1 - 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.

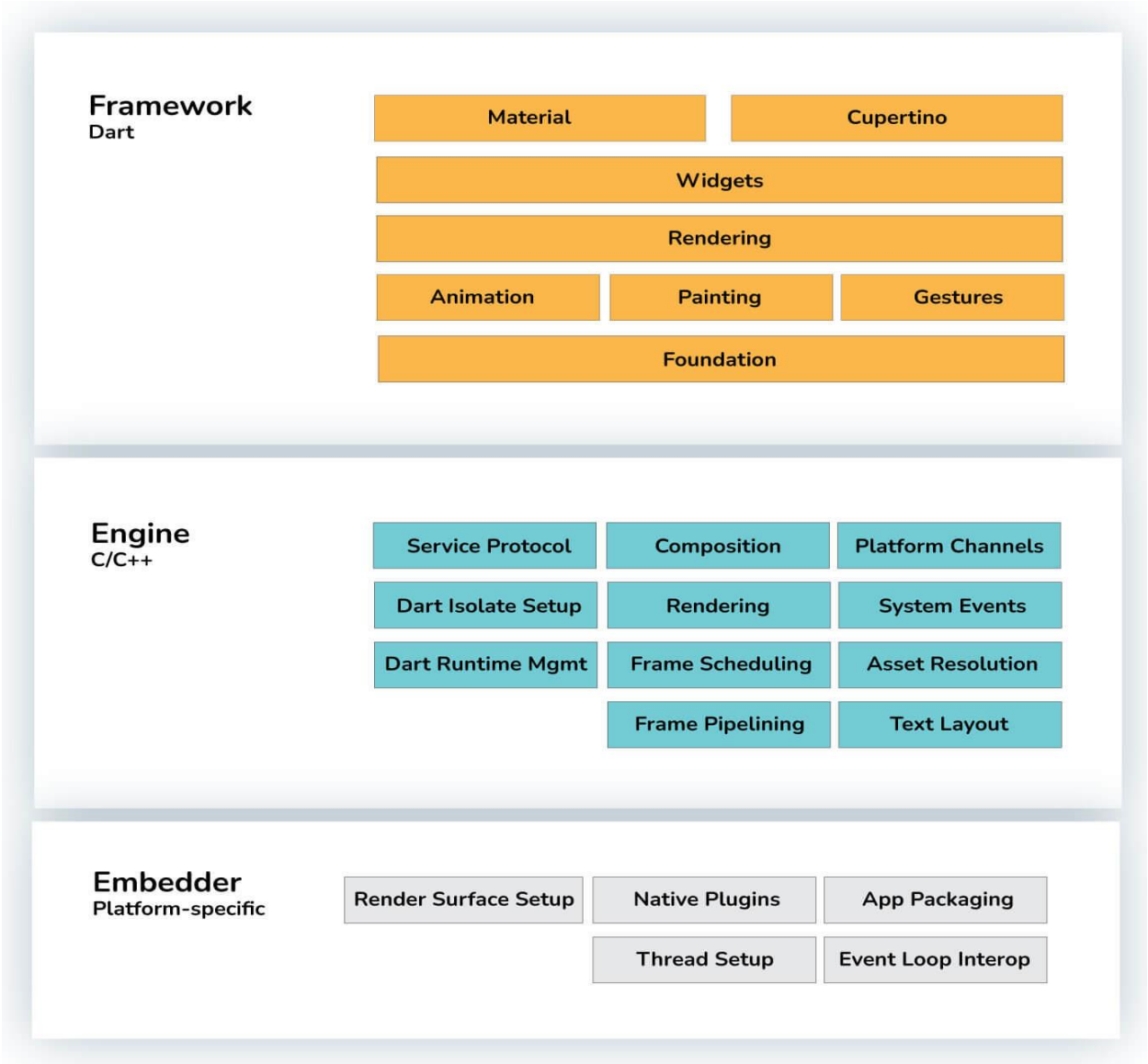
2 - Write the advantages of using flutter?

- **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.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

3- Explain the flutter architecture

- **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.



InterviewBit

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

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?

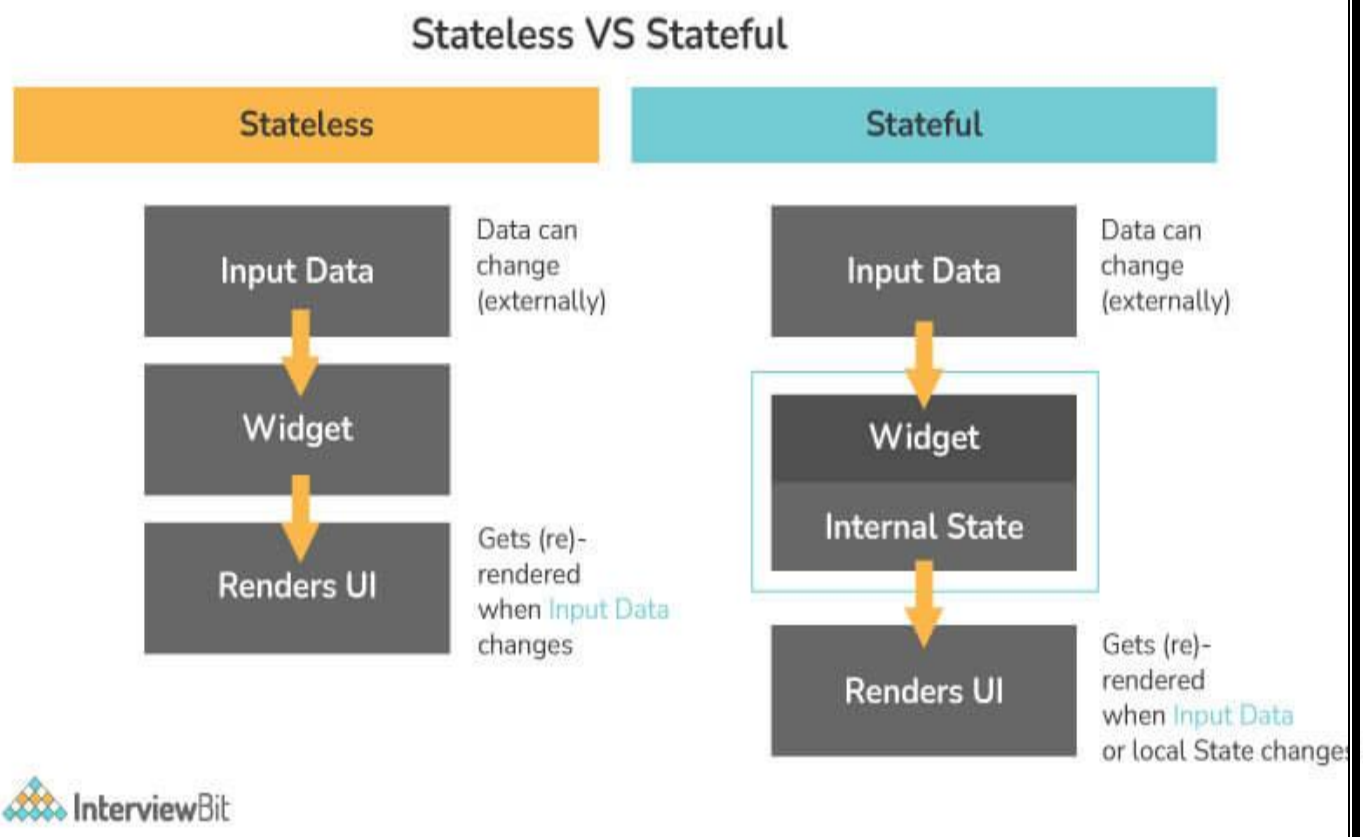
Depending on your development phase, the framework compiles your code in different ways or modes, which we called a build mode. Flutter's tooling allows the application to be compiled in three modes. Depending on our stage of development, we may choose between these compilation modes.

- **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**

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

6 - 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.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

7 - 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.

An overview of Dart's importance:

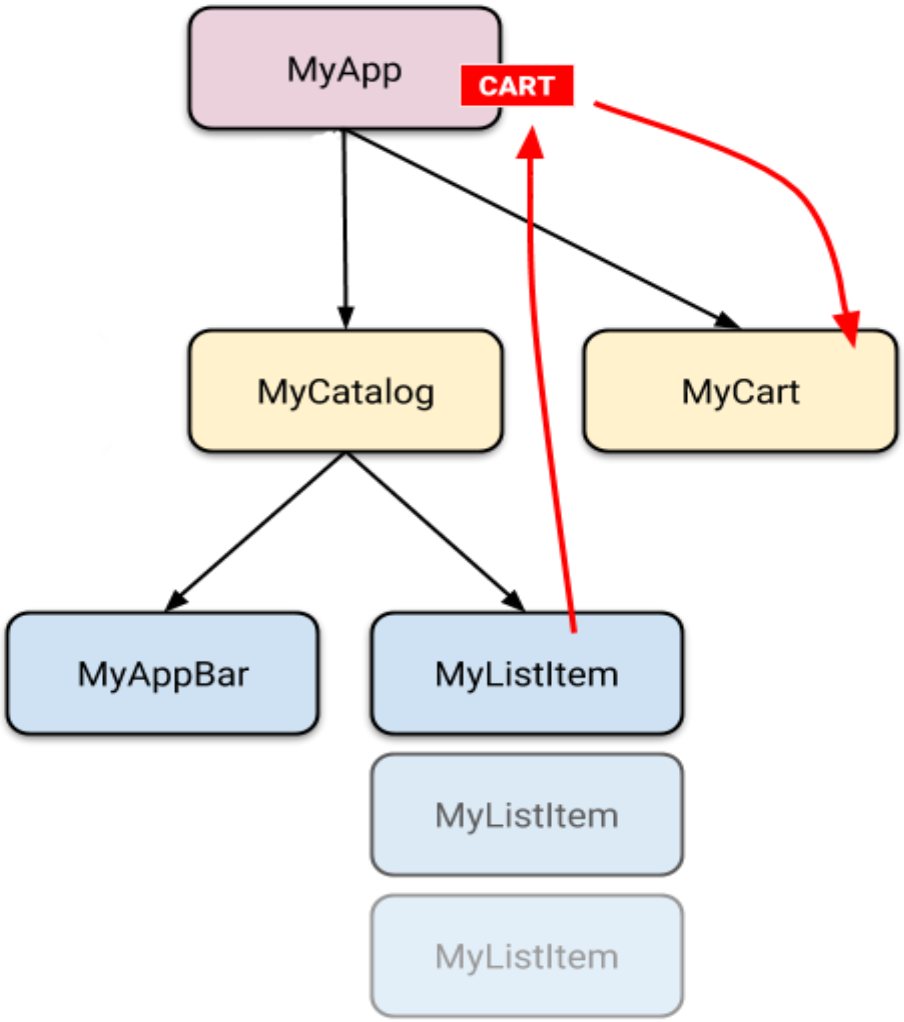
- Developers can read and visualize the layout of Dart very easily and effortlessly since it is declarative and programmatic.
- Unlike other programming languages, it supports the majority of the basic programming concepts like classes, interfaces, and functions.
- Arrays are not directly supported by 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.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

8 - 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.



Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

9 - 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.

10 - 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.

11 - 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

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

12 - 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.

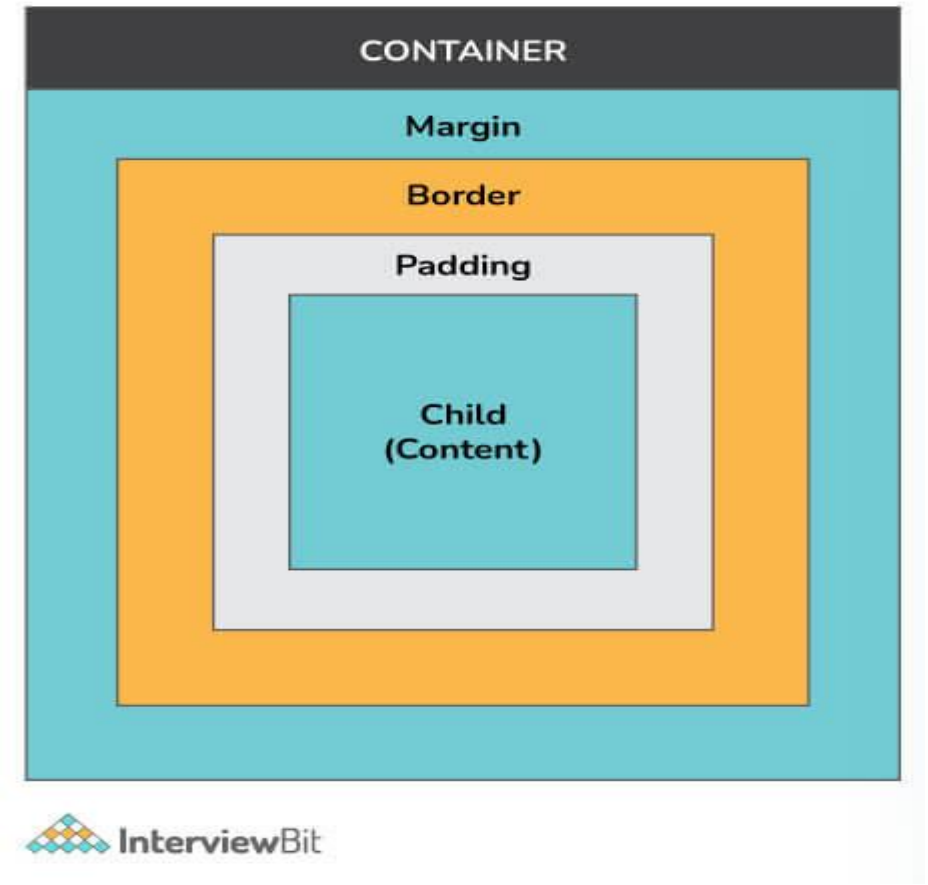
13 - What do you mean by keys in flutter? When one should use it.

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.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

14 - 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:



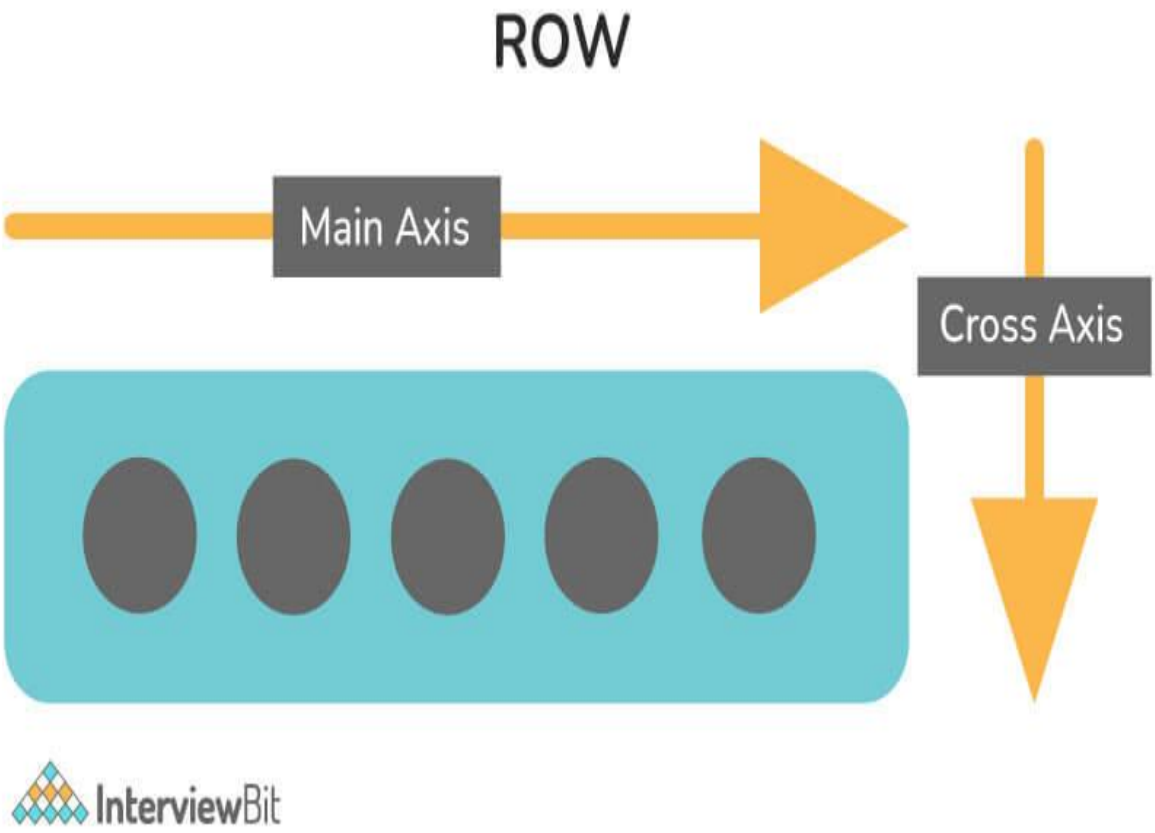
Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

15 - 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

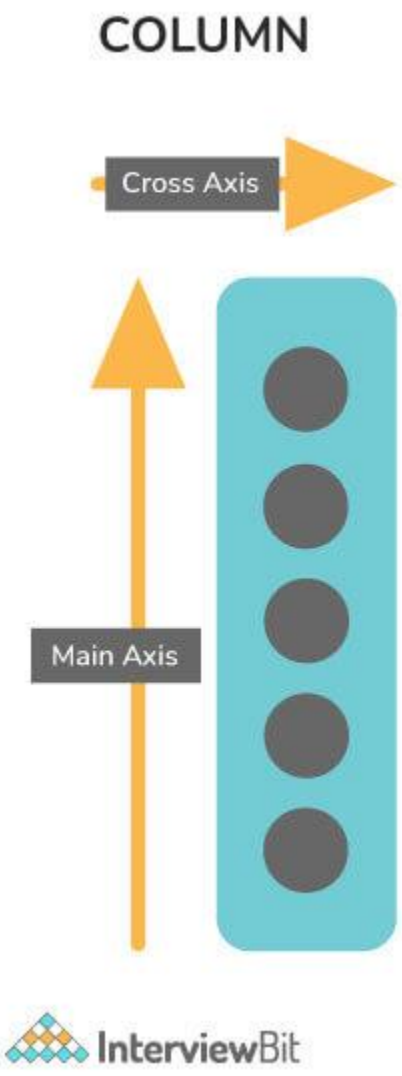
This can be better understood by looking at the image below:



Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

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:



Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

16 - 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.

17 - 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

18 - 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.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

19 - 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');
}
```

20 - 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).

21 - 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.

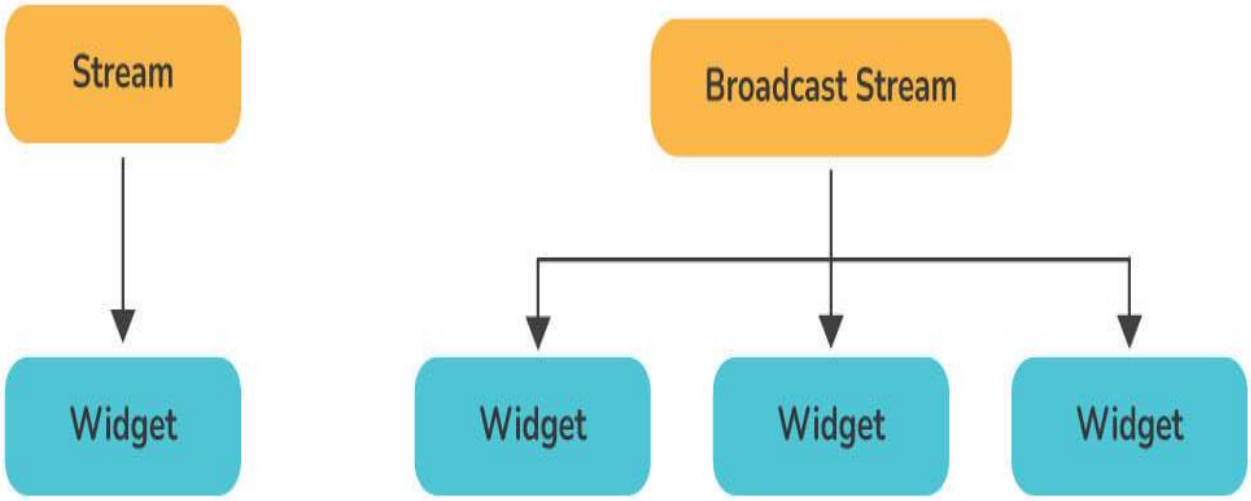
The Stream API provides the await for and listen() methods for processing streams. Streams can be created in many ways, but they can only be used in the same manner. Here is an example:

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

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

22 - 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:



InterviewBit

- **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.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

23 - 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.

24 - 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

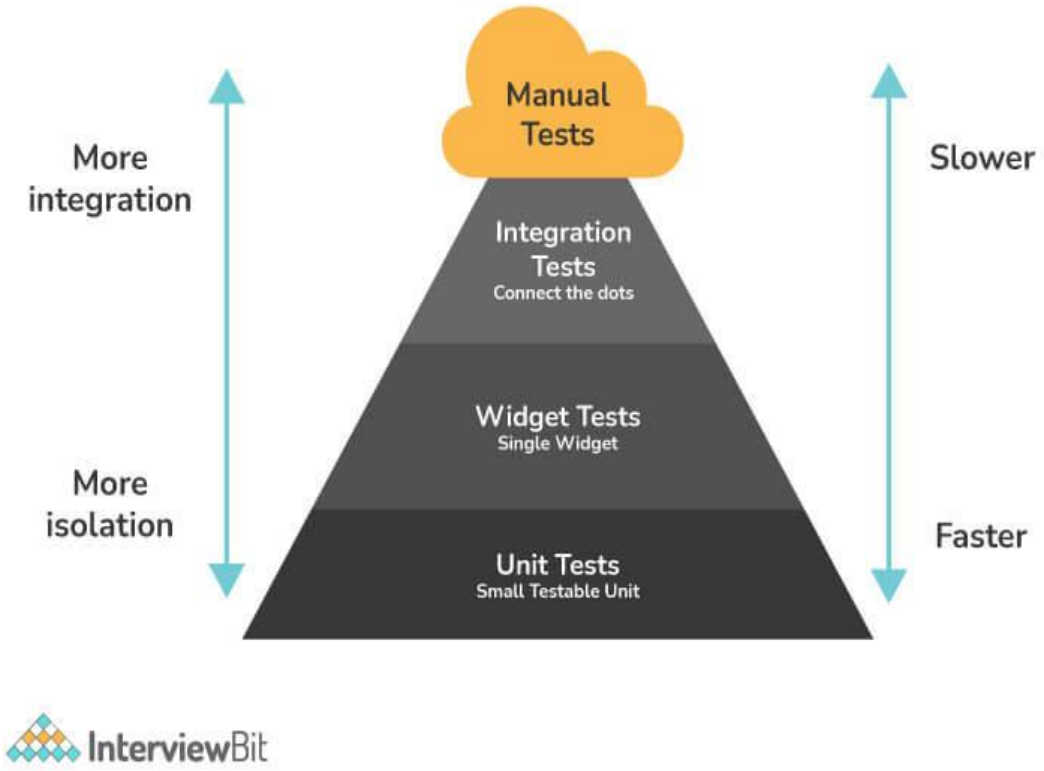
Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

25 - 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.

26 - 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.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

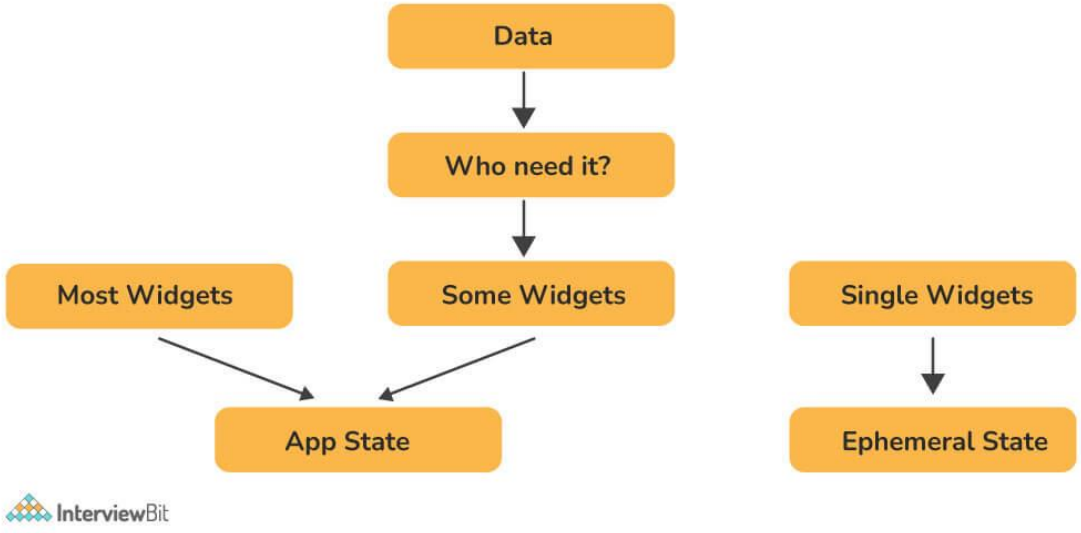
- **Integration tests:** Using Integration testing, you can test the critical flows of the entire app. It is important to check whether all widgets and services work together as expected. You can also use it to measure and benchmark the performance of your app.

27 - 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:



Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

28 - Explain pubspec.yaml file.

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.).

29 - 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.

30 - 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>

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

31 - 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

32 - 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.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

33 - 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 are null. If not, the getter is called and returned.
```

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.

34 - What is Dart?

Dart is a general-purpose, object-oriented programming language with C-style syntax. It is open-source and developed by Google in 2011. The purpose of Dart programming is to create a frontend user interfaces for the web and mobile apps. It is an important language for creating Flutter apps. The Dart language can be compiled both AOT (Ahead-of-Time) and JIT (Just-in-Time.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

35 - 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.

36 - 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.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

37 - 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 StatelessWidget 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.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

38 - 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.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

39 - What are some approaches to state management?

- **setState** is useful only for managing the ephemeral state; that is when the state is widget-specific. It is very straightforward and, in fact, it is not recommended to use a different approach for ephemeral state management.
- **InheritedWidget**: as obvious from the name, this approach is used to manage data transmission between ancestors and children.
- **Redux:** is useful for app state management and particularly for large, complex apps. Many web developers are already familiar with Redux and so using it in Flutter is not much of a hassle.
- Other popular state management approaches include **Provider, BLoC, MobX, Riverpod**, and so on.

40 - Differentiate between required and optional parameters in Dart

Required Parameters

Dart required parameters are the arguments that are passed to a function and the function or method required all those parameters to complete its code block.

```
findVolume(int length, int breath, int height) {
 print('length = $length, breath = $breath, height = $height');
}

findVolume(10,20,30);
```

Optional Parameters

- Optional parameters are defined at the end of the parameter list, after any required parameters.
- In Flutter/Dart, there are 3 types of optional parameters: - Named - Parameters wrapped by `{ }` - eg. `getUrl(int color, [int favNum])` - Positional - Parameters wrapped by `[ ]`) - eg. `getUrl(int color, {int favNum})` - Default - Assigning a default value to a parameter. - eg. `getUrl(int color, [int favNum = 6])`

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

41 - What is ScopedModel / BLoC Pattern?

ScopedModel and **BLoC** (Business Logic Components) are common Flutter app architecture patterns to help separate business logic from UI code and using fewer *Stateful Widgets*.

- **Scoped Model** is a third-party package that is not included into Flutter framework. It's a set of utilities that allow you to easily pass a data Model from a parent Widget down to its descendants. In addition, it also rebuilds all of the children that use the model when the model is updated. This library was originally extracted from the Fuchsia codebase.

- **BLoC** stands for Business Logic Components. It helps in managing state and make access to data from a central place in your project. The gist of BLoC is that everything in the app should be represented as stream of events: widgets submit events; other widgets will respond. BLoC sits in the middle, managing the conversation.

42 - Why do we pass functions to widgets?

- Functions are first class objects in *Dart* and can be passed as parameters to other functions.
- We pass a function to a *widget* essentially saying, “invoke this function when something happens”.
- Callbacks using interfaces like *Android* (<Java 8) have too much boilerplate code for a simple callback.

Java Functions are first class objects in Dart and can be passed as parameters to other functions.callback:

```
button.setOnClickListener(new View.OnClickListener() {
    @override
    public void onClick(View view) {
      // Do something here
    }
  }
);
```

(Notice that this is only the code for setting up a listener. Defining a button requires separate *XML* code.)

Dart equivalent:

```
FlatButton(
  onPressed: () {
    // Do something here
  }
)
```

(**Dart** does both declaration as well as setting up the callback.) This becomes much cleaner and organised and helps us avoid unnecessary complication.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

43 - In What technology is Flutter built?

Flutter is built using C, C++, Skia - 2D rendering engine and Dart (a modern, concise, object-oriented language). For you can visit Flutter System Architecture and Flutter architectural overview.

44 -What is a Cookbook?

The **Cookbook** provides solutions or recipes for common occurring problems while developing flutter apps. Each recipe is an independent complete solution and can be referenced to help you build up an app. For more you can refer Cookbook.

45 - What is use of Navigation.push and Navigation.pop function?

The push method is used to add a route to the stack of routes managed by the navigator. **The pop** method is used to remove the current route from the stack of routes managed by the navigator.

46 - What is Flex box in Flutter?

The Flex class in Flutter is used to display its children in a one-dimensional array. With this widget, you can control the axis where the children are placed. This axis is called as the main axis.

47 - What are staggered Animations?

Staggered animation utilizes few animation items to include **consecutive** or **covering animations**. An animation controller is used to control all the animations. You can use multiple animation objects to create staggered animations and each animation object specifies the animation during the interval. It is a straightforward approach to create visual changes in a series of operations rather than all at once.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

48 - How to access screen size in flutter?

We can access screen size and other properties like pixel density, aspect ratio etc with the help of MediaQuery.

Syntax:

```
MediaQuery.of(context).size.width;

MediaQuery.of(context).size.height;
```

49 - What are the similarities and differences of Future and Stream?

Ans :

Similarity:

Future and Stream both work asynchronously.

Both have some potential value.

Differences:

A Stream may be a combination of Futures.

Future has just one response but Stream could have any number of Response.

50 - How is whenCompleted() different from then() in Future?

Ans: whenComplete will fire a function either when the longer term completes with a mistake or not, while .then returns a replacement Future which is completed with the results of the decision to onValue (if this future completes with a value) or to onError (if this future completes with an error)

.whenCompleted is that the asynchronous equivalent of a "finally" block.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

51 - What Are the varied Methods to control Strings?

Ans : There are various methods to control string that are given in table:

String Methods Description

toLowerCase(): It converts all the string character into small letter .

toUpperCase(): It converts all the string character into capital .

trim(): It returns the string with none leading and trailing whitespace.

compareTo(): It compare objects to a different objects.

52 -What is a Spacer widget?

The Spacer widget controls how much space appears between widgets in a row or column.

53 - What is an AspectRatio widget used for?

AspectRatio Widget tries to find the best size to maintain aspect ration while respecting it's layout constraints. The AspectRatio Widget can be used to adjust the aspect ratio of widgets in your app.

54 - Explain SafeArea in flutter.

SafeArea is a widget that inserts its child by sufficient padding to avoid intrusions by the operating system. For example, this will indent the child to avoid the status bar at the top of the screen.

55 - What is Reverse property in a ListView flutter?

Reversing a List makes the first element to be placed at last position, second element placed at last but one position and so on.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

56 - When should you use WidgetsBindingObserver?

WidgetsBindingObserver should be used when we want to listen to the and call stop/start on our services.

57 - What is the difference between Expanded and Flexible widgets?

 is just a shorthand for
Using expanded this way:

```
Expanded(
      child: Foo(),
);
```

is strictly equivalent to:

```
Flexible(
      fit: FlexFit.tight,
      child: Foo(),
);
```

You may want to use over when you want a different , useful in some responsive layouts.
The difference between and is that loose will allow its child to have a maximum size while tight forces that child to fill all the available space.

58 - Is CI/CD possible in Flutter?

Yes, CI/CD is possible in Flutter. There is CI/CD tool dedicated to Flutter the name is CODE MAGIC. With the help of CODE MAGIC we can easily automate the process of CI/CD for flutter apps from single automation.

59 - How to avoid widget remount while working with Bottom navigation tabs?

The effective way to avoid the no of widget rebuilds while working with multiple Widgets as interfaces of a app that uses Bottom Navigation Tabs as it's stratergy it to utilize IndexedStack that will increase the app performace by 2x and reduces the unnecessary widget rebuilds on tab changes.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

60 - How do you reduce widget rebuild?

You rebuild widgets when the state changes. This is normal and desirable, because it allows the user to see the state changes reflected in the UI. However, rebuilding parts of the UI that don't need to change is wasteful.
There are several things you can do to reduce unnecessary widget rebuilding.
The first is to refactor a large widget tree into smaller individual widgets, each with its own build method.
Whenever possible, use the const constructor, because this will tell Flutter that it doesn't need to rebuild the widget.
Keep the subtree of a stateful widget as small as possible. If a stateful widget needs to have a widget subtree under it, create a custom widget for the stateful widget and give it a child parameter.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

StatefulWidget lifecycle

on Friday, 24th of July, 2020

When a Flutter builds a `StatefulWidget`, it creates a `State` object. This object is where all the **mutable** state for that widget is held.

The concept of state is defined by two things:

1. The data used by the widget might change.

2. The data *can't* be read synchronously when the widget is built. (All state must be established by the time the `build` method is called).

The lifecycle has the following simplified steps:

- createState()
- mounted == true
- initState()
- didChangeDependencies()
- build()
- didUpdateWidget()
- setState()
- deactivate()
- dispose()
- mounted == false

Why Are StatefulWidget and State Separate Classes?

In one word: performance.

The tldr version is that `State` objects are long lived, but `StatefulWidget`s (and all `Widget` subclasses) are thrown away and rebuilt whenever configuration changes. It's very inexpensive ie cheap for Flutter to rebuild a mutable widget.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

As `State` isn't blown away on every rebuild, it avoids expensive computations, and gets at the `state`s property, getters, setters etc everytime something is rebuilt frame by frame.

Important is that this is what allows Flutter animations to exist. As `State` isn't thrown away, it can constantly be rebuilding it's `Widget` in response to data changes, and when required, if any.

1. createState()

When Flutter is instructed to build a `StatefulWidget`, it immediately calls `createState()`. This method *must* exist. A StatefulWidget rarely needs to be more complicated than this.

```
class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => new _MyHomePageState();
}
```

2. mounted is true

When `createState` creates the state class, a `buildContext` is assigned to that state.

A `BuildContext` is, overly simplified, the place in the widget tree in which this widget is placed.

All widgets have a `bool this.mounted` property. It is turns `true` when the `buildContext` is assigned. It is an error to call `setState` when a widget is unmounted.

tip: This property is useful when a method on your state calls `setState()` but it isn't clear when or how often that method will be called. Perhaps its being called in response to a stream updating. You can use `if (mounted) {...` to make sure the State exists before calling `setState()`.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

3. initState()

This is the first method called when the widget is created (after the class constructor, of course.)

`initState` is called **once and only once**. It must also call `super.initState()`.

This `@override` method is the best time to:

1. Initialize data that relies on the specific BuildContext for the created instance of the widget.
2. Initialize properties that rely on this widgets 'parent' in the tree.
3. Subscribe to Streams, ChangeNotifiers, or any other object that could change the data on this widget.

```
@override
initState() {
  super.initState();
  // Add listeners to this class
  cartItemStream.listen((data) {
    _updateWidget(data);
  });
}
```

4. didChangeDependencies()

The didChangeDependencies method is called immediately after `initState` on the first time the widget is built.

It will also be called whenever an object that this widget *depends on data from* is called. For example, if it relies on an InheritedWidget, which updates.

`build` is **always** called after `didChangeDependencies` is called, so this is rarely needed. However, this method is the first change you have to call `BuildContext.inheritFromWidgetOfExactType`. This essentially would make this State 'listen' to changes on a Widget it's inheriting data from.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

The docs also suggest that it could be useful if you need to do network calls (or any other expensive action) when an InheritedWidget updates.

5. build()

This method is called often (think fps + `render`). It is a required, `@override` and must return a `Widget`.

Remember that in Flutter all gui is a widget with a child or children, even 'Padding', 'Center'.

6. didUpdateWidget(Widget oldWidget)

`didUpdateWidget()` is called if the parent widget changes and has to rebuild this widget (because it needs to give it different data), but it's being rebuilt with the same `runtimeType`, then this method is called.

This is because Flutter is re-using the `state`, which is long lived. In this case, required is to initialize some data again, as one would in `initState()`.

If the state's `build()` method relies on a Stream or other object that can change, unsubscribe from the old object and re-subscribe to the new instance in `didUpdateWidget()`.

tip: This method is basically the replacement for 'initState()' if it is expected the `Widget` associated with the widgets's `state` nrrds to to be rebuilt!

Flutter always called `build()` after this, so any subsequent further calls to `setState` is redundant.

```
@override
void didUpdateWidget(Widget oldWidget) {
  if (oldWidget.importantProperty != widget.importantProperty) {
    _init();
  }
}
```

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

7. setState()

The 'setState()' method is called often from the Flutter framework itself and from the developer.

It is used to notify the framework that "data has changed", and the widget at this `build context` should be rebuilt.

`setState()` takes a callback which **cannot be async**. It is for this reason it can be called often as required, because repainting is cheap :-)

```
void updateProfile(String name) {
 setState(() => this.name = name);
}
```

8. deactivate()

This is rarely used.

'deactivate()' is called when `State` is removed from the tree, *but it might be reinserted* before the current frame change is finished. This method exists basically because `State` objects can be moved from one point in a tree to another.

9. dispose()

'dispose()' is called when the `State` object is removed, which is permanent.

This method is where to unsubscribe and cancel all animations, streams, etc.

10. mounted is false

The `state` object can never remount, and an error is thrown is `setState()` is called.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

Flutter’s threads

Flutter uses several threads to do its work, though only two of the threads are shown in the overlay. All of your Dart code runs on the UI thread. Although you have no direct access to any other thread, your actions on the UI thread have performance consequences on other threads.

Platform thread

> The platform’s main thread. Plugin code runs here. For more information, see the UIKit documentation for iOS, or the MainThread documentation for Android. This thread is not shown in the performance overlay.

UI thread

> The UI thread executes Dart code in the Dart VM. This thread includes code that you wrote, and code executed by Flutter’s framework on your app’s behalf. When your app creates and displays a scene, the UI thread creates a *layer tree*, a lightweight object containing device-agnostic painting commands, and sends the layer tree to the raster thread to be rendered on the device. *Don’t block this thread!* Shown in the bottom row of the performance overlay.

Raster thread (previously known as the GPU thread)

> The raster thread takes the layer tree and displays it by talking to the GPU (graphic processing unit). You cannot directly access the raster thread or its data but, if this thread is slow, it’s a result of something you’ve done in the Dart code. Skia, the graphics library, runs on this thread. Shown in the top row of the performance overlay. This thread was previously known as the “GPU thread” because it rasterizes for the GPU. But it is running on the CPU. We renamed it to “raster thread” because many developers wrongly (but understandably) assumed the thread runs on the GPU unit.

I/O thread

> Performs expensive tasks (mostly I/O) that would otherwise block either the UI or raster threads. This thread is not shown in the performance overlay.

For links to more information and videos, see The Framework architecture on the GitHub wiki, and the community article, The Layer Cake.

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005

Questions:

Flutter Interview Questions and Answers | raywenderlich.com

Resources:

Top Flutter Interview Questions (2021) - InterviewBit
29 Flutter Interview Questions Mobile Devs Need To Know (ANSWERED for 2020) | FullStack.Cafe
20+ Flutter Interview Questions in 2021 - Online... (onlineinterviewquestions.com)
Flutter and Dart Interview Questions and Answers | Basic and Advanced (mytectra.com)
15 Most Important Flutter Interview Questions 2021 with Answers - Interview Exam | Interview Questions and Answers with Exam Preparation

Mahmoud salah
Mahmoudsalah37@gmail.com
+20 155 646 0005