Develop in Swift Tutorials | Educator Guide

Develop in Swift Tutorials introduce app development with Swift and Xcode for anyone learning how to develop for Apple platforms.

In each chapter, learners will complete:

A tutorial

- Coding a project, ranging from an app prototype to a fully functioning app
- Building on prior knowledge, getting progressively more challenging

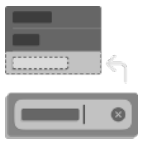
A wrap-up

- Review of concepts
- Ideas for extending an app
- Suggestions for how to apply skills in a different context, often by creating a new project

Choose your approach: You can present the content linearly, or you can incorporate other text, documentation, tutorials, videos, and projects to fit your needs. One option is to have learners complete the tutorial independently, then choose items from the “Continue practicing” section to complete together, allowing learners to work collaboratively and ask questions.

SwiftUI foundations

Get familiar with the tools and technologies you'll use to create apps.

	Chapter	Description	Topics and skills			Estimated time
	**Explore Xcode**	Get to know Xcode and SwiftUI by creating a prototype of a messaging app. Learn about syntax for Swift and how to use the source editor and preview.	• Background • **`Color`** • Creating a new project • Dot notation	• Modifiers • Padding • **`String`** • Swift syntax • **`Text`**	• Views • Xcode error messages • Xcode Library	1 hr
	**Views, structures, and properties**	Learn how to build a custom view to create a multiday weather forecast. In your view, you'll use properties to customize the display for each day.	• Arguments and parameters • **`Bool`** • Computed properties • Custom subviews • **`Font`**	• Foreground style • **`Image`** • Initializers • **`Int`** • **`HStack`** and **`VStack`** • Returning a value	• SF Symbols • Stored properties • String interpolation • Structures • Subviews • Type annotation	1.5 hrs
	**Layout and style**	Build two onboarding screens for an iOS app to learn useful tools for putting views where you want them onscreen and inspecting their size. Define new colors in the asset catalog and use them to create gradient backgrounds.	• Accent color • Arrays • Borders • Brightness • Color assets • Customizing a preview	• **`Font`** • Frames • **`Gradient`** • **`Image`** • Pinning a preview • **`Shape`** • **`Spacer`**	• **`TabView`** • Transparency • Type inference • **`ZStack`**	1.5 hrs
	**Buttons and state**	Explore adding buttons to your apps. Learn about Swift closures and their relationship to buttons. Use state properties to update the user interface automatically.	• Animation • Aspect ratio • Assignment operator • **`Button`** • Button styles • Closures • **`Color`**	• Disabling controls • Dynamic sizing • Equality operator • **`ForEach`** • Hierarchical SF Symbols • Randomization	• Range operator • Resizable images • **`@State`** • Trailing closure syntax • View tint	1.5 hrs
	**Lists and text fields**	Create a dynamic interface that stores a set of items in an array and displays them using lists. Use text fields and bindings to let people enter text.	• Arrays • Adding and removing from arrays • Bindings • Buttons with custom labels	• Disabling autocorrection • Clip shapes • **`ForEach`** • **`List`** • Not (**`!`**) operator	• Symbol rendering modes • Ternary conditional operator • **`TextField`** • **`Toggle`**	1.5 hrs

Data modeling

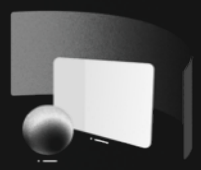
Model real-world concepts and relationships by creating and testing your own custom types.

	Chapter	Description	Topics and skills			Estimated time
	**Custom types and Swift Testing**	Define your first data model by making your own custom types, and prove they work correctly with unit tests. Then use your custom types to keep track of scores in a game.	• Creating a type to contain your app's logic • Creating **`enum`** types • Creating **`struct`** types • Creating unit tests • Fixing test failures	• **`Grid`** and **`GridRow`** • **`Identifiable`** and **`UUID`** • **`.opacity`** and **`.disabled`**	• Running tests • Swift file creation	1 hr
	**Models and persistence**	Build a list of your friends' birthdays, using SwiftData to save and retrieve that data across launches.	• **`Calendar`** • Classes • Data models • **`Date`** • Date formatting	• **`DatePicker`** • **`@Environment`** • Frameworks • **`@Model`** macro • **`NavigationStack`**	• **`@Query`** macro • Safe area • SwiftData context	1.5 hrs
	**Navigation, editing, and relationships**	Create an app to track friends and their favorite movies using SwiftData to manage the model objects. Use a query to display the items in a list, and make a detail view to edit them. Then learn how to create and display relationships between friends and movies, and explore how to create advanced queries.	• **`@Bindable`** • **`ContentUnavailableView`** • Creating sample data • Custom view initializers • Environment **`dismiss`** value • **`Form`** • **`Group`** • Modal interfaces • Multiple previews	• **`ModelConfiguration`** **`ModelContainer`** • Model relationships • Navigation hierarchies • **`NavigationLink`** • **`NavigationSplitView`** • Or (**`||`**) operator • **`Picker`** • Predicate • Property wrappers	• Refactoring • **`Schema`** • Search • **`Section`** • Sheets • Sorting arrays • Toolbars • View tags	3.5 hrs
	**Observation and shareable data models**	Power an alphabet game using Observation. Share a complex data model with many independent views.	• **`Dictionary`** • Documentation comments • **`@Observable`** • **`onChange`**	• Sharing your types through the environment	• **`Task.sleep`** • Xcode's Quick Help and jump bar • **`zip`**	2 hrs

Machine learning

Use machine learning technologies to enhance your apps.

Chapter	Description	Topics and skills			Estimated time
Natural language	Build a sentiment analysis app and use the Natural Language framework to analyze responses to an open-ended survey prompt.	• **`@FocusState`** wrapper • **`Chart`** • **`chartProxy`** • Charts framework • **`GeometryReader`**	• Insert versus append • Natural Language framework • **`NLTagger`** • Plottable protocol	• **`ScrollView`** • Sentiment analysis • **`Textfield.axis`**	1 hr
Recognize text in images	Create an app that uses the Vision Framework and the Translation API to translate text on signs.	• Alert for app processing time • **`ImageResource`** • Overlays	• **`RecognizedTextObservation`** and **`RecognizeTextRequest`** • **Shape** • Translation framework	• **`.translationPresentation`** • **`ViewModifier`** protocol • Vision framework	1 hr
Model training with Create ML	Use Xcode's Create ML tool to train a model to estimate the anticipated difficulty of a hike using provided data.	• Create ML tool in Xcode • CSV files • Machine learning algorithms	• Model accuracy • Previewing output	• Training, validation, and testing data • Xcode developer tools	1 hr
Custom models with Core ML	Integrate a custom machine learning model into an app that predicts the difficulty of an upcoming hike.	• Adding a Core ML model to an app • **`CaseIterable`** protocol	• Core ML framework • Generic views	• Segmented pickers • View builders	1 hr



Spatial computing

Design app experiences for spatial computing.

	Chapter	Description	Topics and skills			Estimated time
	**Windows in visionOS**	Create your first visionOS app with a window using SwiftUI.	• **`Circle`** • **`ColorPicker`** • **`Double`** • **`Grid`**	• **`GridRow`** • Padding for 3D views • Remainder (**`%`**) operator • **`Slider`**	• visionOS simulator • Window resizability • Windows	1 hr
	**Ornaments and multiple windows**	Create multiple windows in visionOS using SwiftUI. Use ornaments to provide access to frequently used controls without crowding or obscuring window contents.	• **`@Environment isEnabled`** • **`@Environment openWindow`** • **`.glassBackgroundEffect`** • **`@Previewable`** previews	• **`TextField`** word wrapping • visionOS **`.ornament`**	• **`WindowGroup`**, **`.windowStyle`**, and **`.windowResizability`**	1 hr
	**Volumes in visionOS**	View 3D content from any angle in the Shared Space using Reality Composer Pro and SwiftUI.	• Arrays • **`DragGesture`** • Environment **`openWindow`** value • **`Model3D`**	• **`NavigationSplitView`** • Reality Composer Pro • Rotation in three dimensions	• Toolbars • Volumes • **`WindowGroup`**	1.5 hrs