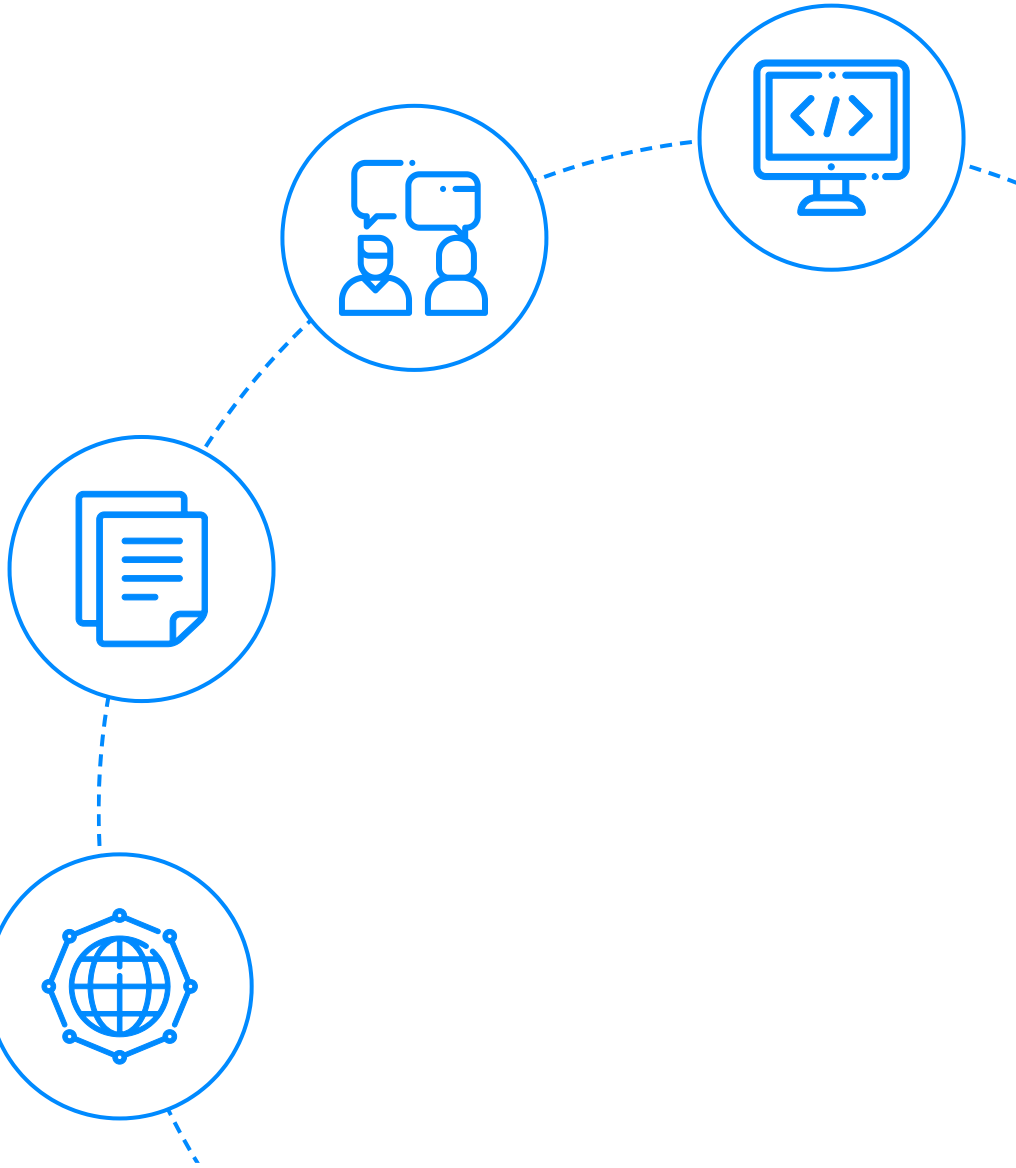
InterviewBit

Flutter Interview Questions



To view the live version of the page, click here.

Contents

Flutter Interview Questions for Freshers

1. Write the advantages of using flutter.
2. Explain the flutter architecture.
3. List some important features of flutter.
4. Write the limitations of flutter.
5. What are different build modes in flutter?
6. Explain the flutter widget and write its importance.
7. What are the types of widgets present in flutter?
8. What do you mean by Dart? Write its importance.
9. Explain App state.
10. Write the difference between runApp() and main() in flutter.
11. Explain packages and plugins in Flutter.
12. Name some best editors for flutter development.
13. Name some apps that mostly use flutter.
14. What do you mean by keys in flutter? When one should use it.
15. Explain Container class in a flutter.
16. Which one is better, either flutter or react native?
17. When to use mainAxisAlignment and crossAxisAlignment.
18. Is Flutter Open Source or not?
19. Why does a flutter app usually take a long developing time?
20. Explain Flutter Inspector.

Flutter Interview Questions for Freshers (.....Continued)

21. What is the use of Ticker in Flutter?
22. How would you execute code only in debug mode?
23. What is the use of Mixins?

Flutter Interview Questions for Experienced

24. What do you mean by Streams?
25. What are different types of Streams?
26. What do you mean by flutter SDK?
27. Write difference between Hot reload and Hot restart.
28. Explain BuildContext.
29. What do you mean by Widget testing?
30. What is state management?
31. Explain pubspec.yaml file.
32. What do you understand about tween animation?
33. How can we create HTTP requests in Flutter?
34. Name two database packages mostly used in Flutter.
35. Explain Flutter Provider.
36. What is await in Flutter? Write it's usage.
37. Write the difference between SizedBox Vs Container.
38. What do you mean by Null-aware operators?

Let's get Started

What is Flutter

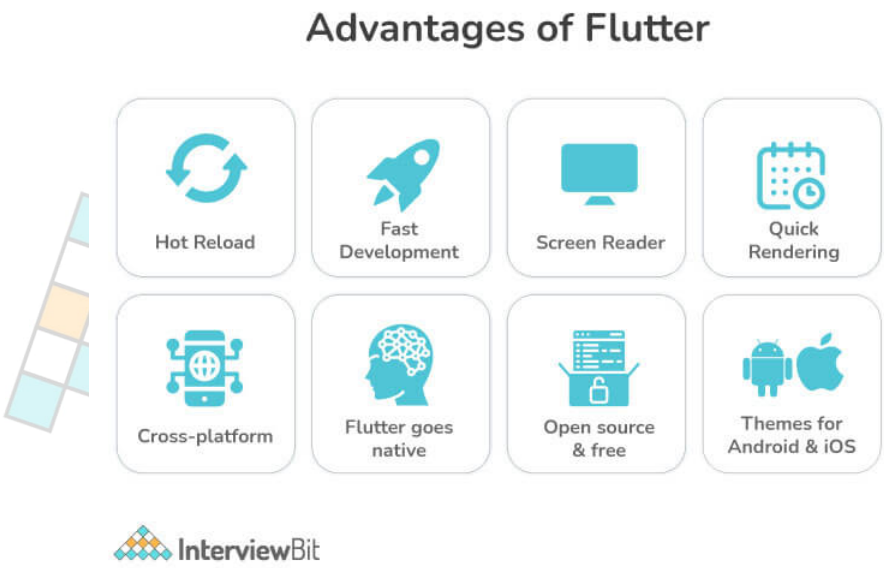
Originally developed by Google and now managed by ECMA, Flutter offers free and open-source UI development support. With one programming language and one codebase, users can create beautiful, natively compiled mobile applications with this UI toolkit. A framework like Flutter allows you to build mobile applications for iOS and Android in a truly platform-independent way. Developers consider it to be the fastest and most expressive way to build native apps. Due to its simplicity, high performance resulting from its development, rich user interface, Flutter will have a significant impact on the development of high-quality, feature-packed mobile applications in the near future.

Here, we have provided you with all the answers to these Flutter interview questions in the simplest way possible to assist you in acing your next job interview.

Flutter Interview Questions for Freshers

1. Write the advantages of using flutter.

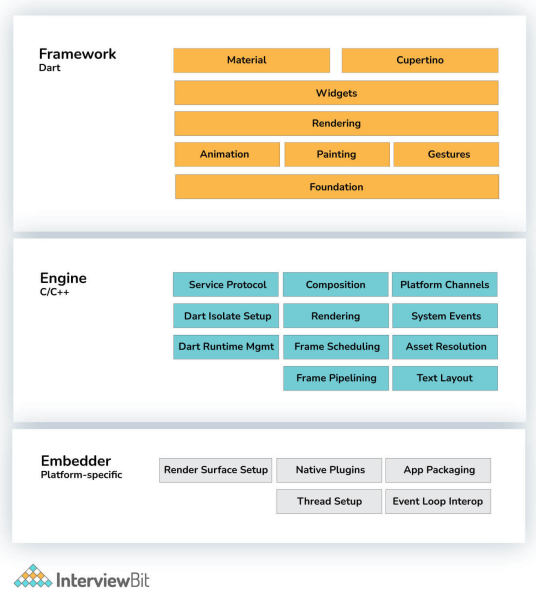
For developing mobile applications, Flutter usually fulfills the custom needs and requirements. It offers the following advantages:



- **Reduce Code Development**: Flutter's hot reload feature allows it to offer faster performance. With it, the application gets compiled using the arm C/C++ library, making it closer to machine code and enabling it to run more quickly. The Flutter team has put lots of effort into providing a wide variety of ready-to-use widgets. Most of them are incredibly customizable, saving your time like no other framework before.
- **Cross-platform Development**: Using Flutter, you can write code, manage, and run it across multiple platforms. For the developers, this saves time, money, and effort.
- **Live and Hot Reloading**: This makes the app development process simpler and faster. Additionally, it also allows us to modify or update the code once a change is made.
- **Similar to Native App performance**: In contrast to most cross-platform frameworks, Flutter does not rely on intermediate code representations or interpretations. The Flutter application is built directly into the machine code, which eliminates any performance issues associated with the interpretation process. With Flutter, you get a fully compiled release application ahead of time.
- **Good Community Support**: Developers can ask questions about issues and get answers quickly.
- **Little/Minimal Code**: Each Flutter app is built using Dart programming language, which uses JIT and AOT compilation for faster startup time, faster performance, and smoother functionality. With the JIT feature, you can increase the speed of development and refresh the UI.
- **Documentation:** Flutter's documentation is well-organized and informative. It serves as a central repository for all written documents.
- **Expressive and Flexible UI**: Flutter offers a customizable layered architecture that allows for highly customizable designs, expressive UIs, and fast rendering.

2. Explain the flutter architecture.

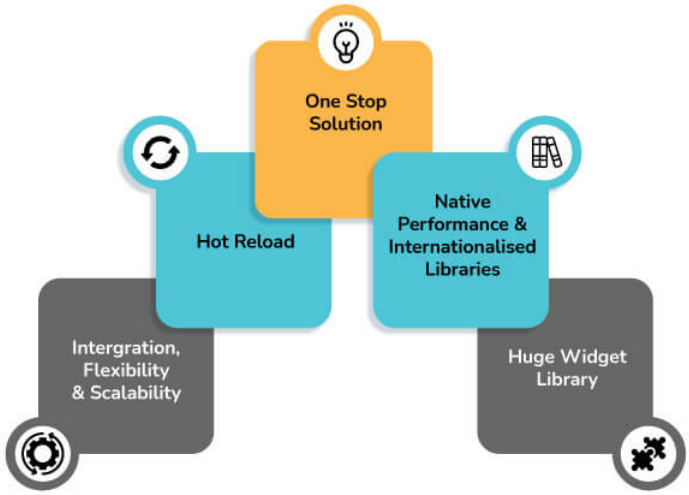
The structure of Flutter consists of three layers:



- **Upper layers**: The Dart-based platform that takes care of app widgets, gestures, animations, illustrations, and materials;
- **Flutter engine**: Handles the display and formatting of text.
- **Built-in service**: Used for the management of plugins, packages, and event loops.

3. List some important features of flutter.

Among the main features of flutter for developing mobile frameworks are:



- **Flexibility, scalability, and integration:** Flutter's easy-to-use and easy-to-integrate framework provides enhanced flexibility, scalability, and integration capabilities.
- **Hot Reload:** If the developer modifies the code, the changes can be seen immediately with Hot Reload. Thus, the changes are instantly visible within the app.
- **One-Stop Solution:** Flutter app development relies on a single framework and platform for the development, deployment, and management of changes, rather than using separate platforms and frameworks for different purposes.
- **Native Performance and Internationalized Flutter Libraries:** Flutter app development provides widgets customized for Android, iOS, and Google Fuchsia. Using widgets, you can integrate all the functionalities of the platform, such as scrolling, navigation, icons, and fonts.
- **Huge Widget Library:** It is because of Flutter's ready-to-use widget library that developers can develop apps faster when using the framework. In addition to a wide variety of widgets, it also includes animations with which you can spice up your application.

4. Write the limitations of flutter.

Flutter has the following limitations:

- **Third-party libraries are limited:** Since Flutter is relatively new, the number of third-party libraries is small. Additionally, some widgets in Flutter are only available on one platform.
- **Release size is larger:** Developers get frustrated when the release size is not as expected.
- **Requirements of Dart:** Dart is an Object-oriented programming language, but it cannot compete with Java, JavaScript, or C# since it is still relatively new. As a result, not many developers choose it.
- **Limited complexity:** Flutter's 3D modeling, Unity integration, and game engines fall short. Therefore, most ad mobile platforms also don't support it.
- **Lack of overall support:** Flutter is not so widely used yet. Even though it enjoys the attention of tech enthusiasts, it still lacks the continuous support that will come with time. Currently, the only support that Flutter receives comes from its community.

5. What are different build modes in flutter?

Depending on your development phase, the framework compiles your code in different ways or modes, which we called a build mode. Flutter's tooling allows the application to be compiled in three modes. Depending on our stage of development, we may choose between these compilation modes.

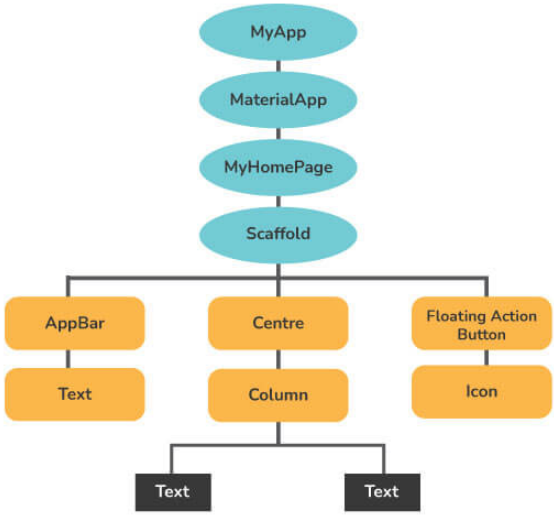
- **Debug Mode**: This mode enables debugging of apps on a physical device, emulator, or simulator. Assertions and service extensions are enabled here. Quick deployment is then achieved by optimizing compilation.
- **Profile Mode**: In this mode, some debugging abilities are maintained, enough to analyze the app's performance while testing. Tracing and some extensions are enabled in this case. On emulators and simulators, profile mode is disabled since their behavior does not reproduce real-world performance. The following command can be used to compile the profile mode: **flutter run --profile**
- **Release Mode**: When deploying the app, this mode is used to minimize the size of the footprint and maximize optimization. Debugging, assertions and service extensions are disabled here. Faster startup, faster execution, and less size are its key features. The following command can be used to compile the release mode: **flutter run --release**

6. Explain the flutter widget and write its importance.

Generally, a Flutter app consists of a number of widgets. A widget is a way to declare and construct user interfaces. In Flutter, you must write code inside a widget in order to build anything. With the widget, you can see how your app would appear with its current configuration. As soon as you modify the code, the widget will rebuild its description based on the difference between the old and new widget, and the changes will sync up with the UI of the app. The Flutter widget can be created as follows:

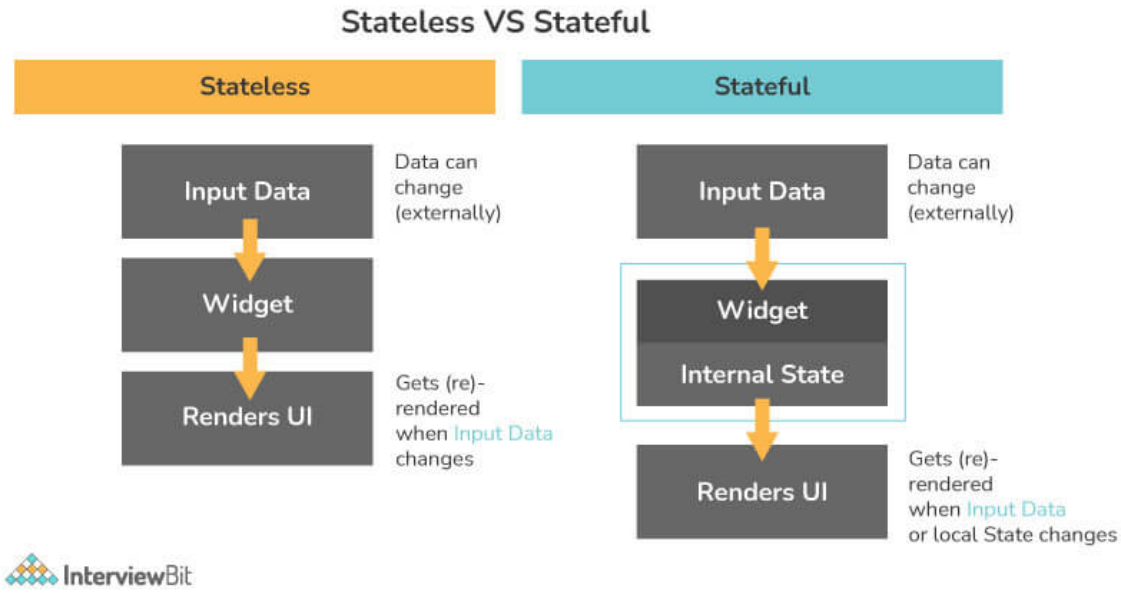
```
Class ImageWidget extends StatelessWidget {
    // Class Stuff
}
```

The app is built by nesting widgets within each other. This means the root of your app is a widget, and everything below it is a widget. Here's a simple image of what the widget tree looks like:



7. What are the types of widgets present in flutter?

In flutter, widgets can be divided into two categories:



- **Stateless Widget:** A widget that does nothing is a Stateless Widget. In essence, they are static and don't store any state. Thus, they don't save values that may change.
- **Stateful Widget:** A widget that does anything is a Stateful Widget. Stateful widgets are dynamic by nature, which means they can monitor changes and update the UI accordingly.

8. What do you mean by Dart? Write its importance.

Dart, a programming language developed by Google, is used to code Flutter apps as well as server and desktop applications. By using Dart, Flutter avoids the use of a separate declarative layout language such as JSX or XML. A simple Dart program is shown in the following example:

```
void main()
{
 for (int i = 0; i < 5; i++)
  {
        print('hello ${i + 1}');
  }
}
```

An overview of Dart's importance:

- Developers can read and visualize the layout of Dart very easily and effortlessly since it is declarative and programmatic.
- Unlike other programming languages, it supports the majority of the basic programming concepts like classes, interfaces, and functions.
- Arrays are not directly supported by Dart. Rather, it supports the collection that replicates the data structure like arrays, generics, and optional typing.
- Despite being similar to JavaScript, Dart runs code several times faster.
- For better performance and to reduce code execution time, the Dart virtual machine (VM) uses both Just-in-Time (JIT) and Ahead-of-Time (AOT) compilers.
- Dart is object-oriented programming, which makes it very scalable and stable for creating even complex applications.

9. Explain App state.

App State may also be referred to as a shared state or application state. It is possible to share app states across sections of your app and maintain user sessions in the same way. Here are some examples of App State:

- Login info
- User preferences
- E-commerce shopping cart
- Social networking notifications, etc.

10. Write the difference between runApp() and main() in flutter.

main(): This function starts the program. Flutter does not allow us to write any program without the main() function.

runApp(): Using runApp(), you are able to return the widgets that are connected to the screen as a root of the widget tree that will be rendered on the screen. This function is called in the main function, which is the driver of the app.

11. Explain packages and plugins in Flutter.

A package is a collection of classes, interfaces, and sub-packages that enable the creation of modular code that can be shared easily among users. Applications can be quickly developed using packages instead of developing everything from scratch. You can import new widgets or functionality into an app using a package in Flutter. There is a slight difference between plugins and packages as given below:

- **Plugins:** Using native code, enables more usability and makes it easier to use the device.
- **Packages:** These are new code or components written in the dart programming language.

Packages and plugins are often referred to as packages on DartPub, and specific distinctions between the two are made only during the creation of a new package.

12. Name some best editors for flutter development.

With the Flutter development tools, developers can make Flutter development faster and thus boost their productivity. In order to develop mobile applications, Flutter IDE and tools require some plugins. With these plugins, we can compile Dart, analyze code, and develop Flutter. Here are some popular IDEs for Flutter development:

- Android Studio
- Visual Studio
- IntelliJ IDEA
- Xcode
- Eclipse
- Emacs
- Vim, etc.

13. Name some apps that mostly use flutter.

Flutter is used today by many organizations for developing apps. The following are some of the most popular apps built on Flutter:

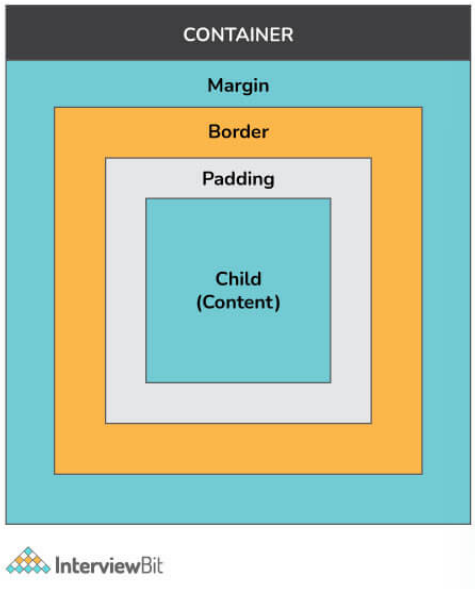
- Google Ads
- Reflectly
- Alibaba
- Birch Finance
- Coach Yourself
- Tencent
- Watermaniac, etc.

14. What do you mean by keys in flutter? When one should use it.

Keys are used in Flutter as identifiers for widgets, elements, and semantic nodes. GlobalKeys and LocalKeys are the subclasses of Key. Within the widget tree, keys are responsible for preserving the state of modified widgets. With keys, you can also reorganize and modify collections of widgets that have an equivalent type and defined state. The primary use of keys is to modify a widget tree that contains stateful widgets, not to modify a tree that is totally composed of stateless widgets.

15. Explain Container class in a flutter.

Basically, in Flutter, a container is a widget that has the capacity to accommodate multiple child widgets and manage them efficiently through dimensions, padding, and background color. Whenever we want to style the background of a widget, either because of a color, shape, or size constraint, we may use a container widget. With the Container class, widgets can be stored and positioned on the screen at our discretion. In general, it resembles a box for storing contents.
In the following image, you see how a basic container has padding, margin, and border properties surrounding its child widget:



16. Which one is better, either flutter or react native?

Today, thousands of mobile apps are being built with the two most popular cross-platform development frameworks i.e., React Native and Flutter. There are many similarities between React Native and Flutter including reloading quickly, excellent UI, awesome tooling, and capability to build native apps.

- **React Native:** This is an entirely JavaScript-based application using React. Facebook backed and open-sourced it in 2015.
- **Flutter:** It is written in the Dart programming language. In comparison to React Native, Flutter is a relatively new framework. Flutter was originally backed by another giant called Google.

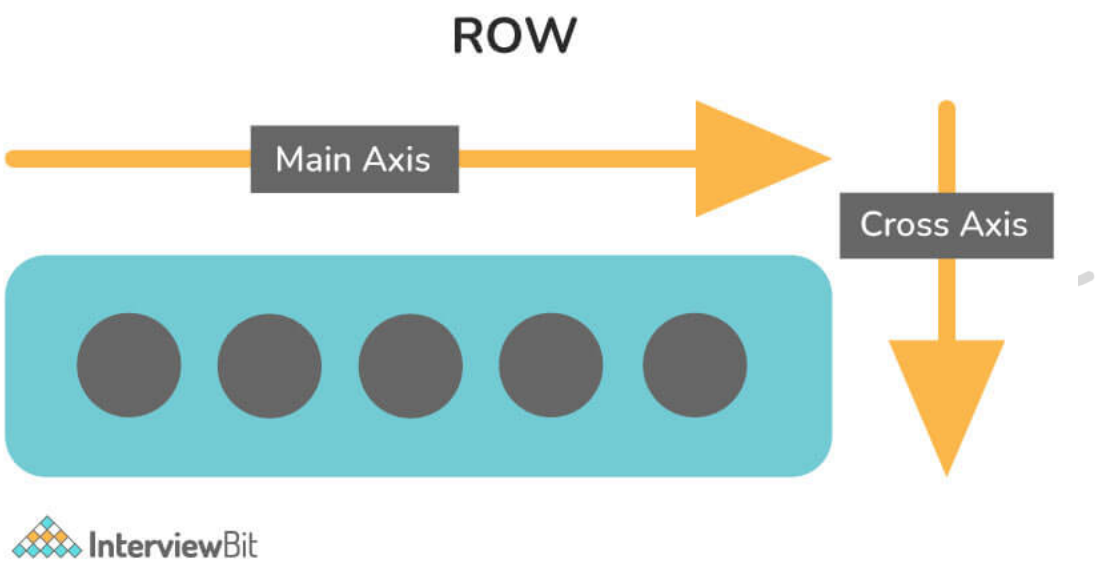
Choosing between them is difficult from the developer's perspective.

17. When to use mainAxisAlignment and crossAxisAlignment.

The mainAxisAlignment is how items are aligned on that axis, whereas crossAxisAlignment is how items are aligned on the other axis. Row and column widgets can align their children according to our preferences using the crossAxisAlignment and the mainAxisAlignment properties.

As Children of the Row Widget are arranged horizontally.
For Row:
mainAxisAlignment = Horizontal Axis
crossAxisAlignment = Vertical Axis

This can be better understood by looking at the image below:



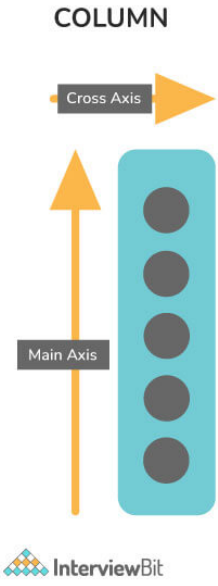
As Children of the Column Widget are arranged vertically.

For Column:

mainAxisAlignment = Vertical Axis

crossAxisAlignment = Horizontal Axis

This can be better understood by looking at the image below:



18. Is Flutter Open Source or not?

Yes, Flutter is a free and open-source UI software development kit from Google that allows developers to build cross-platform mobile apps with ease. Learn More.

19. Why does a flutter app usually take a long developing time?

The first time you build a Flutter application, it takes much longer than usual since Flutter creates a device-specific IPA or APK file. Xcode and Gradle are used in this process to build a file, which usually takes a lot of time.

20. Explain Flutter Inspector.

In the same manner, as with Native Android, the XML file allows us to view our app's blueprint and properties. There is a powerful tool called Flutter Inspector for Flutter applications that allows you to visualize the blueprint of your widgets and their properties. Using it, you can diagnose various layout issues and understand the current layout.
Flutter Inspector offers the following benefits:

- Select widget mode
- Toggle platform
- Show paint baselines
- Show debug paint
- Refresh widget
- Enable slow animations
- Show/hide performance overlay

21. What is the use of Ticker in Flutter?

We use a ticker to tell how often our animation is refreshed in Flutter. Signals are sent at a constant frequency, such as 60 times per second, using this type of signal-sending class. We understand it better with our watch, which ticks constantly. For each tick, a callback method is provided that has the time since the first tick at each second since it was started. The tickers are synchronized immediately, even if they begin at different times.

22. How would you execute code only in debug mode?

We first need to import the dart foundation in order to run the code only in debug mode:

```
import 'package:flutter/foundation.dart' as Foundation;

The next step is to use kReleaseMode as follows:

if (Foundation.kReleaseMode){     // is Release Mode??
   print('release mode');
} else {
             print('debug mode');
}
```

23. What is the use of Mixins?

Multiple inheritances are not supported by Dart. Thus, we need mixins to implement multiple inheritances in Flutter/Dart. The use of mixins makes it easy to write reusable class code in multiple class hierarchy levels. Mixins can also be used to provide some utility functions (such as RenderSliverHelpers in Flutter).

Flutter Interview Questions for Experienced

24. What do you mean by Streams?

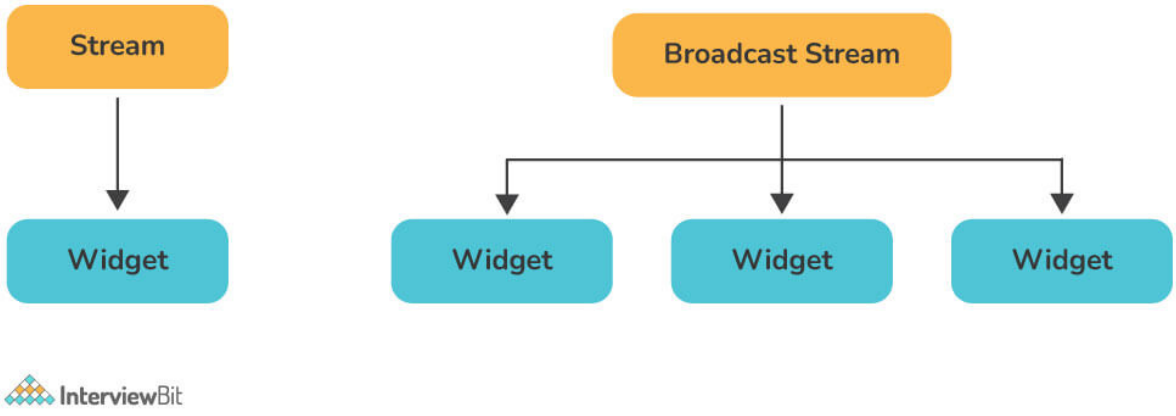
In asynchronous programming, streams are used to provide a sequence of data in an asynchronous manner. Similar to a pipe, we put a value on one end and a listener receives it on the other. Several listeners can be put into one stream, and they'll all get the same value when they're put in the pipeline. It's possible to create and manage streams through the SteamController.

The Stream API provides the await for and listen() methods for processing streams. Streams can be created in many ways, but they can only be used in the same manner. Here is an example:

```
Future<int> sumStream(Stream<int> stream) async {
  var sum = 0;
   await for (var value in stream) {
     sum += value;
   }
   return sum;
}
```

25. What are different types of Streams?

The streams' functionality is part of Dart and is inherited by Flutter. In Flutter, there are two kinds of streams:



- **Single Subscription Streams**: These streams deliver events sequentially. They are considered as sequences contained within a larger whole. These streams are used when the order in which events are received matters, such as reading a file. There can be only one listener throughout the sequence, and without a listener, the event won't be triggered.
- **Broadcast Streams**: These streams deliver events to their subscribers. Upon subscribing to events, subscribers are immediately able to start listening to them. These are versatile streams that allow several listeners to listen simultaneously. Furthermore, one can listen again even after canceling a previous subscription.

26. What do you mean by flutter SDK?

A Flutter SDK (Software Development Kit) enables developers to build applications for mobile, web, and desktop using a single codebase. Flutter SDK includes the following features:

- Dart SDK
- Contains a rendering engine, widgets, APIs for testing and integration, etc.
- Compilation tools for Native Machine Code (code for iOS and Android).
- React-style modern framework
- Provide Interop and plugin APIs to connect with system and 3rd-party SDKs.
- A headless test runner that runs tests on Windows, Linux, and Mac.
- Use the Dart DevTools to test, debug, and profile your app. Use
- Flutter and Dart command-line tools to develop, build, test and compile your apps across platforms.

27. Write difference between Hot reload and Hot restart.

For any dart application, the initial execution requires a fair amount of time. Therefore, to solve this problem, flutter has two features: Hot Reload and Hot Restart, which reduce the execution time of our app after we run it.

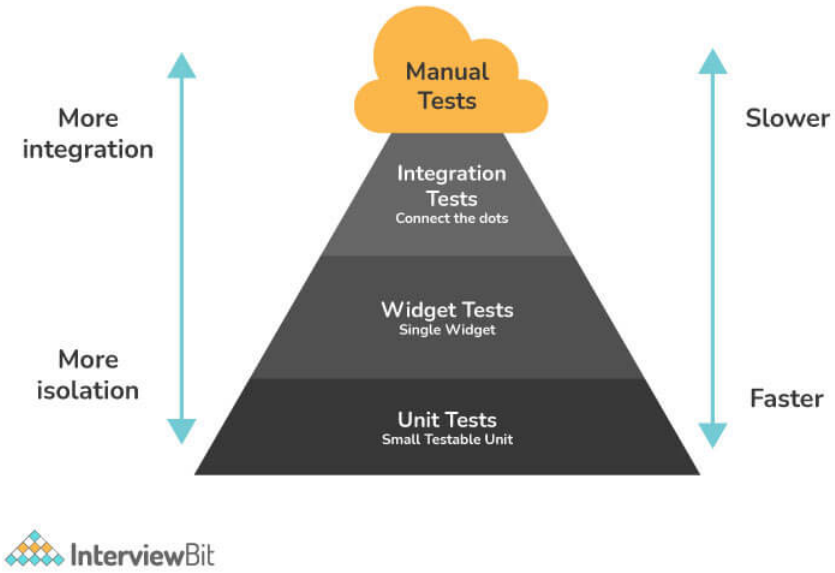
- **Hot Reload**: It is considered an excellent feature of flutter that takes approximately one second to perform its functionality. With this function, you can make changes, fix bugs, create UIs, and add features easily and quickly. By utilizing the hot reload feature, we can quickly compile the new code in a file and send it to Dart Virtual Machine (DVM). As soon as DVM completes the update, it updates the app's UI. The preserved state is not destroyed in hot reload.
- **Hot Restart**: It has a slightly different functionality as compared to a hot reload. In this, the preserved states of our app are destroyed, and the code gets compiled again from the beginning. Although it takes longer than a hot reload, it's faster than a full restart function.

28. Explain BuildContext.

BuildContexts are used to identify or locate widgets in widget trees. Each widget has its own BuildContext, i.e., one BuildContext per widget. Basically, we're using it to find references to other widgets and themes. In addition, you can utilize it to interact with widget parents and access widget data.

29. What do you mean by Widget testing?

Flutter supports three types of tests:



- **Unit tests:** Using unit testing, you can test a class or method. Unit tests do not check for rendering to screen, interacting with external services, or user interactions.
- **Widget tests:** Using widget testing, you can test a single widget. This ensures that the widget's UI looks as expected and responds appropriately to events. In other words, it ensures that the widget design, rendering, and interaction with other widgets are up to the mark.
- **Integration tests:** Using Integration testing, you can test the critical flows of the entire app. It is important to check whether all widgets and services work together as expected. You can also use it to measure and benchmark the performance of your app.

30. What is state management?

Whether you are building a mobile app or a web application, State Management is crucial. Using it, states of various UI controls are centralized to handle data flow across an application. It can be a text field, radio button, checkbox, dropdown, toggle, form, and so on. In Flutter, state management can be categorized into two types as follows:

- **Ephemeral State:** Ephemeral state is also called UI state or local state, and it pertains to a particular widget. In other words, it is a state that is contained within the specific widget. By means of StatefulWidget, Flutter provides support for this state.
- **App State:** This is different from the ephemeral state since it is a state that we intend to share across different parts of the app and which we want to maintain between sessions. These types of states can thus be used globally. By means of scoped_model, Flutter provides support for this state.

The following diagram gives a better explanation of the differences between ephemeral and app states:

The pubspec.yaml file, also known as 'pubspec', is a file that is included when you create a Flutter project and is located at the top of the project tree. This file contains information about the dependencies like packages and their versions, fonts, etc., that a project requires. It makes sure that the next time you build the project, you will get the same package version. Additionally, you can set constraints for the app. During working with the Flutter project, this configuration file of the project will be required a lot. This specification is written in YAML, which can be read by humans.

The following are included in this file:

- General project settings, like name of the project, version, description, etc.
- Dependencies within a project.
- The assets of the project (e.g., images, audio, etc.).

32. What do you understand about tween animation?

The shortened version of in-between animation is **tween animation**. The start and endpoints of an animation must be specified in tween animation. Using this method, the animation can begin at the beginning and can progress through a series of values until it reaches the endpoint. Transition speed and duration are also determined by using the tween animation. Calculating the transition from the beginning to the end will be easier with the widget framework.

33. How can we create HTTP requests in Flutter?

To create HTTP requests, use the HTTP package (import 'package:http/http.dart' as http;). In the following manner, we can make the Requests:

```
http.get('https://jsonplaceholder.typicode.com/albums/1');
```

It will return a Future <http.Response>.

34. Name two database packages mostly used in Flutter.

As far as Flutter is concerned, the following database packages are widely accepted and mostly used:

Firebase database: It gives users access to and control over the cloud database. Firebase basically provides a NoSQL database for Flutter apps with the ability to manage data retrieval and storage through JSON protocol. Data sync and quick loading make it one of the most suitable options for Flutter Apps.
Features:

- NoSQL DB
- APIs (REST only)
- Authentication
- Analytics
- Storage

SQFlite database: Users can access and modify the SQLite database using this. With this database, you have full control over your database, queries, relationships, and anything you could desire.
Features:

- Serverless
- Zero configuration
- Open-Source
- Compact
- Single DB file

35. Explain Flutter Provider.

The provider is built using widgets. You can use all the objects in the provider as if they were just part of Flutter with the new widget subclasses it creates. This also means that the provider is not cross-platform. The provider is the simplest way to handle state management. Basically, it works on the concept of PUB-SUB i.e., there is one provider and several subscribers.

36. What is await in Flutter? Write it's usage.

Until the async method is finished, await interrupts the process flow. Await generally means: Wait here until this function is finished so that you can get its return value. Await can only be used with async. Using this, all currently running functions are put on hold until PF nature is complete.

37. Write the difference between SizedBox Vs Container.

- **Container:** In this parent widget, multiple child widgets can be easily controlled and handled by adjusting their size, padding, and color efficiently. We can wrap a widget in a container widget if it needs any styling, like a color, a shape, or a size constraint, etc.
- **SizedBox:** This is a specific size box. It does not allow us to set the widget's color or decoration, unlike Container. In this case, we only need to resize the widget that is passed as a child. In other words, it forces its child widget to have a specific size.

38. What do you mean by Null-aware operators?

Null-aware operators in dart allow you to make computations based on whether or not a value is null. Dart provides some useful information to handle the null values.

- The "??=" assignment operator: It assigns a value to a variable only if it is null.

```
int a;                          // a is initialized with null value.
a ??= 10;
print(a);                      // It will print 10.
```

- The "??" null-aware operator: This one computes and returns the value between two expressions. In the first step, expression 1 is checked for nullness, and if it is, its value is returned; otherwise, expression 2 is evaluated, and its value is returned.

```
print(5 ?? 10);            // It will print 5.
print(null ?? 10);       // It will print 10.
```

- The “?.” Safe Navigation Operator: It is also known as the Elvis operator. It is possible to use the ?. operator when calling a method/getter on an object, as long as the object isn't null (otherwise, the method will return null).

```
obj?.child?.child?.getter       //The expression returns null if obj, child1, or child2
```

Conclusion:

Flutter is a mobile technology that is among the most innovative and booming in the mobile market as the next revolutionary thing right now. Although it is a relatively new framework for developing cross-platform applications, its popularity is soaring due to which there is an increase in demand for Flutter developers.

Links to More Interview Questions

C Interview Questions	Php Interview Questions	C Sharp Interview Questions
Web Api Interview Questions	Hibernate Interview Questions	Node Js Interview Questions
Cpp Interview Questions	Oops Interview Questions	Devops Interview Questions
Machine Learning Interview Questions	Docker Interview Questions	Mysql Interview Questions
Css Interview Questions	Laravel Interview Questions	Asp Net Interview Questions
Django Interview Questions	Dot Net Interview Questions	Kubernetes Interview Questions
Operating System Interview Questions	React Native Interview Questions	Aws Interview Questions
Git Interview Questions	Java 8 Interview Questions	Mongodb Interview Questions
Dbms Interview Questions	Spring Boot Interview Questions	Power Bi Interview Questions
Pl Sql Interview Questions	Tableau Interview Questions	Linux Interview Questions
Ansible Interview Questions	Java Interview Questions	Jenkins Interview Questions