API Collection

Input events

Respond to input from a hardware device, like a keyboard or a Touch Bar.

Overview

SwiftUI provides view modifiers that enable your app to listen for and react to various kinds of user input. For example, you can create keyboard shortcuts, respond to a form submission, or take input from the digital crown of an Apple Watch.

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

Topics

Responding to keyboard input

```
func onKeyPress(KeyEquivalent, action: () -> KeyPress.Result) -> some View
```

Performs an action if the user presses a key on a hardware keyboard while the view has focus.

```
func onKeyPress(phases: KeyPress.Phases, action: (KeyPress) -> KeyPress.Result) -> some View
```

Performs an action if the user presses any key on a hardware keyboard while the view has focus.

```
func onKeyPress(KeyEquivalent, phases: KeyPress.Phases, action: (KeyPress) -> KeyPress.Result) -> some View
```

Performs an action if the user presses a key on a hardware keyboard while the view has focus.

```
func onKeyPress(characters: CharacterSet, phases: KeyPress.Phases, action: (KeyPress) -> KeyPress.Result) -> some View
```

Performs an action if the user presses one or more keys on a hardware keyboard while the view has focus.

```
func onKeyPress(keys: Set<KeyEquivalent>, phases: KeyPress.Phases, action: (KeyPress) -> KeyPress.Result) -> some View
```

Performs an action if the user presses one or more keys on a hardware keyboard while the view has focus.

```
struct KeyPress
```

Creating keyboard shortcuts

```
func keyboardShortcut(_:)
```

Assigns a keyboard shortcut to the modified control.

```
func keyboardShortcut(KeyEquivalent, modifiers: EventModifiers) -> some View
```

Defines a keyboard shortcut and assigns it to the modified control.

```
func keyboardShortcut(KeyEquivalent, modifiers: EventModifiers, localization: KeyboardShortcut.Localization) -> some View
```

Defines a keyboard shortcut and assigns it to the modified control.

```
var keyboardShortcut: KeyboardShortcut?
```

The keyboard shortcut that buttons in this environment will be triggered with.

`struct KeyboardShortcut`

Keyboard shortcuts describe combinations of keys on a keyboard that the user can press in order to activate a button or toggle.

`struct KeyEquivalent`

Key equivalents consist of a letter, punctuation, or function key that can be combined with an optional set of modifier keys to specify a keyboard shortcut.

`struct EventModifiers`

A set of key modifiers that you can add to a gesture.

Responding to modifier keys

```
func onModifierKeysChanged(mask: EventModifiers, initial: Bool, (Event
Modifiers, EventModifiers) -> Void) -> some View
```

Performs an action whenever the user presses or releases a hardware modifier key.

```
func modifierKeyAlternate<V>(EventModifiers, () -> V) -> some View
```

Builds a view to use in place of the modified view when the user presses the modifier key(s) indicated by the given set.

Responding to hover events

```
func onHover(perform: (Bool) -> Void) -> some View
```

Adds an action to perform when the user moves the pointer over or away from the view's frame.

```
func onContinuousHover(coordinateSpace:perform:)
```

Adds an action to perform when the pointer enters, moves within, and exits the view's bounds.

```
func hoverEffect(_:isEnabled:)
```

Applies a hover effect to this view.

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

Adds a condition that controls whether this view can display hover effects.

```
func defaultHoverEffect(_:)
```

Sets the default hover effect to use for views within this view.

```
var isHoverEffectEnabled: Bool
```

A Boolean value that indicates whether the view associated with this environment allows hover effects to be displayed.

`enum HoverPhase`

The current hovering state and value of the pointer.

`struct HoverEffectPhaseOverride`

Options for overriding a hover effect's current phase.

`struct OrnamentHoverContentEffect`

Presents an ornament on hover using a custom effect.

`struct OrnamentHoverEffect`

Presents an ornament on hover.

Modifying pointer appearance

`func pointerStyle(PointerStyle?) -> some View`

Sets the pointer style to display when the pointer is over the view.

`struct PointerStyle`

A style describing the appearance of the pointer (also called a cursor) when it's hovered over a view.

`func pointerVisibility(Visibility) -> some View`

Sets the visibility of the pointer when it's over the view.

Changing view appearance for hover events

`func hoverEffect(HoverEffect) -> some View`

Applies a hover effect to this view.

`struct HoverEffect`

An effect applied when the pointer hovers over a view.

`func hoverEffect(some CustomHoverEffect, in: HoverEffectGroup?, isEnabled: Bool) -> some View`

Applies a hover effect to this view, optionally adding it to a `HoverEffectGroup`.

`func hoverEffect(in: HoverEffectGroup?, isEnabled: Bool, body: (Empty HoverEffectContent, Bool, GeometryProxy) -> some HoverEffectContent) -> some View`

Applies a hover effect to this view described by the given closure.

`protocol CustomHoverEffect`

A type that represents how a view should change when a pointer hovers over a view, or when someone looks at the view.

`struct ContentHoverEffect`

A `CustomHoverEffect` that applies effects to a view on hover using a closure.

`struct HoverEffectGroup`

Describes a grouping of effects that activate together.

`func hoverEffectGroup() -> some View`

Adds an implicit `HoverEffectGroup` to all effects defined on descendant views, so that all effects added to subviews activate as a group whenever this view or any descendant views are hovered.

`func hoverEffectGroup(HoverEffectGroup?) -> some View`

Adds a `HoverEffectGroup` to all effects defined on descendant views, and activates the group whenever this view or any descendant views are hovered.

`func hoverEffectGroup(id: String?, in: Namespace.ID, behavior: HoverEffectGroup.Behavior) -> some View`

Adds a `HoverEffectGroup` to all effects defined on descendant views, and activates the group whenever this view or any descendant views are hovered.

`struct GroupHoverEffect`

A `CustomHoverEffect` that activates a named group of effects.

`protocol HoverEffectContent`

A type that describes the effects of a view for a particular hover effect phase.

`struct EmptyHoverEffectContent`

An empty base effect that you use to build other effects.

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

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

`struct HandPointerBehavior`

A behavior that can be applied to the hand pointer while the user is interacting with a view.

Responding to submission events

`func onSubmit(of: SubmitTriggers, () -> Void) -> some View`

Adds an action to perform when the user submits a value to this view.

`func submitScope(Bool) -> some View`

Prevents submission triggers originating from this view to invoke a submission action configured by a submission modifier higher up in the view hierarchy.

`struct SubmitTriggers`

A type that defines various triggers that result in the firing of a submission action.

Labeling a submission event

`func submitLabel(SubmitLabel) -> some View`

Sets the submit label for this view.

`struct SubmitLabel`

A semantic label describing the label of submission within a view hierarchy.

Responding to commands

`func onMoveCommand(perform: ((MoveCommandDirection) -> Void)?) -> some View`

Adds an action to perform in response to a move command, like when the user presses an arrow key on a Mac keyboard, or taps the edge of the Siri Remote when controlling an Apple TV.

`func onDeleteCommand(perform: (() -> Void)?) -> some View`

Adds an action to perform in response to the system's Delete command, or pressing either the ⌫ (backspace) or ⌦ (forward delete) keys while the view has focus.

`func pageCommand<V>(value: Binding<V>, in: ClosedRange<V>, step: V) -> some View`

Steps a value through a range in response to page up or page down commands.

`func onExitCommand(perform: (() -> Void)?) -> some View`

Sets up an action that triggers in response to receiving the exit command while the view has focus.

`func onPlayPauseCommand(perform: (() -> Void)?) -> some View`

Adds an action to perform in response to the system's Play/Pause command.

`func onCommand(Selector, perform: (() -> Void)?) -> some View`

Adds an action to perform in response to the given selector.

`enum MoveCommandDirection`

Specifies the direction of an arrow key movement.

Controlling hit testing

`func allowsTightening(Bool) -> some View`

Sets whether text in this view can compress the space between characters when necessary to fit text in a line.

`func contentShape<S>(S, eoFill: Bool) -> some View`

Defines the content shape for hit testing.

`func contentShape<S>(ContentShapeKinds, S, eoFill: Bool) -> some View`

Sets the content shape for this view.

`struct ContentShapeKinds`

A kind for the content shape of a view.

Interacting with the Digital Crown

`func digitalCrownAccessory(Visibility) -> some View`

Specifies the visibility of Digital Crown accessory Views on Apple Watch.

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

Places an accessory View next to the Digital Crown on Apple Watch.

`func digitalCrownRotation<V>(Binding<V>, from: V, through: V, sensitivity: DigitalCrownRotationalSensitivity, isContinuous: Bool, isHapticFeedbackEnabled: Bool, onChange: (DigitalCrownEvent) -> Void, onIdle: () -> Void) -> some View`

Tracks Digital Crown rotations by updating the specified binding.

`func digitalCrownRotation<V>(Binding<V>, onChange: (DigitalCrownEvent) -> Void, onIdle: () -> Void) -> some View`

Tracks Digital Crown rotations by updating the specified binding.

`func digitalCrownRotation(detent:from:through:by:sensitivity:isContinuous:isHapticFeedbackEnabled:onChange:onIdle:)`

Tracks Digital Crown rotations by updating the specified binding.

`func digitalCrownRotation<V>(Binding<V>) -> some View`

Tracks Digital Crown rotations by updating the specified binding.

`func digitalCrownRotation<V>(Binding<V>, from: V, through: V, by: V.Stride?, sensitivity: DigitalCrownRotationalSensitivity, isContinuous: Bool, isHapticFeedbackEnabled: Bool) -> some View`

Tracks Digital Crown rotations by updating the specified binding.

`struct DigitalCrownEvent`

An event emitted when the user rotates the Digital Crown.

`enum DigitalCrownRotationalSensitivity`

The amount of Digital Crown rotation needed to move between two integer numbers.

Managing Touch Bar input

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

Sets the content that the Touch Bar displays.

`func touchBar<Content>(TouchBar<Content>) -> some View`

Sets the Touch Bar content to be shown in the Touch Bar when applicable.

`func touchBarItemPrincipal(Bool) -> some View`

Sets principal views that have special significance to this Touch Bar.

`func touchBarCustomizationLabel(Text) -> some View`

Sets a user-visible string that identifies the view's functionality.

`func touchBarItemPresence(TouchBarItemPresence) -> some View`

Sets the behavior of the user-customized view.

`struct TouchBar`

A container for a view that you can show in the Touch Bar.

`enum TouchBarItemPresence`

Options that affect user customization of the Touch Bar.

Responding to capture events

```
func onCameraCaptureEvent(isEnabled: Bool, action: (AVCaptureEvent) -> Void) -> some View
```

Used to register an action triggered by system capture events.

```
func onCameraCaptureEvent(isEnabled: Bool, primaryAction: (AVCaptureEvent) -> Void, secondaryAction: (AVCaptureEvent) -> Void) -> some View
```

Used to register actions triggered by system capture events.

See Also

Event handling

- Gestures

  Define interactions from taps, clicks, and swipes to fine-grained gestures.

- Clipboard

  Enable people to move or duplicate items by issuing Copy and Paste commands.

- Drag and drop

  Enable people to move or duplicate items by dragging them from one location to another.

- Focus

  Identify and control which visible object responds to user interaction.

- System events

  React to system events, like opening a URL.