Structure

ProgressView

A view that shows the progress toward completion of a task.

iOS 14.0+ | iPadOS 14.0+ | Mac Catalyst 14.0+ | macOS 11.0+ | tvOS 14.0+ | visionOS 1.0+ | watchOS 7.0+

```
struct ProgressView<Label, CurrentValueLabel> where Label : View, CurrentValue
Label : View
```

Mentioned in

Declaring a custom view

Overview

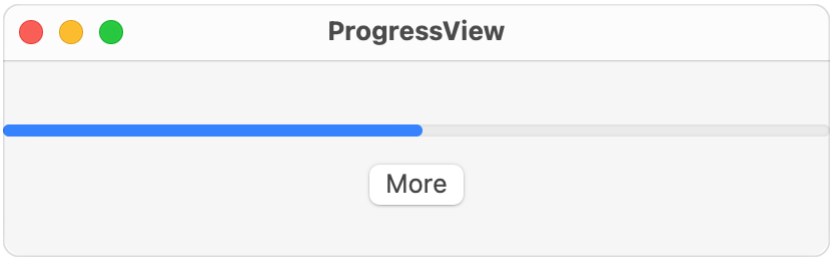
Use a progress view to show that a task is incomplete but advancing toward completion. A progress view can show both determinate (percentage complete) and indeterminate (progressing or not) types of progress.

Create a determinate progress view by initializing a `ProgressView` with a binding to a numeric value that indicates the progress, and a `total` value that represents completion of the task. By default, the progress is `0.0` and the total is `1.0`.

The example below uses the state property `progress` to show progress in a determinate `ProgressView`. The progress view uses its default total of `1.0`, and because `progress` starts with an initial value of `0.5`, the progress view begins half-complete. A “More” button below the progress view allows people to increment the progress in increments of five percent:

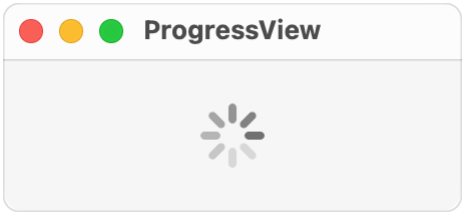
```
struct LinearProgressDemoView: View {
    @State private var progress = 0.5
```

```
    var body: some View {
        VStack {
            ProgressView(value: progress)
            Button("More") { progress += 0.05 }
        }
    }
}
```



To create an indeterminate progress view, use an initializer that doesn't take a progress value:

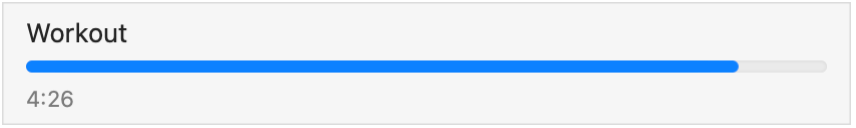
```
var body: some View {
    ProgressView()
}
```



You can also create a progress view that covers a closed range of `Date` values. As long as the current date is within the range, the progress view automatically updates, filling or depleting the progress view as it nears the end of the range. The following example shows a five-minute timer whose start time is that of the progress view's initialization:

```
struct DateRelativeProgressDemoView: View {
    let workoutDateRange = Date()...Date().addingTimeInterval(5*60)

    var body: some View {
         ProgressView(timerInterval: workoutDateRange) {
             Text("Workout")
         }
    }
}
```

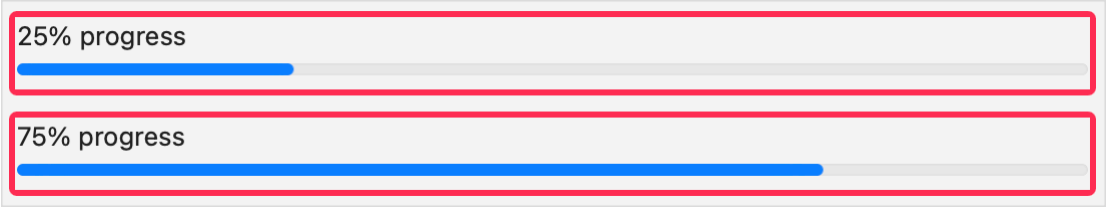


Styling progress views

You can customize the appearance and interaction of progress views by creating styles that conform to the `ProgressViewStyle` protocol. To set a specific style for all progress view instances within a view, use the `progressViewStyle(_:)` modifier. In the following example, a custom style adds a rounded pink border to all progress views within the enclosing `VStack`:

```
struct BorderedProgressViews: View {
    var body: some View {
        VStack {
            ProgressView(value: 0.25) { Text("25% progress") }
            ProgressView(value: 0.75) { Text("75% progress") }
        }
        .progressViewStyle(PinkBorderedProgressViewStyle())
    }
}

struct PinkBorderedProgressViewStyle: ProgressViewStyle {
    func makeBody(configuration: Configuration) -> some View {
        ProgressView(configuration)
            .padding(4)
            .border(.pink, width: 3)
            .cornerRadius(4)
    }
}
```



SwiftUI provides two built-in progress view styles, `linear` and `circular`, as well as an automatic style that defaults to the most appropriate style in the current context. The following example shows a circular progress view that starts at 60 percent completed.

```
struct CircularProgressDemoView: View {
    @State private var progress = 0.6
```

```
    var body: some View {
        VStack {
            ProgressView(value: progress)
                .progressViewStyle(.circular)
        }
    }
}
```

On platforms other than macOS, the circular style may appear as an indeterminate indicator instead.

Topics

Creating an indeterminate progress view

`init()`

Creates a progress view for showing indeterminate progress, without a label.

`init(label: () -> Label)`

Creates a progress view for showing indeterminate progress that displays a custom label.

`init(LocalizedStringKey)`

Creates a progress view for showing indeterminate progress that generates its label from a localized string.

`init<S>(S)`

Creates a progress view for showing indeterminate progress that generates its label from a string.

Creating a determinate progress view

`init(Progress)`

Creates a progress view for visualizing the given progress instance.

`init<V>(value: V?, total: V)`

Creates a progress view for showing determinate progress.

`init(_:value:total:)`

Creates a progress view for showing determinate progress that generates its label from a string.

`init<V>(value: V?, total: V, label: () -> Label)`

Creates a progress view for showing determinate progress, with a custom label.

`init<V>(value: V?, total: V, label: () -> Label, currentValueLabel: () -> CurrentValueLabel)`

Creates a progress view for showing determinate progress, with a custom label.

Create a progress view spanning a date range

`init(timerInterval: ClosedRange<Date>, countsDown: Bool)`

Creates a progress view for showing continuous progress as time passes.

`init(timerInterval: ClosedRange<Date>, countsDown: Bool, label: () -> Label)`

Creates a progress view for showing continuous progress as time passes, with a descriptive label.

`init(timerInterval: ClosedRange<Date>, countsDown: Bool, label: () -> Label, currentValueLabel: () -> CurrentValueLabel)`

Creates a progress view for showing continuous progress as time passes, with descriptive and current progress labels.

Initializers

`init(_:)`

Creates a progress view based on a style configuration.

Relationships

Conforms To

`View`

See Also

Indicating a value

`struct Gauge`

A view that shows a value within a range.

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

Sets the style for gauges within this view.

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

Sets the style for progress views in this view.

`struct DefaultDateProgressLabel`

The default type of the current value label when used by a date-relative progress view.

`struct DefaultButtonLabel`

The default label to use for a button.