

[SwiftUI](#) / Stepper

Structure

Stepper

A control that performs increment and decrement actions.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | visionOS 1.0+ | watchOS 9.0+

```
struct Stepper<Label> where Label : View
```

Overview

Use a stepper control when you want the user to have granular control while incrementing or decrementing a value. For example, you can use a stepper to:

- Change a value up or down by 1.
- Operate strictly over a prescribed range.
- Step by specific amounts over a stepper's range of possible values.

The example below uses an array that holds a number of [Color](#) values, a local state variable, `value`, to set the control's background color, and title label. When the user clicks or taps the stepper's increment or decrement buttons, SwiftUI executes the relevant closure that updates `value`, wrapping the value to prevent overflow. SwiftUI then re-renders the view, updating the text and background color to match the current index:

```
struct StepperView: View {  
    @State private var value = 0  
    let colors: [Color] = [.orange, .red, .gray, .blue,  
                          .green, .purple, .pink]  
  
    func incrementStep() {  
        value += 1  
    }  
}
```

```

        if value >= colors.count { value = 0 }

}

func decrementStep() {
    value -= 1
    if value < 0 { value = colors.count - 1 }
}

var body: some View {
    Stepper {
        Text("Value: \(value) Color: \(colors[value].description)")
    } onIncrement: {
        incrementStep()
    } onDecrement: {
        decrementStep()
    }
    .padding(5)
    .background(colors[value])
}

```



The following example shows a stepper that displays the effect of incrementing or decrementing a value with the step size of `step` with the bounds defined by `range`:

```

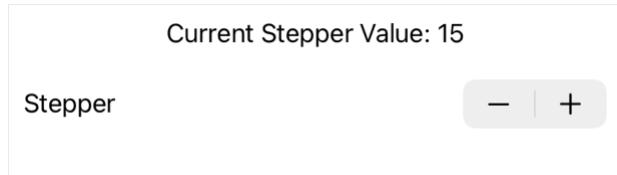
struct StepperView: View {
    @State private var value = 0
    let step = 5
    let range = 1...50

    var body: some View {
        Stepper(
            value: $value,
            in: range,
            step: step
        ) {
            Text("Current: \(value) in \(range.description) " +
                "stepping by \(step)")
        }
        .padding(10)
    }
}

```

```
}
```

```
}
```



Topics

Creating a stepper

```
init<V>(value: Binding<V>, step: V.Stride, label: () -> Label, onEditingChanged: (Bool) -> Void)
```

Creates a stepper configured to increment or decrement a binding to a value using a step value you provide.

```
init<F>(value: Binding<F.FormatInput>, step: F.FormatInput.Stride, format: F, label: () -> Label, onEditingChanged: (Bool) -> Void)
```

Creates a stepper configured to increment or decrement a binding to a value using a step value you provide, displaying its value with an applied format style.

```
init(_:value:step:onEditingChanged:)
```

Creates a stepper with a title and configured to increment and decrement a binding to a value and step amount you provide.

```
init(_:value:step:format:onEditingChanged:)
```

Creates a stepper with a title key and configured to increment and decrement a binding to a value and step amount you provide, displaying its value with an applied format style.

Creating a stepper over a range

```
init<V>(value: Binding<V>, in: ClosedRange<V>, step: V.Stride, label: () -> Label, onEditingChanged: (Bool) -> Void)
```

Creates a stepper configured to increment or decrement a binding to a value using a step value and within a range of values you provide.

```
init<F>(value: Binding<F.FormatInput>, in: ClosedRange<F.FormatInput>, step: F.FormatInput.Stride, format: F, label: () -> Label, onEditingChanged: (Bool) -> Void)
```

Creates a stepper configured to increment or decrement a binding to a value using a step value and within a range of values you provide, displaying its value with an applied format style.

```
init(_:value:in:step:onEditingChanged:)
```

Creates a stepper instance that increments and decrements a binding to a value, by a step size and within a closed range that you provide.

```
init(_:value:in:step:format:onEditingChanged:)
```

Creates a stepper instance that increments and decrements a binding to a value, by a step size and within a closed range that you provide, displaying its value with an applied format style.

Creating a stepper with change behavior

```
init(label: () -> Label, onIncrement: () -> Void)?, onDecrement: () -> Void)?, onEditingChanged: (Bool) -> Void)
```

Creates a stepper instance that performs the closures you provide when the user increments or decrements the stepper.

```
init(_:onIncrement:onDecrement:onEditingChanged:)
```

Creates a stepper that uses a title key and executes the closures you provide when the user clicks or taps the stepper's increment and decrement buttons.

Deprecated initializers

```
init<V>(value: Binding<V>, step: V.Stride, onEditingChanged: (Bool) -> Void, label: () -> Label)
```

Creates a stepper configured to increment or decrement a binding to a value using a step value you provide.

Deprecated

```
init<V>(value: Binding<V>, in: ClosedRange<V>, step: V.Stride, onEditingChanged: (Bool) -> Void, label: () -> Label)
```

Creates a stepper configured to increment or decrement a binding to a value using a step value and within a range of values you provide.

Deprecated

```
init(onIncrement: () -> Void)?, onDecrement: () -> Void)?, onEditingChanged: (Bool) -> Void, label: () -> Label)
```

Creates a stepper instance that performs the closures you provide when the user increments or decrements the stepper.

Deprecated

Relationships

Conforms To

View

See Also

Getting numeric inputs

`struct Slider`

A control for selecting a value from a bounded linear range of values.

`struct Toggle`

A control that toggles between on and off states.

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

Sets the style for toggles in a view hierarchy.