1. Overview

Flutter is an open-source UI software development kit (SDK) created by **Google** for building **natively compiled applications** for **mobile, web, and desktop** from a **single codebase**.

- First released in 2017.
- Uses **Dart** programming language.
- Allows building **fast, expressive, and flexible UIs**.
- Supports **hot reload** for rapid development.
- Targets platforms: Android, iOS, Windows, macOS, Linux, and Web.

---

2. Key Concepts

Widgets

- Everything in Flutter is a **widget** — UI elements, layout structures, and even the app itself.
- Widgets are **immutable** and describe how the UI should look.
- Types:
    - **StatelessWidget**: Immutable, UI doesn't change over time.
    - **StatefulWidget**: Maintains mutable state that can change during runtime.

```
class MyWidget extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Text('Hello, Flutter!');
 }
}
```

UI Composition & Layout

- Widgets can be combined and nested to build complex UIs.
- Common layout widgets:
    - Container

    - Row and Column
    - Stack
    - Expanded and Flexible
    - Padding, Margin
- Uses a **declarative** style for UI.

State Management

- Flutter manages UI state to trigger UI updates.
- Several approaches:
    - **setState()** (basic)
    - Provider
    - Bloc/Cubit
    - Riverpod
    - Redux
    - MobX
- Choosing state management depends on app complexity.

Flutter Architecture

- **Flutter Engine**: Built in C++, handles rendering, accessibility, plugins.
- **Dart Framework**: Widget library and framework APIs.
- **Embedder**: Platform-specific code for iOS, Android, web, desktop.

Rendering

- Flutter doesn't use native UI components.
- Draws widgets directly on a **canvas** via **Skia Graphics Engine**.
- Allows consistent UI across platforms.

Hot Reload & Hot Restart

- **Hot Reload**: Injects updated source code into the running app without full restart. Fast iteration.
- **Hot Restart**: Restarts the app, but preserves some state.

---

3. Core Flutter Libraries

Library	**Purpose**
material.dart	Material Design widgets and themes
cupertino.dart	iOS-style widgets
widgets.dart	Base widget library
animation.dart	Animation APIs
foundation.dart	Core framework classes

---

4. Navigation & Routing

- Flutter supports navigation stacks with routes.
- Simple example:

```
Navigator.push(context, MaterialPageRoute(builder: (context) => SecondPage()));
Navigator.pop(context);
```

- Also supports named routes and deep linking.

---

5. Plugins & Packages

- Rich ecosystem on pub.dev.
- Access native device features: camera, GPS, sensors, storage.
- Popular packages:
  - http for networking
  - provider for state management
  - shared_preferences for local storage
  - firebase_core for Firebase integration

---

6. Integration with Native Code

- Platform channels allow Flutter to call native Android (Java/Kotlin) or iOS (Swift/Objective-C) code.
- Useful for functionality not available via Flutter plugins.

---

7. Debugging & Testing

- Flutter DevTools: Profiling, performance, widget inspection.
- Unit testing with Dart's test package.
- Widget testing (UI tests).
- Integration testing (end-to-end tests).

---

8. Build & Deployment

- Flutter apps compile to native ARM machine code.
- Supports:
    - Android APK / AAB builds.
    - iOS IPA builds.
    - Web builds as JavaScript apps.
    - Desktop builds (macOS, Windows, Linux).

---

9. Advantages

- Single codebase for multiple platforms.
- High performance due to native compilation.
- Customizable UIs with rich widget library.
- Strong community and Google support.
- Fast development cycle (hot reload).