Structure

Spacer

A flexible space that expands along the major axis of its containing stack layout, or on both axes if not contained in a stack.

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

```
@frozen
struct Spacer
```

Mentioned in

- Building layouts with stack views
- Adding a background to your view
- Picking container views for your content

Overview

A spacer creates an adaptive view with no content that expands as much as it can. For example, when placed within an `HStack`, a spacer expands horizontally as much as the stack allows, moving sibling views out of the way, within the limits of the stack's size. SwiftUI sizes a stack that doesn't contain a spacer up to the combined ideal widths of the content of the stack's child views.

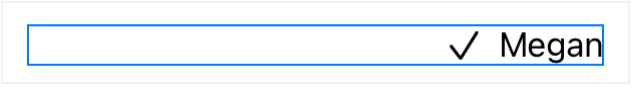
The following example provides a simple checklist row to illustrate how you can use a spacer:

```
struct ChecklistRow: View {
    let name: String

    var body: some View {
        HStack {
```

```
            Image(systemName: "checkmark")
            Text(name)
        }
        .border(Color.blue)
    }
}
```

Adding a spacer before the image creates an adaptive view with no content that expands to push the image and text to the right side of the stack. The stack also now expands to take as much space as the parent view allows, shown by the blue border that indicates the boundary of the stack:

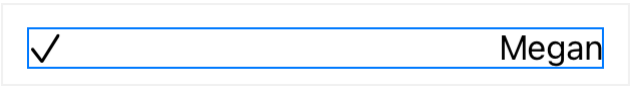
```
struct ChecklistRow: View {
    let name: String

    var body: some View {
        HStack {
            Spacer()
            Image(systemName: "checkmark")
            Text(name)
        }
        .border(Color.blue)
    }
}
```



Moving the spacer between the image and the name pushes those elements to the left and right sides of the `HStack`, respectively. Because the stack contains the spacer, it expands to take as much horizontal space as the parent view allows; the blue border indicates its size:

```
struct ChecklistRow: View {
    let name: String

    var body: some View {
        HStack {
            Image(systemName: "checkmark")
            Spacer()
```

```
            Text(name)
        }
        .border(Color.blue)
    }
}
```



Adding two spacer views on the outside of the stack leaves the image and text together, while the stack expands to take as much horizontal space as the parent view allows:

```
struct ChecklistRow: View {
    let name: String

    var body: some View {
        HStack {
            Spacer()
            Image(systemName: "checkmark")
            Text(name)
            Spacer()
        }
        .border(Color.blue)
    }
}
```

Topics

Creating a spacer

`init(minLength: CGFloat?)`

`var minLength: CGFloat?`

The minimum length this spacer can be shrunk to, along the axis or axes of expansion.

Relationships

Conforms To

`BitwiseCopyable`
`Copyable`
`Sendable`
`SendableMetatype`
`View`

See Also

Separators

`struct Divider`

A visual element that can be used to separate other content.