FLUTTER INTERVIEW QUESTIONS

Made with Love by Sumit Ojha.
For more info visit: http://www.sumitsojha8.co/

1) What is Flutter?

Ans: Flutter is a UI toolkit for creating fast, beautiful, natively compiled mobile applications with one programming language and a single codebase. It is an open-source development framework developed by Google. Generally, Flutter is not a language; it is an SDK. Flutter apps use Dart programming language for creating an app. The first alpha version of Flutter was released in May 2017.

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.

2) What is Dart and why does Flutter use it?

Dart is a general-purpose, object-oriented programming language with C-style syntax. It is open-source and developed by Google in 2011.

Dart was chosen because the language of Flutter for the subsequent reason:

- Dart is **AOT (Ahead Of Time)** compiled to fast, predictable, native code, which allows most of Flutter to be written in Dart.
- This not only makes Flutter fast, virtually everything (including all the widgets) are often customized.
- Dart also can be **JIT (Just In Time)** compiled for exceptionally fast development cycles and game-changing workflow (including Flutter's popular sub-second stateful hot reload).
- Dart allows Flutter to avoid the necessity for a separate declarative layout language like JSX or XML, or separate visual interface builders, because Dart's declarative,

3) Name a number of the favored apps that use flutter?

Many organizations across the planet are building apps using flutter. this is often thanks to its ability to create native apps in varied categories. Therefore, the foremost popular apps built with flutter are;

- Firstly, Google Ads for utility bills.
- Alibaba for commerce
- Tencent

4) What are the benefits of Flutter?

- Learn Once write Everywhere
- Cross-platform Development
- Faster Development
- Good Community
- Live and Hot Reloading
- Native Performance
- Provides Native Look and Feel
- Expressive and versatile UI

5) What are the Flutter widgets?

A Flutter app is always considered as a tree of widgets. Whenever you are going to code for building anything in Flutter, it will be inside a widget. Widgets describe how your app view should look like with their current configuration and state. When you made any alteration in the code, the widget rebuilt its description by calculating the difference of previous and current widget to determine the minimal changes for rendering in the app's UI.

Widgets are nested with each other to build the app. It means your app's root is itself a widget, and all the way down is a widget also. For example, a widget can display something, can define design, can handle interaction, etc.

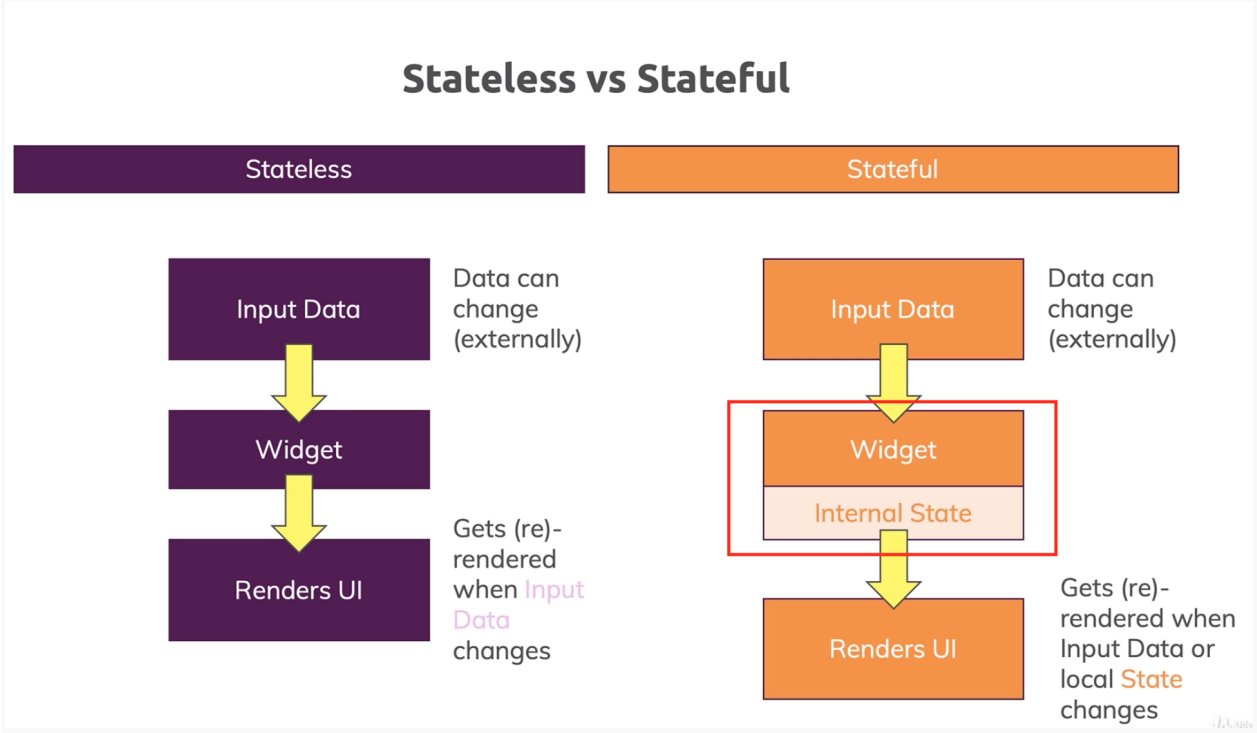
6) Differentiate StatelessWidget and StatefulWidget?

Ans :

- **Stateless:** Widget state creates just one occasion , then it can update values but not state explicitly.
- **StatelessWidget:** A widget that does not require a mutable state. **StatefulWidget:** A widget that has a mutable state.
- That’s why it's just one class which extends with **StatelessWidget**. they will neve
- r rerun the build() method again.
- **Stateful:** Widgets can update their STATE (locally) & values multiple times upon event triggered. That’s the rationale ,
- The implementation is additionally different. In this, we've 2 classes, one is StatefulWidget & the opposite is its State implementation handler i.e. State. So if I say, they will re-run the build() method again & again to support events triggered.
- A **StatelessWidget** will never rebuild by itself (but can from external events). A StatefulWidget can.
- **A StatelessWidget is static whereas a StatefulWidget is dynamic.**

Examples of Stateful widgets are: Slider, Checkbox, Form, Radio, InkWell, and textFiled.

Examples of Stateless widgets are: Row, Text, Container, Columns, etc.



7) What is pubspec.yaml file?

It is the project's configuration file that will use a lot while working with the Flutter project. It allows you how your application works. It also allows us to set the constraints for the app. This file contains:

- Project general settings such as name, description, and version of the project.
- Project dependencies.
- Project assets (e.g., images, audio, etc.).

8) What are the advantages of Flutter?

The popular advantages of the Flutter framework are as follows:

- **Cross-platform Development:** 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.
- **Faster Development:** The performance of the Flutter application is fast. Flutter compiles the application by using the arm C/C++ library that makes it closer to machine code and gives the app a better native performance.
- **Good Community:** Flutter has good community support where the developers can ask the issues and get the result quickly.
- **Live and Hot Reloading:** It makes the app development process extremely fast. This feature allows us to change or update the code as soon as the alterations are made.
- **Minimal code:** Flutter app is developed by Dart programming language, which uses JIT and AOT compilation to improve the overall start-up time, functioning and accelerates the performance. JIT enhances the development system and refreshes the UI without putting extra effort into building a new one.
- **UI Focused:** It has an excellent user interface because it uses a design-centric widget, high-development tools, advanced APIs, and many more features.
- **Documentation:** Flutter has very good documentation support. It is organized and more informative. We can get everything that we want to be written in one place.

9) Why is the Android and iOS folder in the Flutter project?

Android: This folder holds a complete Android project. It is used when you create the Flutter application for Android. When the Flutter code is compiled into the native code, it will get injected into this Android project, so that the result is a native Android application.

For Example: When you are using the Android emulator, this Android project is used to build the Android app, which is further deployed to the Android Virtual Device.

iOS: This folder holds a complete Mac project. It is used when you build the Flutter application for iOS. It is similar to the Android folder, which is used when developing an app for Android. When the Flutter code is compiled into the native code, it will get injected into this iOS project, so that the result is a native iOS application. Building a Flutter application for iOS is only possible when you are working on macOS and Xcode IDE.

10) What is Tween Animation?

It is the short form of in-betweening. In a tween animation, it is required to define the start and endpoint of animation. It means the animation begins with the start value, then goes through a series of intermediate values and finally reached the end value. It also provides the timeline and curve, which defines the time and speed of the transition. The widget framework provides a calculation of how to transition from the start and endpoint.

11) Explain Hot Reload in Flutter?

The hot reload feature allows you to quickly and easily perform an experiment in the project. It helps to build a UI, add new features, fix bugs, and make app development fast. To perform hot reloading of a Flutter app, do the following steps:

- Run the app in a supported Flutter editor or terminal window.
- Modify any of the Dart files in the project.
- If you use an IDE that supports Flutter, then select Save All or click the Hot Reload button on the toolbar. Immediately, you can see the result in your emulator or real device.

12) Name the popular database package used in the Flutter?

The most used and popular database packages used in the Flutter are as follows:

- **sqlite database:** It allows access and manipulation of SQLite database.
- **Firebase database:** It will enable you to access and manipulate the cloud database.

13) What is the difference between Hot Restart and Hot Reload?

The following are the essential differences between Hot Restart and Hot Reload:

Hot Reload	Hot Restart

It works with a small r key on the terminal or commands prompt.	It mainly works with States value.
The hot reload feature allows us to quickly compile the newly added code in the file and sent them to Dart Virtual Machine (DVM). After DVM completes the updation, it immediately updates the UI of the app.	It allows developers to get a fully compiled application because it destroys the preserves State values and sets them to their defaults. On every Hot Restart, our app widget tree is completely rebuilt with the new typed code.
It helps to build UI, add new features, fix bugs, and make app development fast.	It takes more time than Hot Reload to compile and update the app.

14) What is the difference between "main()" and "runApp()" functions in Flutter?

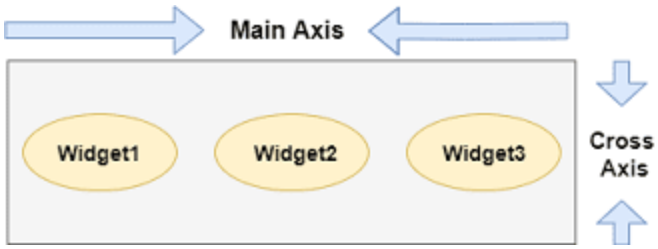
We can differentiate the main and runApp functions in Flutter as below:

- **The main() function** is responsible for starting the program. Without the main() function, we cannot write any program on Flutter.
- **The runApp() function** is responsible for returning the widgets that are attached to the screen as a root of the widget tree and will be rendered on the screen.

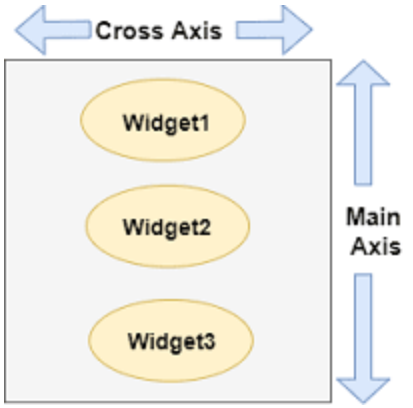
13) When should you use mainAxisAlignment and crossAxisAlignment?

We can use the **crossAxisAlignment** and **mainAxisAlignment** to control how a row and column widgets align its children based on our choice.

The row's cross-axis will run vertically, and the main axis will run horizontally. See the below visual representation to understand it more clearly.



The column's cross-axis will run horizontally, and the main axis will run vertically. The below visual representation explains it more clearly.



14) What is the difference between SizedBox VS Container?

The Container in Flutter is a parent widget that can contain multiple child widgets and manage them efficiently through width, height, padding, background color, etc. If we have a widget that needs some background styling may be a color, shape, or size constraints, we may wrap it in a container widget.

The SizedBox widget in Flutter is a box that comes with a specified size. Unlike Container, it does not allows us to set color or decoration for the widget. We can only use it for sizing the widget passed as a child. It means it forces its child widget to have a specific width or height.

15) What is Stream in Flutter?

A stream is a sequence of asynchronous events. It provides an asynchronous sequence of data. It is the same as a pipe where we put some value on the one end, and if we have a listener on the other end, it will receive that value. We can keep multiple listeners in a stream, and all of those will receive the same value when put in the pipeline.

We can process a stream by using the await for or listen() from the Stream API. It has a way to respond to errors. We can create streams in many ways, but they can be used in the same way. See the below example:

```
1.  Future<int> sumStream(Stream<int> stream) async {
2.      var sum = 0;
3.      await for (var value in stream) {
4.        sum = sum + value;
5.      }
6.      return sum;
7.    }
```

16) Explain the different types of Streams?

Streams can be of two types, which are:

Single subscription streams

It is the most common type of stream that contains a sequence of events, which is the parts of a larger whole. It will deliver the events in the correct order and without missing any of them. If any of the events are missing, then the rest of the stream makes no sense. This stream is mainly used to read a file or receive a web request. It will listen once, and if it is listening again, it means missing an initial event. When it starts listening, the data will be fetched and provided in chunks.

Broadcast streams

It is a type of stream used for individual messages that can be handled one at a time without the knowledge of the previous events. It can have multiple listeners to listen simultaneously, and we can listen again after canceling the previous subscription. This mouse events in a browser is a kind of this stream.

---

17) Why is the build() method on State and not StatefulWidgets?

The main reason behind this is that the StatefulWidget uses a separate State class without building a method inside its body. It means all fields inside a Widget are immutable and includes all its sub-classes.

On the other hand, the **StatelessWidge**t has its build and associated methods inside its body. It is due to the nature of StatelessWidget, which is rendered completely on the screen using the provided info. It also doesn't allow any future changes in its State information.

The **StatefulWidget** allows us to change the State information during the course of the app. Therefore, it is not suitable for storage in a build method to satisfy Widget class conditions where all fields are immutable. This is the main reason to introduce the State class. Here, we only need to override the createState() function to attach the defined State with the StatefulWidget, and then all expected changes happen in a separate class.

18) What are the different build modes in Flutter?

The Flutter tooling supports three modes while compiling the application. These compilation modes can be chosen by depending on where we are in the development cycle. The name of the modes are:

- Debug
- Profile
- Release

19) What are keys in Flutter, and when to use it?

- Keys in Flutter are used as an identifier for Widgets, Elements and SemanticsNodes. We can use it when a new widget tries to update an existing

element; then, its key should be the same as the current widget key associated with the element.

- Keys should not be different amongst the Elements within the same parent.
- The subclasses of Key must be a GlobalKey or LocalKey.
- Keys are useful when we try to manipulate (such as adding, removing, or reordering) a collection of widgets of the same type that hold some state.

20) What is profile mode, and when do you use it?

Profile mode is used to measure the performance of our applications. In this mode, some debugging ability is maintained to profile your app's performance. This mode is disabled on the emulator and simulator because they are not representative of real performance.

We can use the below command to compile the profile mode:

1. flutter run --profile

21) What is release mode, and when do you use it?

Release mode allows us to optimize the codes and generate them without any debug data in a fully optimized form. In this mode, many of the application's code will be entirely removed or rewritten.

We use this mode when we are ready to release the app. It enables maximum optimization and minimal footprint size of the application.

We can use the below command to compile the release mode:

1. flutter run --release

22) What is the difference between WidgetsApp and MaterialApp?

The below comparison chart explains the basic differences between WidgesApp and MaterialApp:

WidgetsApp	MaterialApp
WidgetsApp is used for basic navigation. It includes many foundational widgets together with the widgets library that Flutter uses to create the UI of our app.	MaterialApp, along with the material library, is a layer that is built on the top of WidgetsApp and its library. It implements Material Design that provides a unified look and feels to our app on any platform.
WidgetsApp class is the base class for MaterialApp class.	It offers many interesting tools such as Navigator or Theme for developing the application.

It wraps several widgets that are required for building the application.	It wraps several widgets that are required for building material design applications.

23) What is BuildContext?

BuildContext in Flutter is the part of the widgets in the Element tree so that each widget has its own BuildContext. We mainly use it to get a reference to another widget or theme. For example, if we want to use a material design element, it is required to reference it to the scaffold. We can get it using the Scaffold.of(context) method.

24) What types of tests can you perform in Flutter?

Testing is an activity used to verify and validate the application, which is bug-free and meets the user requirements. Generally, we can use these three types of tests in Flutter:

Unit Tests: It tests a single function, method, or class. Its goal is to ensure the correctness of code under a variety of conditions. This testing is used for checking the validity of our business logic.

Widget Tests: It tests a single widget. Its goal is to ensure that the widget's UI looks and interacts with other widgets as expected.

Integration Tests: It validates a complete app or a large part of the app. Its goal is to ensure that all the widgets and services work together as expected.

Flutter also provides one additional testing known as a golden test. Its goal is to ensure that you have an image of a widget or screen and check to see whether the actual widget matches it or not.

25) Which one is better between Flutter and React Native?

Flutter and **React Native** are both used to develop the native hybrid app from a single codebase. These apps can run on iOS and Android platforms.

React Native develop by Facebook, whereas the Flutter framework was first introduced by Google. So, both framework has a very good feature and community.

Flutter uses Dart language to create applications, whereas **React Native** uses JavaScript to build the applications.

From the developer's point of view, it is very difficult to choose amongst them. Thus, it is very challenging to choose a winner between Flutter and React Native.