Publication date : 04/15/2024

Course : SwiftUI, developing applications for the Apple ecosystem

Practical course - 3d - 21h00 - Ref. WUI
Price : 1940 CHF E.T.

4,7 / 5

SwiftUI is Apple's new framework for creating graphical interfaces for iPhone, iPad, Mac, Apple Watch, Apple TV and Apple Vision Pro. At the end of the course, you'll be able to develop applications for the Apple ecosystem, and particularly the iPhone and iPad, using SwiftUI.

Teaching objectives

At the end of the training, the participant will be able to:

- Understanding the advantages and special features of SwiftUI compared with UIkit
- Swift language features useful for SwiftUI
- Create rich, flexible and versatile interfaces
- Understanding SwiftUI's data flow
- Create a complete connected and persistent application
- Building views around XCode previews
- Integrating SwiftUI into existing UIkit projects

Intended audience

Developers and architects.

Prerequisites

Knowledge of the Swift language and XCode environment.

Course schedule

1 SwiftUI concept

- What is SwiftUI and why a new framework?
- Fundamental differences with UIkit.
- SwiftUI with iOS, iPadOS, macOS, watchOS, tvOS and visionOS.

PARTICIPANTS

Developers and architects.

PREREQUISITES

Knowledge of the Swift language and XCode environment.

TRAINER QUALIFICATIONS

The experts leading the training are specialists in the covered subjects. They have been approved by our instructional teams for both their professional knowledge and their teaching ability, for each course they teach. They have at least five to ten years of experience in their field and hold (or have held) decision-making positions in companies.

ASSESSMENT TERMS

The trainer evaluates each participant's academic progress throughout the training using multiple choice, scenarios, hands-on work and more.
Participants also complete a placement test before and after the course to measure the skills they've developed.

TEACHING AIDS AND TECHNICAL RESOURCES

• The main teaching aids and instructional methods used in the training are audiovisual aids, documentation and course material, hands-on application exercises and corrected exercises for practical training courses, case studies and coverage of real cases for training seminars.
• At the end of each course or seminar, ORSYS provides participants with a course evaluation questionnaire that is analysed by our instructional teams.
• A check-in sheet for each half-day of attendance is provided at the end of the training, along with a course completion certificate if the trainee attended the entire session.

2 Swift features

- Opaque return types.
- Property wrappers.
- Function builders.
- Asynchronous with async/await.
- The MainActor.

Hands-on work

Practical application of Swift features useful for SwiftUI.

3 Interface composition

- SwiftUI application architecture.
- XCode previews.
- Create views.
- Simple views: text, labels, images, buttons, shapes...
- Style modifiers.
- The different types of layout.
- Interactions and gestures.
- Separate, reuse and test views.

Hands-on work

Creating and composing multiple views. Use XCode previews.

4 Status management

- How data flow works in SwiftUI.
- States and bindings.
- State views: textfields, toggles, sliders, pickers...
- Animations and transitions.
- The observable macro.
- Environments.
- Dependency injection.
- AppStorage and SceneStorage.
- The life cycle of a view.

Hands-on work

Using states and bindings. Using animations. Dependency injection.

5 Lists and navigation

- Dynamic lists: lists, grids, forms, lazy...
- Navigating between views.
- Modal views and alerts.
- Toolbars and menus.
- TabView and SplitView.

Hands-on work

Create dynamic lists with a navigation stack. Creation of modal views.

TERMS AND DEADLINES

Registration must be completed 24 hours before the start of the training.

ACCESSIBILITY FOR PEOPLE WITH DISABILITIES

Do you need special accessibility accommodations? Contact Mrs. Fosse, Disability Manager, at psh-accueil@orsys.fr to review your request and its feasibility.

6 Network and persistence

- Making network calls with SwiftUI.
- Asynchronous and error management.
- Data persistence with SwiftData.

Hands-on work
Creating a connected application with basic persistence using SwiftData.

7 SwiftUI integration

- Integrate SwiftUI views into a UIkit project.
- Integrate UIkit components into a SwiftUI project.

Dates and locations

REMOTE CLASS
2026 : 1 July, 9 Nov.