Structure

ZStack

A view that overlays its subviews, aligning them in both axes.

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

```
@frozen
struct ZStack<Content> where Content : View
```

Mentioned in

- Building layouts with stack views
- Laying out a simple view
- Adding a background to your view
- Aligning views within a stack
- Creating performant scrollable stacks

Overview

The `ZStack` assigns each successive subview a higher z-axis value than the one before it, meaning later subviews appear "on top" of earlier ones.

The following example creates a `ZStack` of 100 x 100 point `Rectangle` views filled with one of six colors, offsetting each successive subview by 10 points so they don't completely overlap:

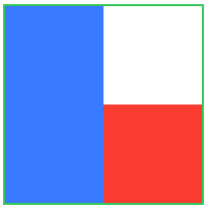
```
let colors: [Color] =
    [.red, .orange, .yellow, .green, .blue, .purple]

var body: some View {
```

```
    ZStack {
        ForEach(0..<colors.count) {
            Rectangle()
                .fill(colors[$0])
                .frame(width: 100, height: 100)
                .offset(x: CGFloat($0) * 10.0,
                        y: CGFloat($0) * 10.0)
        }
    }
}
```

The `ZStack` uses an `Alignment` to set the x- and y-axis coordinates of each subview, defaulting to a `center` alignment. In the following example, the `ZStack` uses a `bottomLeading` alignment to lay out two subviews, a red 100 x 50 point rectangle below, and a blue 50 x 100 point rectangle on top. Because of the alignment value, both rectangles share a bottom-left corner with the `ZStack` (in locales where left is the leading side).

```
var body: some View {
    ZStack(alignment: .bottomLeading) {
        Rectangle()
            .fill(Color.red)
            .frame(width: 100, height: 50)
        Rectangle()
            .fill(Color.blue)
            .frame(width:50, height: 100)
    }
    .border(Color.green, width: 1)
}
```



> **Note**
>
> If you need a version of this stack that conforms to the `Layout` protocol, like when you want to create a conditional layout using `AnyLayout`, use `ZStackLayout` instead.

Topics

Creating a stack

`init(alignment: Alignment, content: () -> Content)`

Creates an instance with the given alignment.

Supporting symbols

`struct ZStackContent3D`

A type that adds spacing to a `ZStack`.

Initializers

`init<V>(alignment: Alignment, spacing: CGFloat?, content: () -> V)`

Creates an instance with the given spacing and alignment.

Relationships

Conforms To

`View`

See Also

Layering views

Adding a background to your view

Compose a background behind your view and extend it beyond the safe area insets.

```
func zIndex(Double) -> some View
```

Controls the display order of overlapping views.

```
func background<V>(alignment: Alignment, content: () -> V) -> some View
```

Layers the views that you specify behind this view.

```
func background<S>(S, ignoresSafeAreaEdges: Edge.Set) -> some View
```

Sets the view's background to a style.

```
func background(ignoresSafeAreaEdges: Edge.Set) -> some View
```

Sets the view's background to the default background style.

```
func background(_:in:fillStyle:)
```

Sets the view's background to an insettable shape filled with a style.

```
func background(in:fillStyle:)
```

Sets the view's background to an insettable shape filled with the default background style.

```
func overlay<V>(alignment: Alignment, content: () -> V) -> some View
```

Layers the views that you specify in front of this view.

```
func overlay<S>(S, ignoresSafeAreaEdges: Edge.Set) -> some View
```

Layers the specified style in front of this view.

```
func overlay<S, T>(S, in: T, fillStyle: FillStyle) -> some View
```

Layers a shape that you specify in front of this view.

```
var backgroundMaterial: Material?
```

The material underneath the current view.

```
func containerBackground<S>(S, for: ContainerBackgroundPlacement) ->
some View
```

Sets the container background of the enclosing container using a view.

```
func containerBackground<V>(for: ContainerBackgroundPlacement,
alignment: Alignment, content: () -> V) -> some View
```

Sets the container background of the enclosing container using a view.

```
struct ContainerBackgroundPlacement
```

The placement of a container background.