
`UnevenRoundedRectangle`
`VSplitView`
`VStack`
`ViewThatFits`
`WindowVisibilityToggle`
`ZStack`
`ZStackContent3D`

Conforms when `Content` conforms to `View`.

See Also

Creating a view

Declaring a custom view

Define views and assemble them into a view hierarchy.

`struct ViewBuilder`

A custom parameter attribute that constructs views from closures.