

Documentation

[Xcode](#) / Creating your app's interface with SwiftUI

Article

Creating your app's interface with SwiftUI

Develop apps in SwiftUI with an interactive preview that keeps the code and layout in sync.

Overview

If you choose the SwiftUI framework to develop your app, you can see an interactive preview in the canvas as you lay out your user interface. Xcode keeps the changes you make to the source code, the user interface layout, and the canvas in sync. For example, when you edit views in the source editor, Xcode updates the corresponding views in the canvas.

If you create your project from the Multiplatform template or choose SwiftUI as the interface when you select another template, Xcode adds default preview macros to the SwiftUI code for you. Otherwise, you can generate a preview using coding intelligence or add preview macros yourself.

For more information about Xcode templates, see [Creating an Xcode project for an app](#). For more details about adding previews, see [Previewing your app's interface in Xcode](#).

Display the SwiftUI preview

To show the preview, select a file that contains a preview macro in the Project navigator and, if necessary, choose Editor > Canvas to show the canvas. Then click the Resume button in the upper-right corner of the canvas to start the preview. Xcode builds and runs the code, displaying the results directly in the canvas.

The screenshot shows the Xcode interface with the project 'HelloWorld' open. The left sidebar shows the file structure with 'ContentView' selected. The main editor shows the following Swift code:

```
1 //  
2 // ContentView.swift  
3 // HelloWorld  
4 //  
5 // Created by Ravi Patel on 9/4/25.  
6 //  
7  
8 import SwiftUI  
9  
10 struct ContentView: View {  
11     var body: some View {  
12         VStack {  
13             Image(systemName: "globe")  
14                 .imageScale(.large)  
15                 .foregroundStyle(.tint)  
16             Text("Hello, world!")  
17         }  
18     }  
19 }  
20  
21  
22 #Preview {  
23     ContentView()  
24 }
```

The preview area on the right shows an iPhone 16 device with the text "Hello, world!" displayed on its screen.

Use the controls at the bottom of the canvas to switch between different preview modes, add variants for appearances, select device settings such as orientation, and change the device.

Note

If you add multiple preview and playground macros to a file, you can switch between them using the tabs that appear at the top of the canvas. To add playgrounds to your Swift code, see [Running code snippets using the playground macro](#).

Add views and modifiers using the library

You can add views and modifiers to your code from the library, and Xcode keeps the preview in sync with your changes.

To open the library, choose **View > Show Library** (press the Option key to open it as a window). Then click the Views or Modifiers button in the toolbar and drag user interface elements from the library to your source code. To find elements more quickly, begin entering the element's name in the search field at the top of the Library.



Controls



Button



Color Picker



Date Picker



Disclosure Group



Edit Button



Form



Gauge



Group Box

Button

A control that initiates an action.

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

You create a button by providing an action and a label. The action is either a method or closure property that does something when a user clicks or taps the button. The label is a view that describes the button's action — for example, by showing text, an icon, or both.

The label of a button can be any kind of view, such as a [Text](#) view for text-only labels:

```
Button(action: signIn) {  
    Text("Sign In")  
}
```

Or a [Label](#) view, for buttons with both a title and an icon:

```
Button(action: signIn) {  
    Label("Sign In", systemImage: "arrow.up")  
}
```

For those common cases, you can also use the convenience initializers that take a title string or [LocalizedStringKey](#) as their first parameter, and optionally a system image name or [ImageResource](#) as their second parameter, instead of a trailing closure:

Edit user interface elements

Edit the user interface elements in your source code to see the changes in the preview. To highlight elements in the source code that appear in the preview, click the Selectable mode below the canvas and click the elements in the preview. Then go back to the default Live mode to test and interact with your views in the preview.

The screenshot shows the Xcode interface with the project navigation bar at the top. Below it is the file browser showing the project structure, with 'ContentView' selected. The main editor area contains the Swift code for 'ContentView'. The bottom right corner of the editor shows '13 characters'. To the right of the editor is a preview window showing a smartphone screen with a white background. On the screen, there is a globe icon and the text 'Hello, world!' in a blue font.

```
1 //  
2 //  Content View  
3 //  Hello World  
4 //  
5 //  Created by Ravi Patel on 9/4/25.  
6 //  
7  
8 import SwiftUI  
9  
10 struct ContentView: View {  
11     var body: some View {  
12         VStack {  
13             Image(systemName: "globe")  
14                 .imageScale(.large)  
15                 .foregroundStyle(.tint)  
16             Text("Hello, world!")  
17                 .italic()  
18         }  
19         .padding()  
20     }  
21 }  
22  
23 #Preview {  
24     ContentView()  
25 }
```

Then use code completion, code intelligence, and the library of code snippets to help you write Swift and SwiftUI code. For more information, see [Editing source files in Xcode](#) and [Writing code with intelligence in Xcode](#).

See Also

Essentials

- [Creating an Xcode project for an app](#)
Start developing your app by creating an Xcode project from a template.
- [Previewing your app's interface in Xcode](#)
Iterate designs quickly and preview your apps' displays across different Apple devices.
- [Building and running an app](#)
Compile your source files and assemble an app bundle to run on a device or simulator.
- [Xcode updates](#)
Learn about important changes to Xcode.