Instance Method

tag(_:includeOptional:)

Sets the unique tag value of this view.

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

```
nonisolated
func tag<V>(
    _ tag: V,
    includeOptional: Bool = true
) -> some View where V : Hashable
```

Parameters

tag
A Hashable value to use as the view's tag.

includeOptional
If the tag value for Optional<V> should also be set.

Return Value

A view with the specified tag set.

Discussion

Use this modifier to differentiate among certain selectable views, like the possible values of a Picker or the tabs of a TabView. Tag values can be of any type that conforms to the Hashable protocol.

This modifier will write the tag value for the type `V`, as well as `Optional<V>` if `include Optional` is enabled. Containers checking for tags of either type will see the value as set.

In the example below, the `ForEach` loop in the `Picker` view builder iterates over the `Flavor` enumeration. It extracts the string value of each enumeration element for use in constructing the row label, and uses the enumeration value as input to the `tag(_:)` modifier.

```
struct FlavorPicker: View {
    enum Flavor: String, CaseIterable, Identifiable {
        case chocolate, vanilla, strawberry
        var id: Self { self }
    }

    @State private var selectedFlavor: Flavor? = nil

    var body: some View {
        Picker("Flavor", selection: $selectedFlavor) {
            ForEach(Flavor.allCases) { flavor in
                Text(flavor.rawValue)
                    .tag(flavor)
            }
        }
    }
}
```

The selection type of the `Picker` is an `Optional<Flavor>` and so it will look for tags on its contents of `Optional<Flavor>` type. Since the tag modifier defaults to having `include Optional` enabled, even though the tag for each option is a non-optional `Flavor`, the tag modifier writes values for both the non-optional, and optional versions of the value, allowing the contents to be selectable by the `Picker`.

A `ForEach` automatically applies a default tag to each enumerated view using the `id` parameter of the corresponding element. If the element's `id` parameter and the picker's `selection` input have exactly the same type, or the same type but optional, you can omit the explicit tag modifier.

To see examples that don't require an explicit tag, see `Picker`.

See Also

Managing the view hierarchy

`func id<ID>(ID) -> some View`

Binds a view's identity to the given proxy value.

```
func equatable() -> EquatableView<Self>
```

Prevents the view from updating its child view when its new value is the same as its old value.