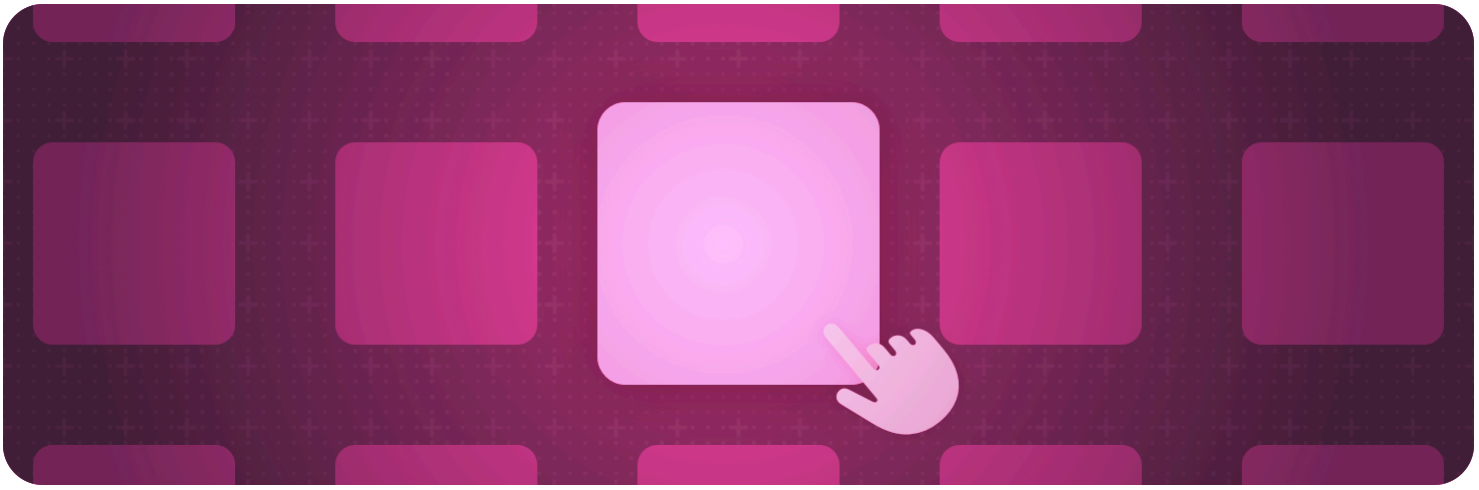
API Collection

# Focus

Identify and control which visible object responds to user interaction.

## Overview

Focus indicates which element in the display receives the next input. Use view modifiers to indicate which views can receive focus, to detect which view has focus, and to programmatically control focus state.



For design guidance, see Focus and selection in the Human Interface Guidelines.

## Topics

### Essentials

{} Focus Cookbook: Supporting and enhancing focus-driven interactions in your SwiftUI app

Create custom focusable views with key-press handlers that accelerate keyboard input and support movement, and control focus programmatically.

## Indicating that a view can receive focus

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

Specifies if the view is focusable.

```
func focusable(Bool, interactions: FocusInteractions) -> some View
```

Specifies if the view is focusable, and if so, what focus-driven interactions it supports.

```
struct FocusInteractions
```

Values describe different focus interactions that a view can support.

## Managing focus state

```
func focused<Value>(FocusState<Value>.Binding, equals: Value) -> some View
```

Modifies this view by binding its focus state to the given state value.

```
func focused(FocusState<Bool>.Binding) -> some View
```

Modifies this view by binding its focus state to the given Boolean state value.

```
var isFocused: Bool
```

Returns whether the nearest focusable ancestor has focus.

```
struct FocusState
```

A property wrapper type that can read and write a value that SwiftUI updates as the placement of focus within the scene changes.

```
struct FocusedValue
```

A property wrapper for observing values from the focused view or one of its ancestors.

```
macro Entry()
```

Creates an environment values, transaction, container values, or focused values entry.

```
protocol FocusedValueKey
```

A protocol for identifier types used when publishing and observing focused values.

```
struct FocusedBinding
```

A convenience property wrapper for observing and automatically unwrapping state bindings from the focused view or one of its ancestors.

`func searchFocused(FocusState<Bool>.Binding) -> some View`

Modifies this view by binding the focus state of the search field associated with the nearest searchable modifier to the given Boolean value.

`func searchFocused<V>(FocusState<V>.Binding, equals: V) -> some View`

Modifies this view by binding the focus state of the search field associated with the nearest searchable modifier to the given value.

## Exposing value types to focused views

`func focusedValue<T>(T?) -> some View`

Sets the focused value for the given object type.

`func focusedValue(_:_:)`

Modifies this view by injecting a value that you provide for use by other views whose state depends on the focused view hierarchy.

`func focusedSceneValue<T>(T?) -> some View`

Sets the focused value for the given object type at a scene-wide scope.

`func focusedSceneValue(_:_:)`

Modifies this view by injecting a value that you provide for use by other views whose state depends on the focused scene.

`struct FocusedValues`

A collection of state exported by the focused scene or view and its ancestors.

## Exposing reference types to focused views

`func focusedObject(_:)`

Creates a new view that exposes the provided object to other views whose whose state depends on the focused view hierarchy.

`func focusedSceneObject(_:)`

Creates a new view that exposes the provided object to other views whose whose state depends on the active scene.

`struct FocusedObject`

A property wrapper type for an observable object supplied by the focused view or one of its ancestors.

## Setting focus scope

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

Creates a focus scope that SwiftUI uses to limit default focus preferences.

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

Indicates that the view's frame and cohort of focusable descendants should be used to guide focus movement.

## Controlling default focus

```
func prefersDefaultFocus(Bool, in: Namespace.ID) -> some View
```

Indicates that the view should receive focus by default for a given namespace.

```
func defaultFocus<V>(FocusState<V>.Binding, V, priority: DefaultFocus
EvaluationPriority) -> some View
```

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

```
struct DefaultFocusEvaluationPriority
```

Prioritizations for default focus preferences when evaluating where to move focus in different circumstances.

## Resetting focus

```
var resetFocus: ResetFocusAction
```

An action that requests the focus system to reevaluate default focus.

```
struct ResetFocusAction
```

An environment value that provides the ability to reevaluate default focus.

## Configuring effects

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

Adds a condition that controls whether this view can display focus effects, such as a default focus ring or hover effect.

`var isFocusEffectEnabled: Bool`

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

---

# See Also

## Event handling

- Gestures

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

- Input events

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

- 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.

- System events

  React to system events, like opening a URL.