Instance Method

onChange(of:perform:) Deprecated

Adds an action to perform when the given value changes.

iOS 14.0–17.0 Deprecated | iPadOS 14.0–17.0 Deprecated | Mac Catalyst 14.0–17.0 Deprecated | macOS 11.0–14.0 Deprecated | tvOS 14.0–17.0 Deprecated | visionOS 1.0–1.0 Deprecated | watchOS 7.0–10.0 Deprecated

```
nonisolated
func onChange<V>(
    of value: V,
    perform action: @escaping (V) -> Void
) -> some View where V : Equatable
```

Deprecated

Use `onChange(of:initial:_:)` or `onChange(of:initial:_:)` instead. The trailing closure in each case takes either zero or two input parameters, compared to this method which takes one.

Be aware that the replacements have slightly different behavior. This modifier's closure captures values that represent the state before the change. The new modifiers capture values that correspond to the new state. The new behavior makes it easier to perform updates that rely on values other than the one that caused the modifier's closure to run.

Discussion

Use this modifier to trigger a side effect when a value changes, like the value associated with an `Environment` value or a `Binding`. For example, you can clear a cache when you notice that a scene moves to the background:

```
struct MyScene: Scene {
    @Environment(\.scenePhase) private var scenePhase
    @StateObject private var cache = DataCache()

    var body: some Scene {
        WindowGroup {
            MyRootView()
        }
        .onChange(of: scenePhase) { newScenePhase in
            if newScenePhase == .background {
                cache.empty()
            }
        }
    }
}
```

The system may call the action closure on the main actor, so avoid long-running tasks in the closure. If you need to perform such tasks, detach an asynchronous background task:

```
.onChange(of: scenePhase) { newScenePhase in
    if newScenePhase == .background {
        Task.detached(priority: .background) {
            // ...
        }
    }
}
```

The system passes the new value into the closure. If you need the old value, capture it in the closure.

See Also

Input and events modifiers

~~func onTapGesture(count: Int, coordinateSpace: CoordinateSpace, perform: (CGPoint) -> Void) -> some View~~

Adds an action to perform when this view recognizes a tap gesture, and provides the action with the location of the interaction.

Deprecated

~~func onLongPressGesture(minimumDuration: Double, maximumDistance: CGFloat, pressing: ((Bool) -> Void)?, perform: () -> Void) -> some View~~

Adds an action to perform when this view recognizes a long press gesture.

Deprecated

~~func onLongPressGesture(minimumDuration: Double, pressing: ((Bool) -> Void)?, perform: () -> Void) -> some View~~

Adds an action to perform when this view recognizes a long press gesture.

Deprecated

~~func onPasteCommand(of: [String], perform: ([NSItemProvider]) -> Void) -> some View~~

Adds an action to perform in response to the system's Paste command.

Deprecated

~~func onPasteCommand<Payload>(of: [String], validator: ([NSItemProvider]) -> Payload?, perform: (Payload) -> Void) -> some View~~

Adds an action to perform in response to the system's Paste command with items that you validate.

Deprecated

~~func onDrop(of: [String], delegate: any DropDelegate) -> some View~~

Defines the destination for a drag and drop operation with the same size and position as this view, with behavior controlled by the given delegate.

Deprecated

func onDrop(of:isTargeted:perform:)

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

~~func focusable(Bool, onFocusChange: (Bool) -> Void) -> some View~~

Specifies if the view is focusable and, if so, adds an action to perform when the view comes into focus.

Deprecated

~~func onContinuousHover(coordinateSpace: CoordinateSpace, perform: (HoverPhase) -> Void) -> some View~~

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

Deprecated