

[SwiftUI](#) / Image

Structure






Image

A view that displays an image.

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

```
@frozen
struct Image
```

Mentioned in

-  Building layouts with stack views
-  Configuring views
-  Creating performant scrollable stacks
-  Displaying data in lists
-  Fitting images into available space

Overview

Use an `Image` instance when you want to add images to your SwiftUI app. You can create images from many sources:

- Image files in your app's asset library or bundle. Supported types include PNG, JPEG, HEIC, and more.
- Instances of platform-specific image types, like [UIImage](#) and [NSImage](#).
- A bitmap stored in a Core Graphics [CGImage](#) instance.
- System graphics from the SF Symbols set.

The following example shows how to load an image from the app's asset library or bundle and scale it to fit within its container:

```
Image("Landscape_4")  
    .resizable()  
    .aspectRatio(contentMode: .fit)  
Text("Water wheel")
```



Water wheel

You can use methods on the `Image` type as well as standard view modifiers to adjust the size of the image to fit your app's interface. Here, the `Image` type's `resizable(capInsets: resizingMode:)` method scales the image to fit the current view. Then, the `aspectRatio(_: contentMode:)` view modifier adjusts this resizing behavior to maintain the image's original aspect ratio, rather than scaling the x- and y-axes independently to fill all four sides of the view. The article [Fitting images into available space](#) shows how to apply scaling, clipping, and tiling to `Image` instances of different sizes.

An `Image` is a late-binding token; the system resolves its actual value only when it's about to use the image in an environment.

Making images accessible

To use an image as a control, use one of the initializers that takes a `label` parameter. This allows the system's accessibility frameworks to use the label as the name of the control for users who use features like VoiceOver. For images that are only present for aesthetic reasons, use an initializer with the `decorative` parameter; the accessibility systems ignore these images.

Topics

Creating an image

`init(String, bundle: Bundle?)`

Creates a labeled image that you can use as content for controls.

`init(String, variableValue: Double?, bundle: Bundle?)`

Creates a labeled image that you can use as content for controls, with a variable value.

`init(ImageResource)`

Initialize an Image with an image resource.

Creating an image for use as a control

`init(String, bundle: Bundle?, label: Text)`

Creates a labeled image that you can use as content for controls, with the specified label.

`init(String, variableValue: Double?, bundle: Bundle?, label: Text)`

Creates a labeled image that you can use as content for controls, with the specified label and variable value.

`init(CGImage, scale: CGFloat, orientation: Image.Orientation, label: Text)`

Creates a labeled image based on a Core Graphics image instance, usable as content for controls.

Creating an image for decorative use

`init(decorative: String, bundle: Bundle?)`

Creates an unlabeled, decorative image.

`init(decorative: String, variableValue: Double?, bundle: Bundle?)`

Creates an unlabeled, decorative image, with a variable value.

`init(decorative: CGImage, scale: CGFloat, orientation: Image.Orientation)`

Creates an unlabeled, decorative image based on a Core Graphics image instance.

Creating a system symbol image

```
init(systemName: String)
```

Creates a system symbol image.

```
init(systemName: String, variableValue: Double?)
```

Creates a system symbol image with a variable value.

Creating an image from another image

```
init(uiImage: UIImage)
```

Creates a SwiftUI image from a UIKit image instance.

```
init(nsImage: NSImage)
```

Creates a SwiftUI image from an AppKit image instance.

Creating an image from drawing instructions

```
init(size: CGSize, label: Text?, opaque: Bool, colorMode: Color  
RenderingMode, renderer: (inout GraphicsContext) -> Void)
```

Initializes an image of the given size, with contents provided by a custom rendering closure.

Resizing images

```
func resizable(capInsets: EdgeInsets, resizeMode: Image.ResizingMode)  
-> Image
```

Sets the mode by which SwiftUI resizes an image to fit its space.

Specifying rendering behavior

```
func antialiased(Bool) -> Image
```

Specifies whether SwiftUI applies antialiasing when rendering the image.

```
func symbolRenderingMode(SymbolRenderingMode?) -> Image
```

Sets the rendering mode for symbol images within this view.

```
func renderingMode(Image.TemplateRenderingMode?) -> Image
```

Indicates whether SwiftUI renders an image as-is, or by using a different mode.

```
func interpolation(Image.Interpolation) -> Image
```

Specifies the current level of quality for rendering an image that requires interpolation.

```
enum TemplateRenderingMode
```

A type that indicates how SwiftUI renders images.

```
enum Interpolation
```

The level of quality for rendering an image that requires interpolation, such as a scaled image.

Specifying dynamic range

```
func allowedDynamicRange(Image.DynamicRange?) -> Image
```

Returns a new image configured with the specified allowed dynamic range.

```
var allowedDynamicRange: Image.DynamicRange?
```

The allowed dynamic range for the view, or nil.

```
struct DynamicRange
```

Instance Methods

```
func symbolColorRenderingMode(SymbolColorRenderingMode?) -> Image
```

Sets the color rendering mode of the image.

```
func symbolVariableValueMode(SymbolVariableValueMode?) -> Image
```

Sets the variable value mode mode for symbol images within this view.

```
func widgetAccentedRenderingMode(WidgetAccentedRenderingMode?) -> some View
```

Specifies the how to render an Image when using the WidgetKit/WidgetRendering Mode/accented mode.

Enumerations

```
enum Orientation
```

The orientation of an image.

```
enum ResizingMode
```

The modes that SwiftUI uses to resize an image to fit within its containing view.

```
enum Scale
```

A scale to apply to vector images relative to text.

Relationships

Conforms To

- Copyable
- Equatable
- JournalingSuggestionAsset
- Sendable
- SendableMetatype
- Transferable
- View