Glossary

Introduction to Flutter and Dart

Welcome! This alphabetized glossary contains many terms used in this module. Understanding these terms is essential when working in the industry, participating in user groups, and participating in other certificate programs.

Term	Definition
Code reusability	The ability of Flutter to provide a single codebase to build applications for iOS, Android, web, and desktop platforms. React Native and Xamarin allow code reuse primarily between iOS and Android, with platform-specific adjustments.
Dart Analyzer	A static analysis tool that checks Dart code for errors, potential issues, and coding style, offering real-time feedback in IDEs like Visual Studio Code.
Dart SDK	The programming language used in Flutter. It is optimized for UI development and offers strong typing, asynchronous programming, and both JIT and AOT compilation.
Emulators	Simulated devices for testing apps across platforms and screen sizes, enabling early detection of issues.
Flutter DevTools	A suite of Flutter-specific debugging and performance tools, featuring a widget inspector, timeline view, memory profiler, and performance view.
Flutter Doctor	A command-line tool that checks the Flutter development environment for issues with dependencies or configurations, aiding in troubleshooting.
Flutter Engine	The C++ runtime environment for Flutter apps, using Google's Skia library for fast, smooth rendering.
Flutter SDK	A complete toolkit with libraries, tools, and APIs for developing, testing, and deploying Flutter apps, including UI rendering libraries.
Hot reload	A feature that enables instant viewing of code changes without restarting the app, preserving its current state.
Integrated development environment (IDE)	Tools like Visual Studio Code, Android Studio, and IntelliJ IDEA that support Flutter development with features such as code completion, debugging, and emulation.
ListView	A widget that displays a list of items in a scrollable view.
Native development	Creating apps for iOS or Android using native languages (Swift, Kotlin, and Java) and tools (Xcode and Android Studio) for optimal performance and control.
Packages	Pre-written code or libraries on pub.dev that enhance Flutter projects with additional functionality, including UI components and networking utilities.
Performance comparison	Flutter compiles directly to ARM code for near-native performance, while React Native uses a bridge between JavaScript and native code, and Xamarin compiles C# to native code.
Performance view	A Flutter DevTools tool that offers insights into app performance metrics, helping identify bottlenecks and optimize responsiveness.
Physical devices	Real mobile devices used to test Flutter applications for performance, interaction, and debugging, offering more accurate feedback than emulators.
Provider	A state management library that allows widgets to efficiently listen and react to changes in app state.
React Native	A popular cross-platform framework by Facebook that enables mobile app development using JavaScript and React, with some reliance on native components for rendering.
SQLite	A local database for storing structured data in Flutter applications, ideal for persisting user data across sessions with a SQL-like interface.
Stateful widget	A widget that retains its state and can change over time in response to user input or app changes.
Stateless widget	A widget that does not maintain state and is used for displaying static content.
Timeline view	A Flutter DevTools tool that monitors app performance over time, aiding developers in diagnosing issues like slow rendering or excessive memory usage.
Widget Inspector	A Flutter DevTools tool that visually represents the widget tree, enabling developers to inspect the layout and structure of their UI.
Xamarin	A Microsoft cross-platform framework that enables mobile app development using C# and .NET, allowing shared codebases for iOS, Android, and Windows.

Author(s)

Skills Network