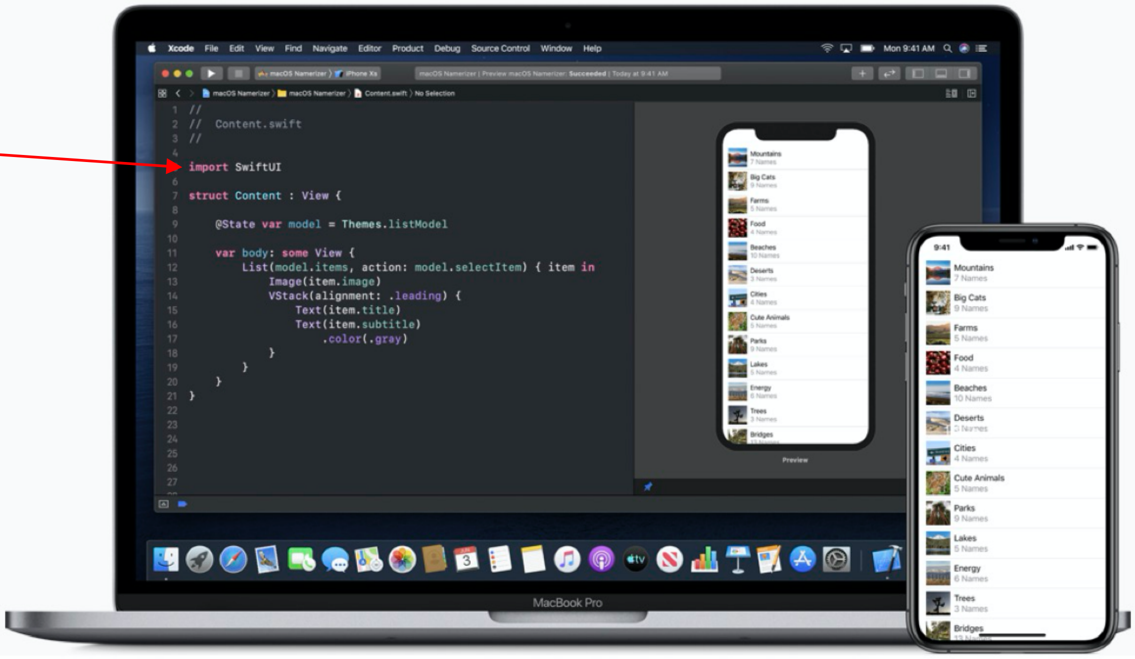
SwiftUI

Better apps. Less code.

SwiftUI is an innovative, exceptionally simple way to build user interfaces across all Apple platforms with the power of Swift. Build user interfaces for any Apple device using just one set of tools and APIs. With a declarative Swift syntax that's easy to read and natural to write, SwiftUI works seamlessly with new Xcode design tools to keep your code and design perfectly in sync. Automatic support for Dynamic Type, Dark Mode, localization, and accessibility means your first line of SwiftUI code is already the most powerful UI code you've ever written.

Software code to import framework



```
//
//  Content.swift
//

import SwiftUI

struct Content : View {

    @State var model = Themes.listModel

    var body: some View {
        List(model.items, action: model.selectItem) { item in
            Image(item.image)
            VStack(alignment: .leading) {
                Text(item.title)
                Text(item.subtitle)
                    .color(.gray)
            }
        }
    }
}
```

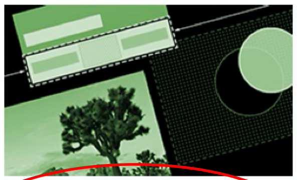
Declarative Syntax

SwiftUI uses a declarative syntax so you can simply state what your user interface should do. For example, you can write that you want a list of items consisting of text fields, then describe alignment, font, and color for each field. Your code is simpler and easier to read than ever before, saving you time and maintenance.

```
import SwiftUI

struct Content : View {

    @State var model = Themes.listModel

    var body: some View {
        List(model.items, action: model.selectItem) { item in
            Image(item.image)
            VStack(alignment: .leading) {
                Text(item.title)
                Text(item.subtitle)
                    .color(.gray)
            }
        }
    }
}
```

This declarative style even applies to complex concepts like animation. Easily add animation to almost any control and choose a collection of ready-to-use effects with only a few lines of code. At runtime, the system handles all of the steps needed to create a smooth movement, and even deals with interruption to keep your app stable. With animation this easy, you'll be looking for new ways to make your app come alive.

SwiftUI Tutorials



Receive step-by-step guidance and in-depth details on using SwiftUI with comprehensive tutorials.

View the tutorials ›

Videos and Documentation

Learn how SwiftUI and Xcode 11 seamlessly work together to make building user interfaces simple.

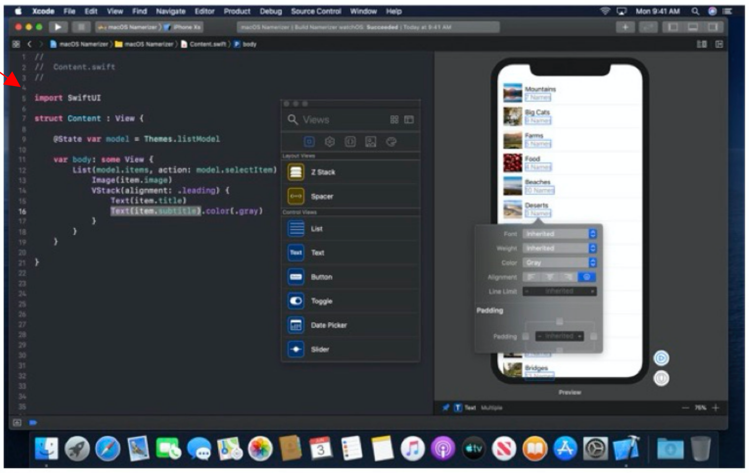
SwiftUI Videos

SwiftUI Reference

Design Tools

Xcode 11 includes intuitive new design tools that make building interfaces with SwiftUI as easy as dragging and dropping. As you work in the design canvas, everything you edit is completely in sync with the code in the adjoining editor. Code is instantly visible as a preview as you type, and any change you make to that preview immediately appears in your code. Xcode recompiles your changes instantly and inserts them into a running version of your app, visible, and editable at all times.

Software code to import framework


```
//
//  Content.swift
//

import SwiftUI

struct Content : View {

    @State var model = Themes.listModel

    var body: some View {
        List(model.items, action: model.selectItem)
            Image(item.image)
            VStack(alignment: .leading) {
                Text(item.title)
                Text(item.subtitle).color(.gray)
            }
        }
    }

}
```

Drag and drop. Arrange components within your user interface by simply dragging controls on the canvas. Click to open an inspector to select font, color, alignment, and other design options, and easily re-arrange controls with your cursor. Many of these visual editors are also available within the code editor, so you can use inspectors to discover new modifiers for each control, even if you prefer hand-coding parts of your interface. You can also drag controls from your library and drop them on the design canvas or directly on the code.

Dynamic replacement. The Swift compiler and runtime are fully embedded throughout Xcode, so your app is constantly being built and run. The design canvas you see isn't just an approximation of your user interface — it's your live app. And Xcode can swap edited code directly in your live app with "dynamic replacement", a new feature in Swift.

Previews. You can now create one or many previews of any SwiftUI views to get sample data, and configure almost anything your users might see, such as large fonts, localizations, or Dark Mode. Previews can also display your UI in any device and any orientation.

Native on All Apple Platforms

SwiftUI was built on decades of experience in creating the most innovative and intuitive user interfaces in the world. Everything users love about Apple ecosystems, such as controls and platform-specific experiences, is beautifully presented in your code. SwiftUI is truly native, so your apps directly access the proven technologies of each platform with a small amount of code and an interactive design canvas.

Tutorial – videos detailing steps to import framework and design software

Developer Discover Design Develop Distribute Support Account

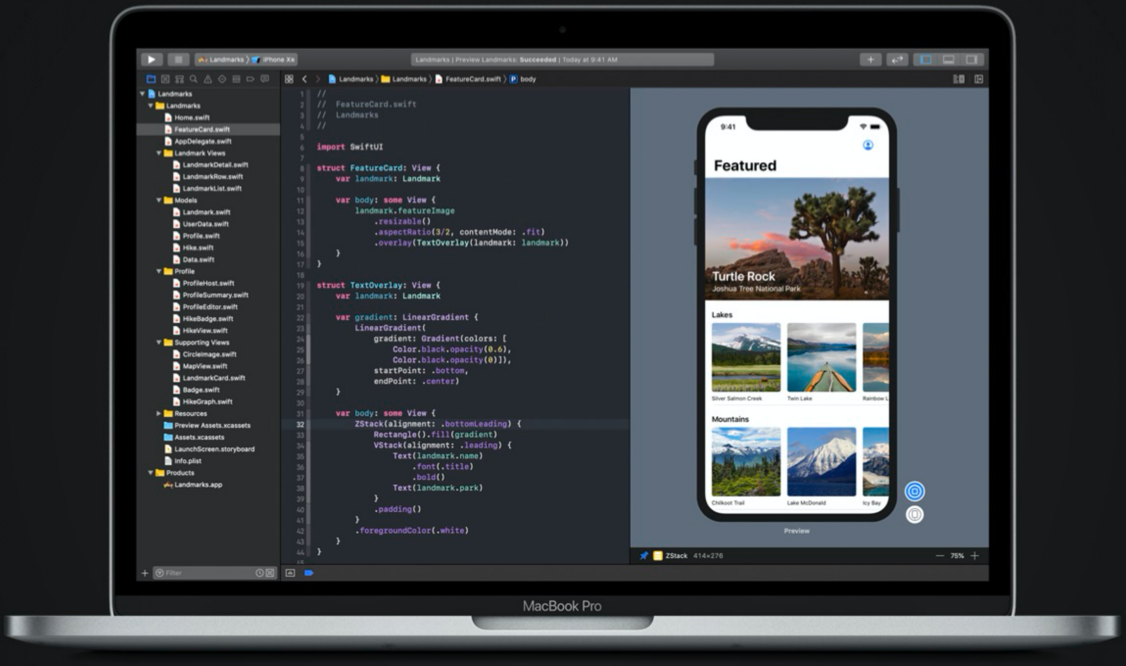
SwiftUI Tutorials

Introducing SwiftUI

SwiftUI is a modern way to declare user interfaces for any Apple platform. Create beautiful, dynamic apps faster than ever before.

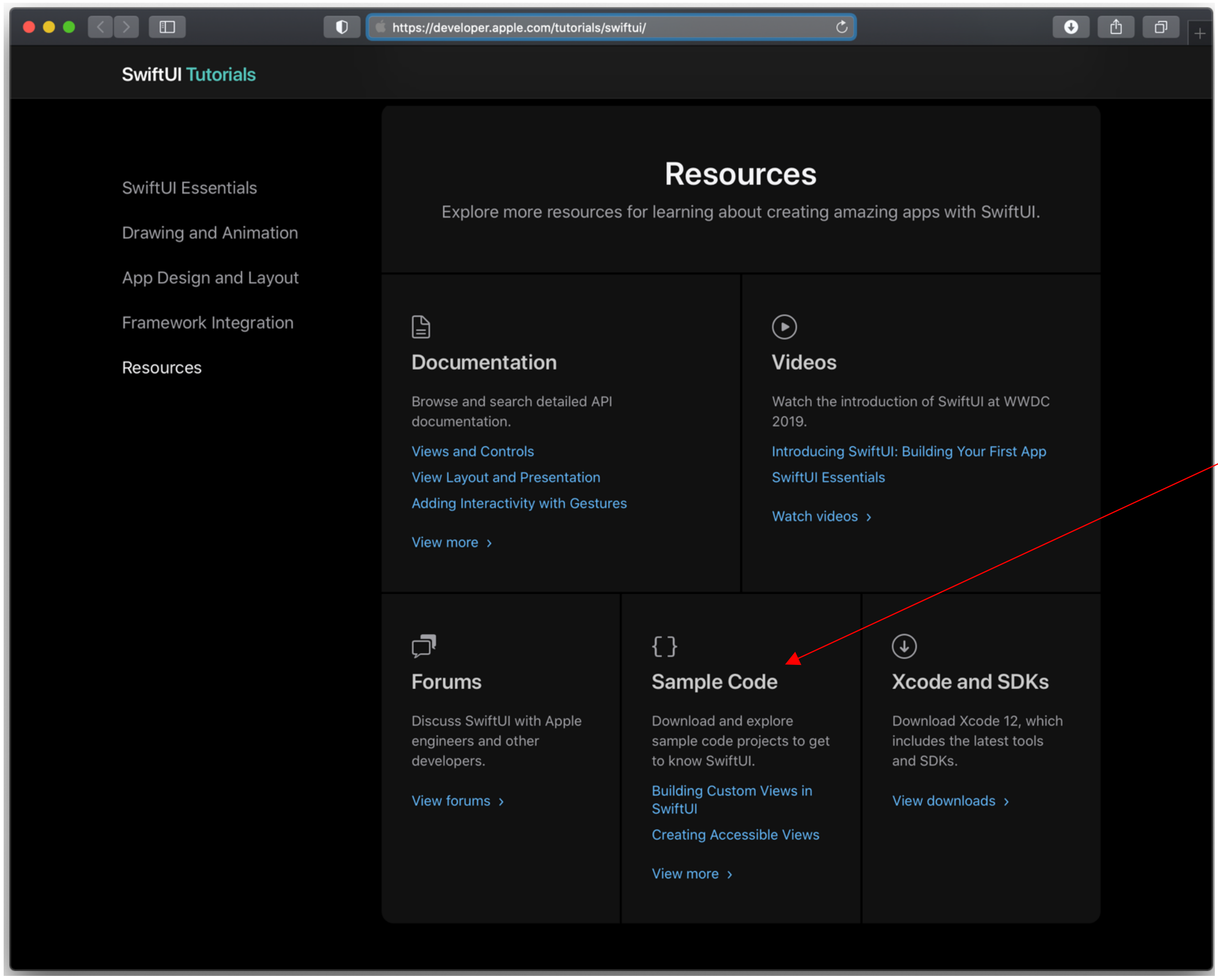
4hr 25min Estimated Time

Get started


Landmarks | Preview Landmarks: Succeeded | Today at 9:41 AM
Landmarks 〉 Landmarks 〉 FeatureCard.swift 〉 body
Featured
Turtle Rock
Joshua Tree National Park
Lakes
Mountains
Preview
MacBook Pro

```
//
//  FeatureCard.swift
//  Landmarks
//

import SwiftUI

struct FeatureCard: View {
    var landmark: Landmark

    var body: some View {
        landmark.featureImage
            .resizable()
            .aspectRatio(3/2, contentMode: .fit)
            .overlay(TextOverlay(landmark: landmark))
    }
}

struct TextOverlay: View {
    var landmark: Landmark

    var gradient: LinearGradient {
        LinearGradient(
            gradient: Gradient(colors: [
                Color.black.opacity(0.6),
                Color.black.opacity(0)]),
            startPoint: .bottom,
            endPoint: .center)
    }

    var body: some View {
        ZStack(alignment: .bottomLeading) {
            Rectangle().fill(gradient)
            VStack(alignment: .leading) {
                Text(landmark.name)
                    .font(.title)
                    .bold()
                Text(landmark.park)
            }
            .padding()
        }
        .foregroundColor(.white)
    }
}
```

Website accessed: October 19, 2020



Resources

Explore more resources for learning about creating amazing apps with SwiftUI.

Documentation

Browse and search detailed API documentation.

Views and Controls

View Layout and Presentation

Adding Interactivity with Gestures

View more ›

Videos

Watch the introduction of SwiftUI at WWDC 2019.

Introducing SwiftUI: Building Your First App

SwiftUI Essentials

Watch videos ›

Forums

Discuss SwiftUI with Apple engineers and other developers.

View forums ›

Sample Code

Download and explore sample code projects to get to know SwiftUI.

Building Custom Views in SwiftUI

Creating Accessible Views

View more ›

Xcode and SDKs

Download Xcode 12, which includes the latest tools and SDKs.

View downloads ›

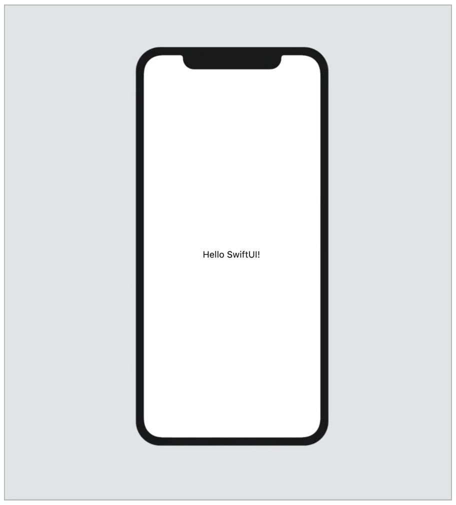
Sample code available for download

Section 1

Create a New Project and Explore the Canvas

Create a new Xcode project that uses SwiftUI. Explore the canvas, previews, and the SwiftUI template code.

To preview and interact with views from the canvas in Xcode, ensure your Mac is running macOS Catalina 10.15.



SwiftUI Tutorials

Creating and Combining Views › Create a New Project and Explore the Canvas (1 of 7)

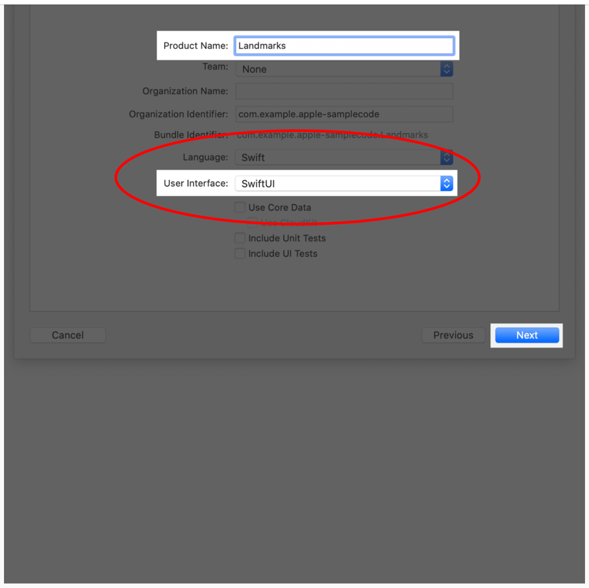
Step 1

Open Xcode and either click **Create a new Xcode project** in Xcode's startup window, or choose **File > New > Project**.

Step 2

In the template selector, select **iOS** as the platform, select the **Single View App** template, and then click **Next**.

Step 3


Product Name: Landmarks
Team: None
Organization Name:
Organization Identifier: com.example.apple-samplecode
Bundle Identifier: com.example.apple-samplecode.Landmarks
Language: Swift
User Interface: SwiftUI
☐ Use Core Data
☐ Use CloudKit
☐ Include Unit Tests
☐ Include UI Tests

Enter “Landmarks” as the Product Name, select **SwiftUI** for the user interface, and click **Next**. Choose a location to save the Landmarks project on your Mac.

Step 4

In the Project navigator, select `ContentView.swift`.

By default, SwiftUI view files declare two structures. The first structure conforms to the `View` protocol and describes the view’s content and layout. The second structure declares a preview for that view.

Step 5

In the canvas, click **Resume** to display the preview.

Tip

If the canvas isn’t visible, select **Editor > Editor and Canvas** to show it.

Step 6

Inside the `body` property, change “Hello World” to a greeting for yourself.

As you change the code in a view’s `body` property, the preview updates to reflect your changes.

Developer Discover Design Develop Distribute Support Account

Documentation SwiftUI App Structure and Behavior Fruta: Building a Feature-Rich App with SwiftUI Language: Swift

API Changes: None

Sample Code for download

Sample Code

Fruta: Building a Feature-Rich App with SwiftUI

Create a shared codebase to build a multiplatform app that offers widgets and an App Clip.

Download

Availability

iOS 14.0+

macOS 11.0+ Beta

Xcode 12.2+ Beta

Framework

SwiftUI

On This Page

Overview

See Also

Overview

Note

This sample project is associated with WWDC 2020 sessions 10637: Platforms State of the Union, 10146: Configure and Link Your App Clips, 10120: Streamline Your App Clip, 10118: Create App Clips for Other Businesses, 10096: Explore Packages and Projects with Xcode Playgrounds, and 10028: Meet WidgetKit.

The Fruta sample app builds an app for macOS, iOS, and iPadOS that implements SwiftUI platform features like widgets or App Clips. Users can order smoothies, save favorite drinks, collect rewards, and browse recipes.

The sample app's Xcode project includes widget extensions that enable users to add a widget to their iOS Home screen or the macOS Notification Center and view their rewards or a favorite smoothie. The Xcode project also includes an App Clip target. With the App Clip, users can discover and instantly launch some of the app's functionality on their iPhone or iPad without installing the full app.

The Fruta sample app leverages Sign in with Apple and PassKit (Apple Pay and Wallet) to provide a streamlined user experience, and promotes code reuse by bundling shared code and localized assets as Swift Packages.

Configure the Sample Code Project

To build this project for iOS 14, use Xcode 12. The runtime requirement is iOS 14. To build this project for macOS 11 Big Sur beta 9, use Xcode 12.2 beta 2.

1. To run on your devices, including on macOS, set your team in the targets' Signing & Capabilities panes. Xcode manages the provisioning profiles for you.

Sample Code for download

Sample Code

Building Custom Views in SwiftUI

Create a custom view with data driven transitions and animations in SwiftUI.

Download

Availability

macOS 10.15+

Xcode 11.0+

Framework

SwiftUI

On This Page

Overview

See Also

Overview

Note

This sample code project is associated with WWDC 2019 session 237: Building Custom Views in SwiftUI.

For more information, see Drawing and Animation.

Configure the Sample Code Project

Before you run the sample code project in Xcode:

- Use Xcode 11 beta.
- To run on a device, chose your development team in the Signing and Capabilities tab of the target settings.

See Also

Essentials

`protocol Shape`

A 2D shape that you can use when drawing a view.

Framework

SwiftUI

Declare the user interface and behavior for your app on every platform.

Availability
iOS 13.0+
macOS 10.15+
Mac Catalyst 13.0+
tvOS 13.0+
watchOS 6.0+

Framework
SwiftUI

On This Page
Overview
Topics

Overview

SwiftUI provides views, controls, and layout structures for declaring your app's user interface. The framework provides event handlers for delivering taps, gestures, and other types of input to your app, and tools to manage the flow of data from your app's models down to the views and controls that users will see and interact with.

Define your app structure using the App protocol, and populate it with scenes that contain the views that make up your app's user interface. Create your own custom views that conform to the `View` protocol, and compose them with SwiftUI views for displaying text, images, and custom shapes using stacks, lists, and more. Apply powerful modifiers to built-in views and your own views to customize their rendering and interactivity. Share code between apps on multiple platforms with views and controls that adapt to their context and presentation.

You can integrate SwiftUI views with objects from the UIKit, AppKit, and WatchKit frameworks to take further advantage of platform-specific functionality. You can also customize accessibility support in SwiftUI, and localize your app's interface for different languages, countries, or cultural regions.

Topics

Essentials

Introducing SwiftUI
SwiftUI is a modern way to declare user interfaces for any Apple platform. Create beautiful, dynamic apps faster than ever before.

App Structure and Behavior
Define the entry point and top-level organization of your app.

User Interface

Views and Controls
Present your content onscreen and handle user interactions.

View Layout and Presentation
Combine views in stacks, generate groups and lists of views dynamically, and define view presentations and hierarchy.

Drawing and Animation
Enhance your views with colors, shapes, and shadows, and customize animated transitions between view states.

Framework Integration
Integrate SwiftUI views into existing apps, and embed AppKit, UIKit, and WatchKit views and controllers into SwiftUI view hierarchies.

Data and Events

State and Data Flow
Control and respond to the flow of data and changes within your app's models.

Gestures
Define interactions from taps, clicks, and swipes to fine-grained gestures.

Previews in Xcode

Previews
Generate dynamic, interactive previews of your custom views.