# Cross Platform Mobile Application Development

# Introduction to flutter

- In general, creating a mobile application is a very complex and challenging task.
- There are many frameworks available, which provide excellent features to develop mobile applications.
- For developing mobile apps, Android provides a native framework based on Java and Kotlin language, while iOS provides a framework based on Objective-C/Swift language.
- Thus, we need two different languages and frameworks to develop applications for both OS.
- Today, to overcome form this complexity, there are several frameworks have introduced that support both OS along with desktop apps.
- These types of the framework are known as cross-platform development tools.

## Introduction

- The cross-platform development framework has the ability to write one code and can deploy on the various platform (Android, iOS, and Desktop).
- It saves a lot of time and development efforts of developers.
- There are several tools available for cross-platform development, including web-based tools, such as Ionic from
- Drifty Co. in 2013, Phonegap from Adobe, Xamarin from Microsoft, and React Native form Facebook.
- Each of these frameworks has varying degrees of success in the mobile industry.
- In recent, a new framework has introduced in the cross-platform development family named Flutter developed from Google.

## What is Flutter?

- Flutter is a UI toolkit for building fast, beautiful, natively compiled applications for mobile, web, and desktop with one programing language and single codebase.
- It is free and open-source. Initially, it was developed from Google and now manages by an ECMA (European Computer Manufacturers Association) standard. ECMA a nonprofit standards organization for information and communication systems.
- Flutter apps use Dart programming language for creating an app.
- The first version of Flutter was announced in the year 2015 at the Dart Developer Summit.

## Flutter

- It was initially known as codename Sky and can run on the Android OS. On December 4, 2018, the first stable version of the Flutter framework was released, denoting Flutter 1.0.
- Latest version : 3.19

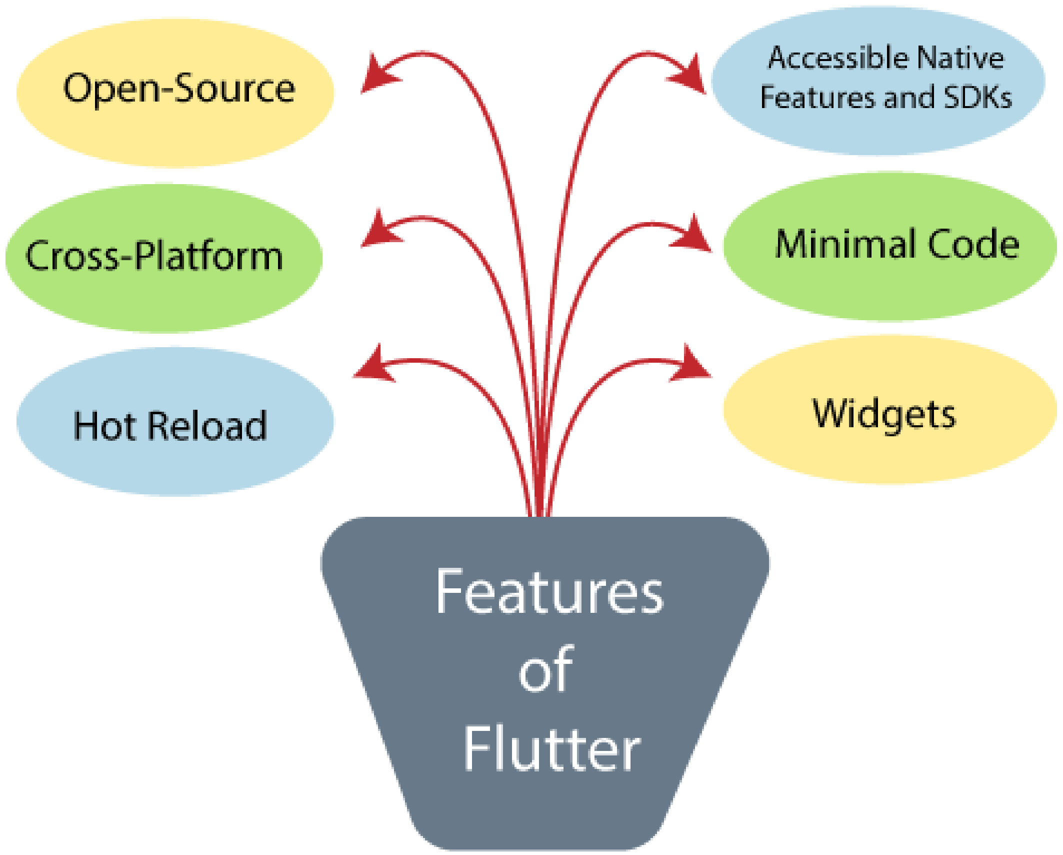
## Flutter

- Flutter apps use Dart programming language for creating an app.
- The dart programming shares several same features as other programming languages, such as Kotlin and Swift, and can be trans-compiled into JavaScript code.
- A source-to-source translator, source-to-source compiler (S2S compiler), transcompiler, or transpiler is a type of translator that takes the source code of a program written in a programming language as its input and produces an equivalent source code in the same or a different programming language.
- Flutter is mainly optimized for 2D mobile apps that can run on both Android and iOS platforms.
- We can also use it to build full-featured apps, including camera, storage, geolocation, network, third-party SDKs, and more.

## What makes flutter unique?

- Flutter is different from other frameworks because it neither uses WebView nor the OEM widgets that shipped with the device.
- Instead, it uses its own high-performance rendering engine to draw widgets.
- It also implements most of its systems such as animation, gesture, and widgets in Dart programing language that allows developers to read, change, replace, or remove things easily.
- It gives excellent control to the developers over the system

## Features of Flutter



- **Open-Source:** Flutter is a free and open-source framework for developing mobile applications.
- **Cross-platform:** This feature allows Flutter to write the code once, maintain, and can run on different platforms. It saves the time, effort, and money of the developers.
- **Hot Reload:** Whenever the developer makes changes in the code, then these changes can be seen instantaneously with Hot Reload. It means the changes immediately visible in the app itself.
- **Accessible Native Features and SDKs:** This feature allows the app development process easy and delightful through Flutter's native code, third-party integration, and platform APIs.

- **Minimal code:** Flutter app is developed by Dart programming language, which uses JIT(Just-in-Time) and AOT(Ahead-of-Time) compilation to improve the overall start-up time, functioning and accelerates the performance.
- AoT Compilation: Occurs before the program is executed, typically at build time or during installation. The entire source code is translated into machine code or an intermediate representation ahead of running the program.
- JIT Compilation: Takes place at runtime, as the program is executing
- **Widgets**: The Flutter framework offers widgets, which are capable of developing customizable specific designs. Most importantly, Flutter has two sets of widgets: Material Design and Cupertino widgets that help to provide a glitch-free experience on all platforms.

## Advantages of Flutter

- Dart has a large repository of software packages which lets you to extend the capabilities of your application.
- Developers need to write just a single code base for both applications (both Android and iOS platforms). Flutter may to be extended to other platform as well in the future.
- Flutter needs lesser testing. Because of its single code base, it is sufficient if we write automated tests once for both the platforms.
- With Flutter, developers has full control over the widgets and its layout.

## Disadvantages of Flutter

- Since it is coded in Dart language, a developer needs to learn new language (though it is easy to learn).
- Modern framework tries to separate logic and UI(user interface) as much as possible but, in Flutter, user interface and logic is intermixed. We can overcome this using smart coding and using high level module to separate user interface and logic.
- Flutter is yet another framework to create mobile application. Developers are having a hard time in choosing the right development tools in hugely populated segment.

## Flutter widgets

- Each element on a screen of the Flutter app is a widget.
- At the core of Flutter is the Flutter engine, which is mostly written in C++ and supports the primitives necessary to support all Flutter applications.
- Widgets are basically user interface components used to create the user interface of the application.
- The view of the screen completely depends upon the choice and sequence of the widgets used to build the apps. And the structure of the code of an apps is a tree of widgets.

## Category of Widgets:14

- Accessibility: These are the set of widgets that make a flutter app more easily accessible.
- Animation and Motion: These widgets add animation to other widgets.
- Assets, Images, and Icons: These widgets take charge of assets such as display images and show icons.
- Async: These provide async functionality in the flutter application.
- Basics: These are the bundle of widgets that are absolutely necessary for the development of any flutter application.
- Cupertino: These are the iOS designed widgets.An iOS-style full-screen modal route that opens when the child is long-pressed. Used to display relevant actions for your content.
- Input: This set of widgets provides input functionality in a flutter application.

- Interaction Models: These widgets are here to manage touch events and route users to different views in the application.
- Layout: This bundle of widgets helps in placing the other widgets on the screen as needed.
- Material Components: This is a set of widgets that mainly follow material design by Google.
- Painting and effects: This is the set of widgets that apply visual changes to their child widgets without changing their layout or shape.
- Scrolling: This provides scrollability of to a set of other widgets that are not scrollable by default.
- Styling: This deals with the theme, responsiveness, and sizing of the app.
- Text: This displays text.

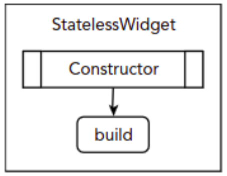
## What are RenderObjects?

- RenderObjects are those particular “Objects” responsible for controlling the sizes, layouts, and logic used for painting widgets to the screen and forming the UI for the application.
- There are some specific instances where a super complex design needs precise implementation.
- It might not be entirely possible to use widgets or make a particular widget easier to use by building it from scratch with more spice. In instances like these, RenderObjects would be the right tool to use.

# StatelessWidget Lifecycle

"The State is information that can read synchronously when the widget is build and might change during the lifetime of the widget."

- A StatelessWidget is built based on its own configuration and does not change dynamically.



- For example, the screen displays an image with a description and will not change.

## StatefulWidget Lifecycle

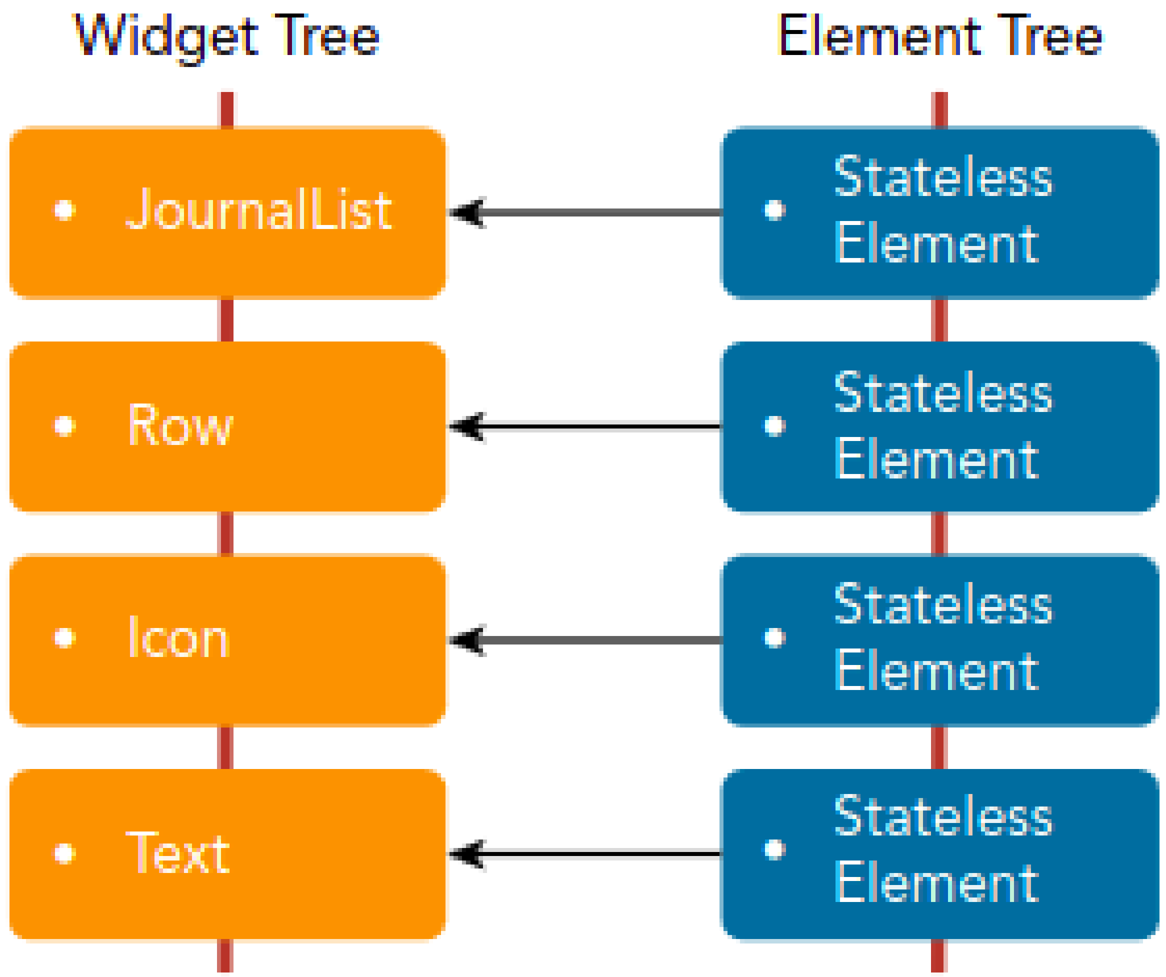
- A StatefulWidget is built based on its own configuration but can change dynamically.
- For example,the screen displays an icon with a description, but values can change based on the user’s interaction,like choosing a different icon or description.
- This type of widget has a mutable state that can change over time.
- The stateful widget is declared with two classes, the StatefulWidget class and the Stateclass

## Differences Between Stateless and Stateful Widget:

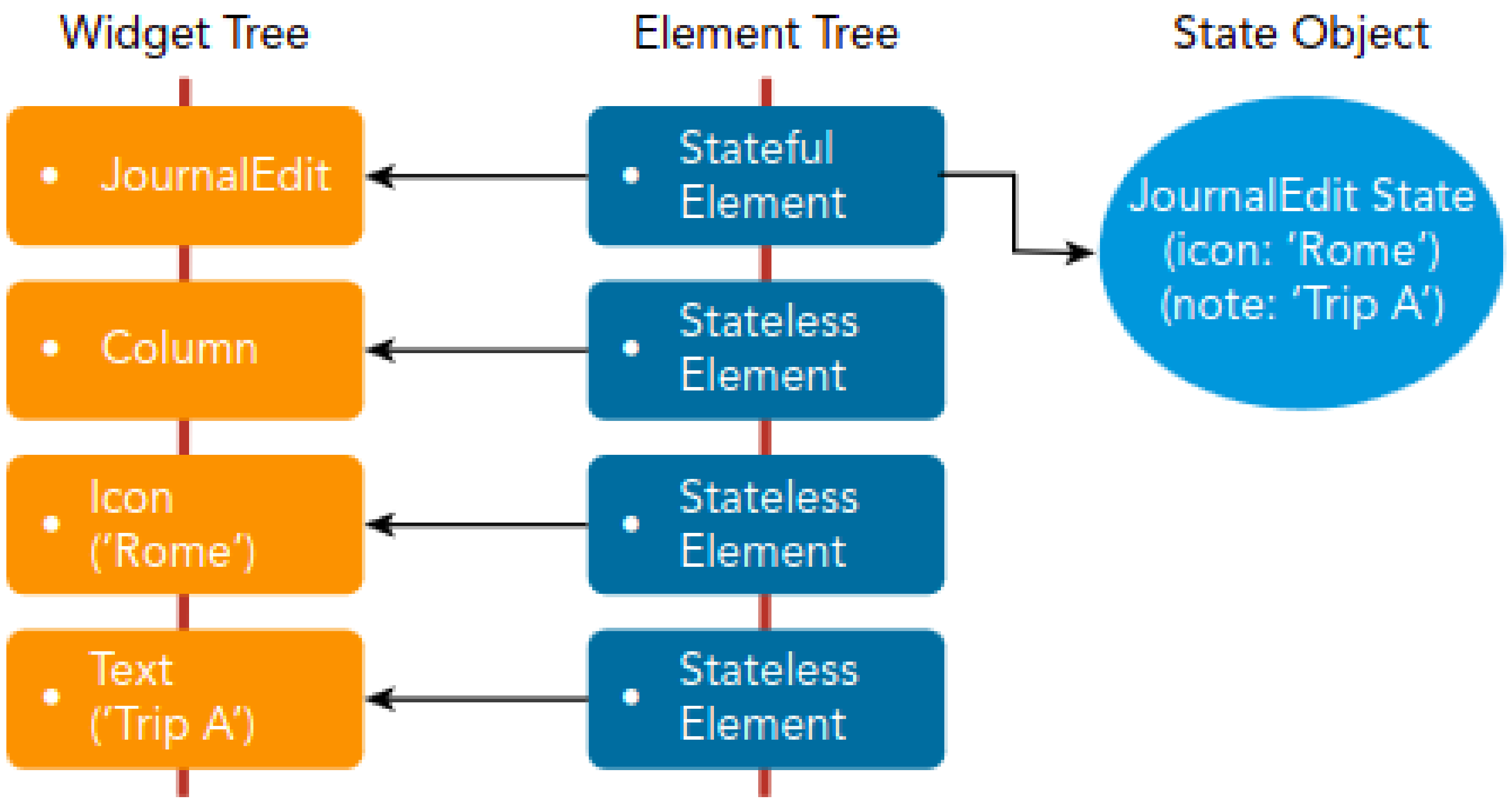
- **Stateless Widget:**
- Stateless Widgets are static widgets.
- They do not depend on any data change or any behavior change.
- Stateless Widgets do not have a state, they will be rendered once and will not update themselves, but will only be updated when external data changes.
- For Example: Text, Icon, RaisedButton are Stateless Widgets.

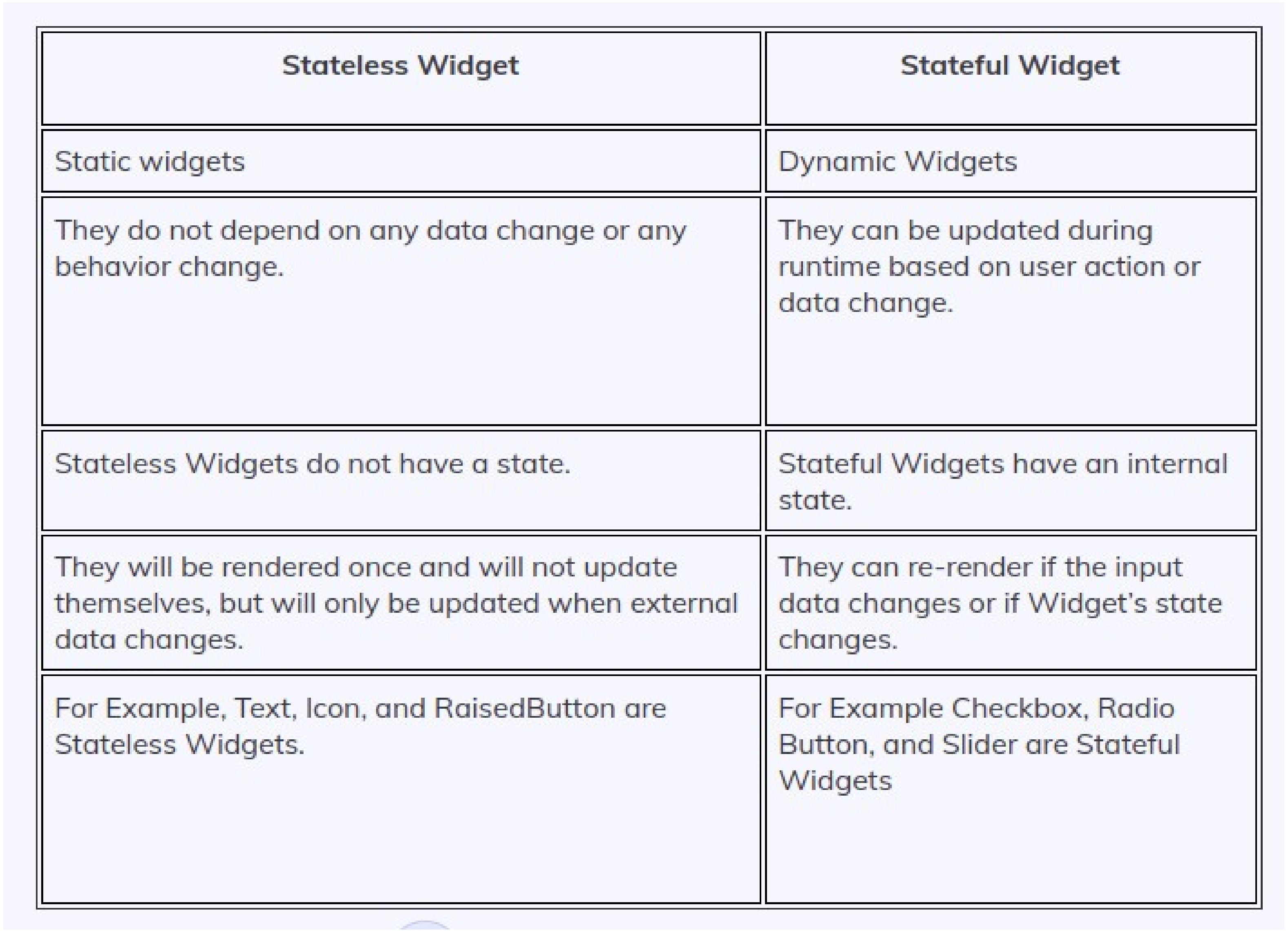
- **Stateful Widget:**
- Stateful Widgets are dynamic widgets.
- They can be updated during runtime based on user action or data change.
- Stateful Widgets have an internal state and can re-render if the input data changes or if Widget’s state changes.
- For Example: Checkbox, Radio Button, Slider are Stateful Widgets

# Widget Tree: Stateless



## Stateful


Widget Tree
Element Tree
State Object
JournalEdit
Stateful Element
JournalEdit State (icon: 'Rome') (note: 'Trip A')
Column
Stateless Element
Icon ('Rome')
Stateless Element
Text ('Trip A')
Stateless Element



| Stateless Widget | Stateful Widget |
|---|---|
| Static widgets | Dynamic Widgets |
| They do not depend on any data change or any behavior change. | They can be updated during runtime based on user action or data change. |
| Stateless Widgets do not have a state. | Stateful Widgets have an internal state. |
| They will be rendered once and will not update themselves, but will only be updated when external data changes. | They can re-render if the input data changes or if Widget's state changes. |
| For Example, Text, Icon, and RaisedButton are Stateless Widgets. | For Example Checkbox, Radio Button, and Slider are Stateful Widgets |