FLUTTER TUTORIAL

When it's said that Flutter has "one codebase," it means that you can use a single codebase to develop apps that run on multiple platforms, such as iOS, Android, web, desktop, and potentially more. This is made possible because Flutter uses a single programming language (Dart) and a single codebase to create applications that can run natively on different platforms.

Traditionally, when developing apps for multiple platforms, developers would need to write separate codebases for each platform. For example, one codebase for Android (using Java or Kotlin) and another for iOS (using Swift or Objective-C). This approach often results in duplicated effort, increased development time, and maintenance challenges.

With Flutter, however, developers can write code once and use it across different platforms, thanks to its cross-platform framework. This is achieved through Flutter's rendering engine, which uses Skia to draw widgets and UI components directly onto the screen, providing a consistent look and feel across platforms.

By having only one codebase, developers can streamline the development process, reduce code duplication, and maintain a single code repository for all platforms. This not only saves time and effort but also ensures consistency and easier maintenance of the app across different platforms.

Yes, Flutter does involve two compilation processes, primarily due to its architecture and the use of Dart as its programming language.

Dart Compilation: Dart, the programming language used for Flutter app development, undergoes compilation. Dart code is compiled ahead of time (AOT) or just-in-time (JIT) depending on the platform and the compilation mode. When compiling ahead of time (AOT), Dart code is compiled to native machine code, which improves app startup time and performance. When compiling just-in-time (JIT), Dart code is compiled to machine code at runtime, which allows for hot reload during development.

Flutter Compilation: Flutter itself also involves a compilation process. The Flutter framework, written in Dart, provides a set of widgets and tools for building user interfaces. When you write Flutter code, you are essentially writing code that utilizes the Flutter framework. When you run a Flutter app, the Flutter framework code, along with your Dart code, is compiled and translated into native code specific to the target platform (iOS, Android, web, etc.). This compilation process involves translating Flutter's widget tree into platform-specific UI components.

So, in summary, both Dart and Flutter involve compilation processes, with Dart being compiled either ahead of time or just-in-time, and Flutter code being compiled into native code specific to the target platform.