Flutter Interview Questions

Questions with Answer

Flutter -Questions with Answer Key

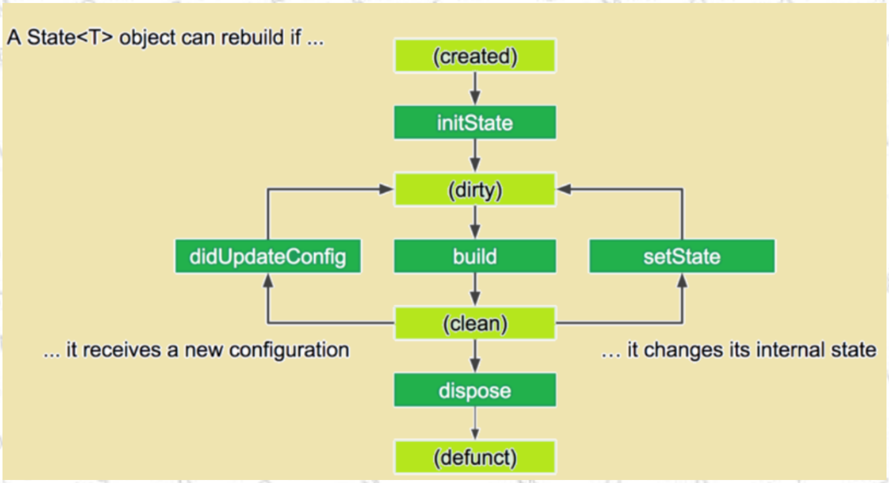
1.What is the difference between a StatelessWidget and a StatefulWidget in Flutter?

Stateless Widget A stateless widget can not change their state during the runtime of an app which means it can not redraw its self while the app is running. Stateless widgets are immutable.
Stateful Widget A stateful widget can redraw itself multiple times, while the app is running which means its state is mutable. For example, when a button is pressed, the state of the widget is changed

---

2.Explain the Stateful Widget Lifecycle?

The lifecycle has the following simplified steps: createState() mounted == true initState() didChangeDependencies() build() didUpdateWidget() setState() deactivate() dispose() mounted == false


A State<T> object can rebuild if ...
(created)
initState
(dirty)
didUpdateConfig
build
setState
(clean)
... it receives a new configuration
... it changes its internal state
dispose
(defunct)

3.What is Flutter tree shaking (flutter web)?

When compiling a Flutter web application, the JavaScript bundle is generated by the dart2js compiler. A release build has the highest level of optimization, which includes tree shaking your code. Tree shaking is the process of eliminating dead code, by only including code that is guaranteed to be executed. This means that you do not need to worry about the size of your app's included libraries because unused classes or functions are excluded from the compiled JavaScript bundle

---

4.What is a Spacer widget?

Spacer manages the empty space between the widgets with flex container. Evenly with the Row and Column MainAxis alignment we can manage the space as well

---

5.What is the difference between hot restart and hot reload?

What is Hot Reload in Flutter:
Flutter hot reload features works with combination of Small r key on command prompt or Terminal. Hot reload feature quickly compile the newly added code in our file and sent the code to Dart Virtual Machine. After done updating the Code Dart Virtual Machine update the app UI with widgets. Hot Reload takes less time then Hot restart. There is also a draw back in Hot Reload, If you are using States in your application then Hot Reload preservers the States so they will not update on Hot Reload our set to their default values

What is Hot Restart in Flutter:
Hot restart is much different than hot reload. In Hot restart it destroys the preserves State value and set them to their default. So if you are using States value in your application then After every hot restart the developer gets fully compiled application and all the states will set to their defaults. The app widget tree is completely rebuilt with new typed code. Hot Restart takes much higher time than Hot reload

Flutter -Questions with Answer Key

7.Why is the build() method on State and not StatefulWidget?

8.What is a pubspec file in Dart?

The pubspec file manages the assets and dependencies for a Flutter app

---

9.How is Flutter native?

Flutter uses only the canvas of the native platform and draws the UI and all the components from scratch. All the UI elements look the same as native ones. This mainly reduces the burden of time for converting through some language to the native one and speeds up the UI rendering time. As a result, the UI performance is remarkably high

---

10.What is a Navigator and what are Routes in Flutter?

Navigation and routing are some of the core concepts of all mobile application, which allows the user to move between different pages. We know that every mobile application contains several screens for displaying different types of information. For example, an app can have a screen that contains various products. When the user taps on that product, immediately it will display detailed information about that product

---

11.What is a PageRoute?

Allow us to add animation transaction to the route

12.Explain async, await and Future?

Async means that this function is asynchronous and you might need to wait a bit to get its result. Await literally means - wait here until this function is finished and you will get its return value. Future is a type that 'comes from the future' and returns value from your asynchronous function. It can complete with success(.then) or with an error(.catchError)

13.how can you update a listview dynamically?

By using setState to update the listview item source and rebuild the UI

14.What is a Stream?

A stream is like a pipe, you put a value on the one end and if there's a listener on the other end that listener will receive that value. A Stream can have multiple listeners and all of those listeners will receive the same value when it's put in the pipeline. The way you put values on a stream is by using a StreamController

15.What are keys in Flutter and when should you use it?

You don't need to use Keys most of the time, the framework handles it for you and uses them internally to differentiate between widgets. There are a few cases where you may need to use them though.

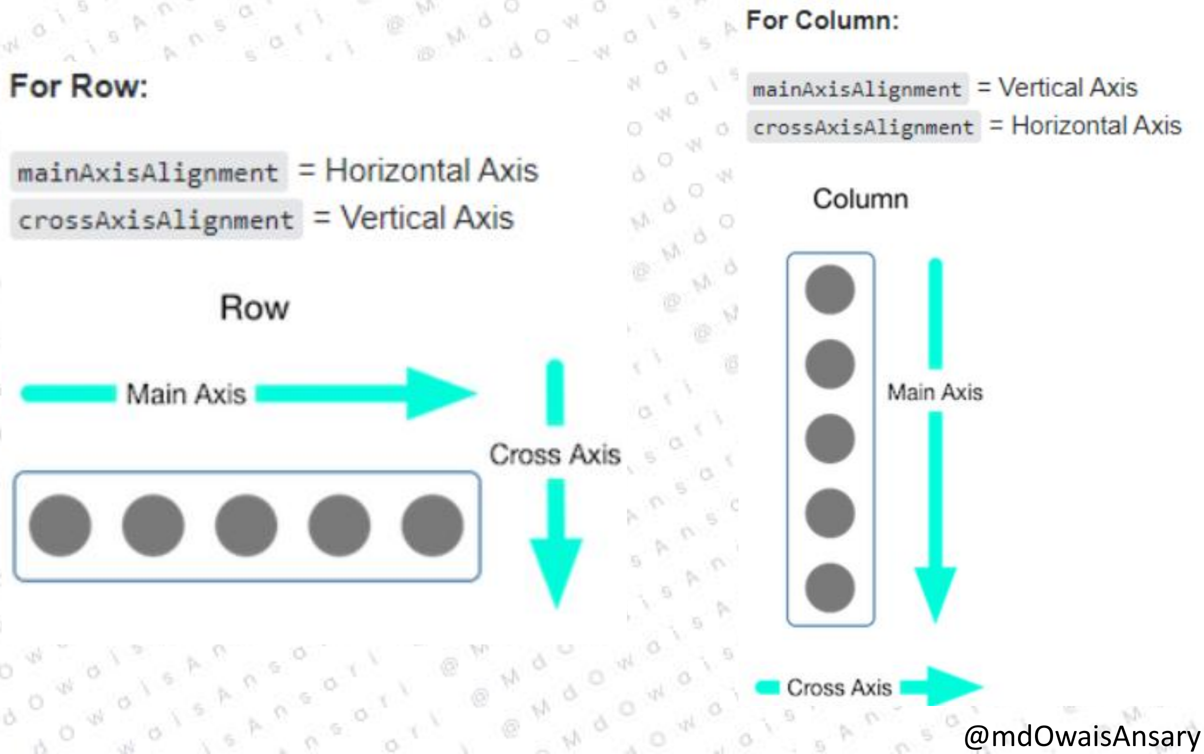
A common case is if you need to differentiate between widgets by their keys, ObjectKey and ValueKey can be useful for defining how the widgets are differentiated

Another example is that if you have a child you want to access from a parent, you can make a GlobalKey in the parent and pass it to the child's constructor. Then you can do globalKey.state to get the child's state (say for example in a button press callback). Note that this shouldn't be used excessively as there are often better ways to get around it

16.What are GlobalKeys?

GlobalKeys have two uses: they allow widgets to change parents anywhere in your app without losing state, or they can be used to access information about another widget in a completely different part of the widget tree. An example of the first scenario might if you wanted to show the same widget on two different screens, but holding all the same state, you'd want to use a GlobalKey.

17.When should you use mainAxisAlignment and crossAxisAlignment?

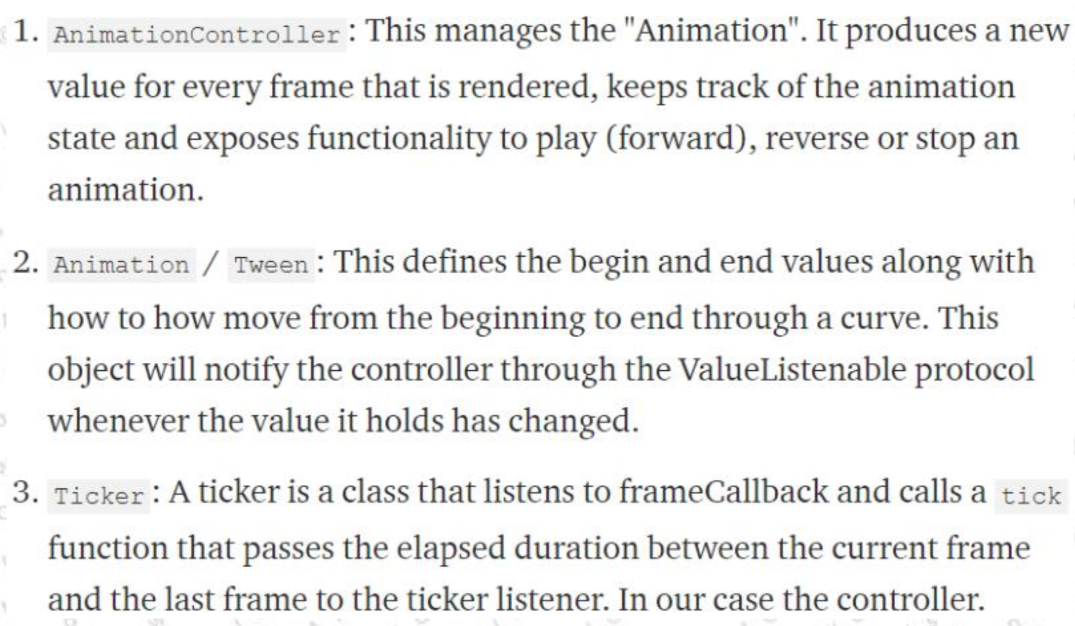

For Row:
mainAxisAlignment = Horizontal Axis
crossAxisAlignment = Vertical Axis
Row
Main Axis
Cross Axis
For Column:
mainAxisAlignment = Vertical Axis
crossAxisAlignment = Horizontal Axis
Column
Main Axis
Cross Axis

18.When can you use double.INFINITY?

When you want the widget to be big as the parent widget allow

---

19.What is Ticker, Tween and AnimationController?



1. `AnimationController` : This manages the "Animation". It produces a new value for every frame that is rendered, keeps track of the animation state and exposes functionality to play (forward), reverse or stop an animation.

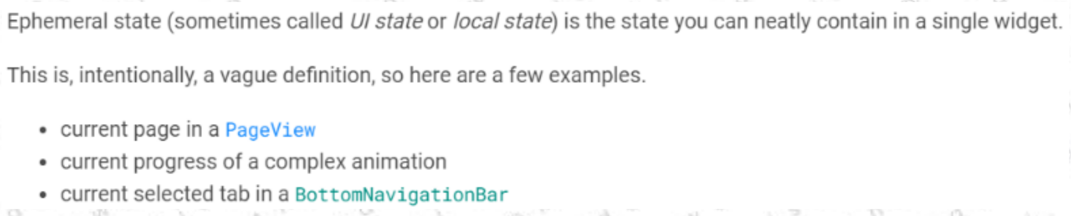
2. `Animation` / `Tween` : This defines the begin and end values along with how to how move from the beginning to end through a curve. This object will notify the controller through the ValueListenable protocol whenever the value it holds has changed.

3. `Ticker` : A ticker is a class that listens to frameCallback and calls a `tick` function that passes the elapsed duration between the current frame and the last frame to the ticker listener. In our case the controller.

Animation Sequences To achieve sequence animation we'll introduce a new Widget that also helps with reducing animation code called AnimatedBuilder which allows you to rebuild your widget through a builder function every time a new animation value is calculated

---

20.What is ephemeral state?



Ephemeral state (sometimes called *UI state* or *local state*) is the state you can neatly contain in a single widget.

This is, intentionally, a vague definition, so here are a few examples.

- current page in a `PageView`
- current progress of a complex animation
- current selected tab in a `BottomNavigationBar`

21.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

---

22.How would you access StatefulWidget properties from its State?

Using the widget property

23.Mention two or more operations that would require you to use or turn a Future

1. Calling api using http
2. Getting result from geolocator package
3. With FutureBuilder widget

24.What is the purpose of a SafeArea?

SafeArea is basically a glorified Padding widget. If you wrap another widget with SafeArea, it adds any necessary padding needed to keep your widget from being blocked by the system status bar, notches, holes, rounded corners and other "creative" features by manufactures

---

25.When to use a mainAxisSize?

When you use MainAxisSize on your Column or Row, they will determine the size of the Column or Row along the main axis, i.e, height for Column and width for Row

---

27.List the Visibility widgets in flutter and the differences?

1. Visibility
2. Opacity
3. Offstage

28.Can we use Color and Decoration property simultaneously in the Container?

No
The color property is a shorthand for creating a BoxDecoration with a color field. If you are adding a box decoration, simply place the color on the BoxDecoration.

29.Inorder for the CrossAxisAlignment.baseline to work what is another property that we need to set?

crossAxisAlignment: CrossAxisAlignment.baseline textBaseline: TextBaseline.ideographic,

---

30.when should we use a resizeToAvoidBottomInset?

If true the body and the scaffold's floating widgets should size themselves to avoid the onscreen keyboard whose height is defined by the ambient MediaQuery's MediaQueryData.viewInsets bottom property.

For example, if there is an onscreen keyboard displayed above the scaffold, the body can be resized to avoid overlapping the keyboard, which prevents widgets inside the body from being obscured by the keyboard

---

32.What is the importance of a TextEditingController?

Whenever the user modifies a text field with an associated TextEditingController, the text field updates value and the controller notifies its listeners. Listeners can then read the text and selection properties to learn what the user has typed or how the selection has been updated

---

33.Why do we use a Reverse property in a Listview?

List animals = ['cat', 'dog', 'duck']; List reversedAnimals = animals.reversed.toList();

36.What is an UnmodifiableListView?

Cannot change the list items by adding or removing

37.Difference between these operators ?? and ?.

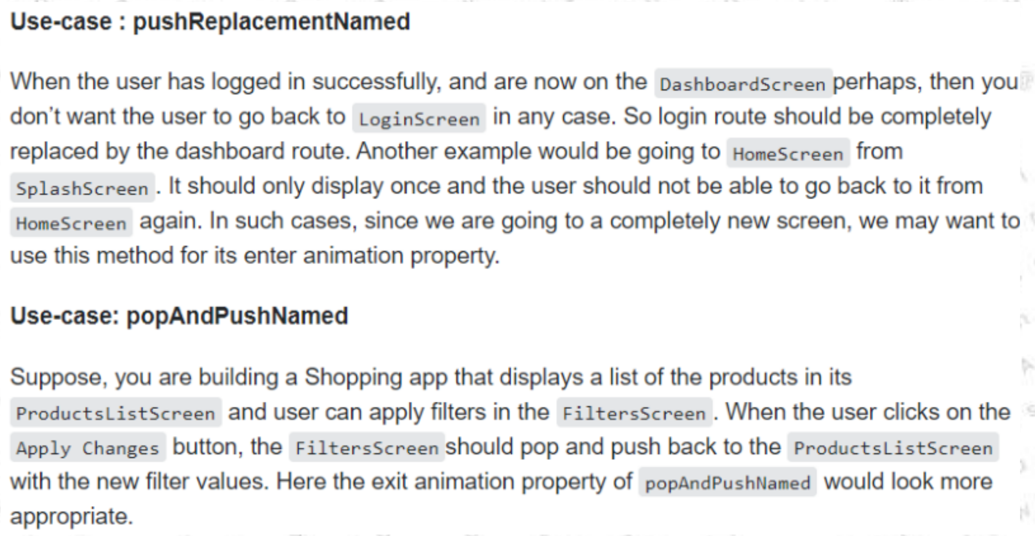
?? expr1 ?? expr2 If expr1 is non-null, returns its value; otherwise, evaluates and returns the value of expr2.

?. Like . but the leftmost operand can be null; example: foo?.bar selects property bar from expression foo unless foo is null (in which case the value of foo?.bar is null)

38.What is the purpose of ModalRoute.of()?

ModalRoute.of() method. This method returns the current route with the arguments
final args = ModalRoute.of(context).settings.arguments;

39.Difference between a Navigator.pushNamed and Navigator.pushReplacementNamed?



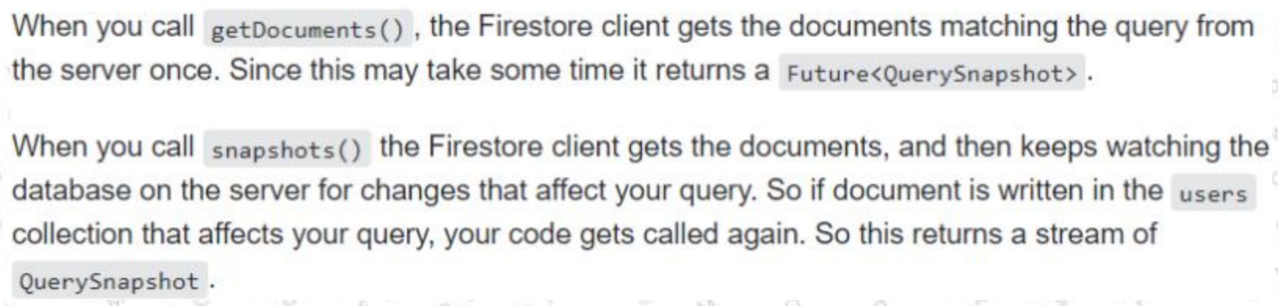
Use-case : pushReplacementNamed

When the user has logged in successfully, and are now on the `DashboardScreen` perhaps, then you don't want the user to go back to `LoginScreen` in any case. So login route should be completely replaced by the dashboard route. Another example would be going to `HomeScreen` from `SplashScreen`. It should only display once and the user should not be able to go back to it from `HomeScreen` again. In such cases, since we are going to a completely new screen, we may want to use this method for its enter animation property.

Use-case: popAndPushNamed

Suppose, you are building a Shopping app that displays a list of the products in its `ProductsListScreen` and user can apply filters in the `FiltersScreen`. When the user clicks on the `Apply Changes` button, the `FiltersScreen` should pop and push back to the `ProductsListScreen` with the new filter values. Here the exit animation property of `popAndPushNamed` would look more appropriate.

41.Difference between getDocuments() vs snapshots()?



When you call `getDocuments()`, the Firestore client gets the documents matching the query from the server once. Since this may take some time it returns a `Future<QuerySnapshot>`.

When you call `snapshots()` the Firestore client gets the documents, and then keeps watching the database on the server for changes that affect your query. So if document is written in the `users` collection that affects your query, your code gets called again. So this returns a stream of `QuerySnapshot`.

42.What is a vsync?

Vsync basically keeps the track of screen, so that Flutter does not renders the animation when the screen is not being displayed

---

43.When does the animation reach completed or dismissed status?

animations that progress from 0.0 to 1.0 will be dismissed when their value is 0.0. An animation might then run forward (from 0.0 to 1.0) or perhaps in reverse (from 1.0 to 0.0). Eventually, if the animation reaches the end of its range (1.0), the animation reaches the completed status.

44.Difference between AnimationController and Animation?

AnimationController is for how long the animation would be and how to control from time, upper and lower boundary, how to control data with time, length, sequence, etc. while AnimationTween is for the range of animation with time, colour, range, sequence, etc as long the animation would be while

---

46.Define a TweenAnimation ?

Short for in-betweening. In a tween animation, the beginning and ending points are defined, as well as a timeline, and a curve that defines the timing and speed of the transition. The framework calculates how to transition from the beginning point to the end point

---

47.State the importance of a Ticker ?

Ticker is the refresh rate of our animations. This is what we want to pause when our clock is hidden.

A bonus for using Ticker is that this allows the dev-tool to "slow" our animation. If we use "Slow animations", then our clock is slowed by 50%. This is a good sign, as it means it will be a lot easier to test our clock!

48.Why do we need a mixins ?

Mixins are very helpful when we want to share a behavior across multiple classes that don't share the same class hierarchy, or when it doesn't make sense to implement such a behavior in a superclass

---

49.When do you use the WidgetsBindingObserver?

To check when the system puts the app in the background or returns the app to the foreground

---

50.Why does the first flutter app take a very long developing time?

When you are going to build the Flutter app for the first time, it takes a very long time than usual because Flutter builds a device-specific IPA or APK file. In this process, the Xcode and Gradle are used to build a file, which usually takes a long time

---

51.Define what is an App State?

The App State is also called an application state or shared state. The app state can be distributed across multiple areas of your app and the same is maintained with user sessions.

Following are the examples of App State:

Login info User preferences The shopping cart of an e-commerce application

---

52.What do you know about Dart Isolates?

To gain concurrency Dart makes use of the Isolates method which works on its own without sharing memory but uses passing or message communication.

53.What are the two types of Streams available in Flutter?

Single subscription streams:
It is a popular and common type of stream. It consists of a series of events that are parts of a large whole. Here all events have to be delivered in a defined order without even missing a single event. It is a type of stream that you get when you get a web request or receive a file. This stream can only be listed once. Listing it, again and again, means missing initial values and overall stream makes no sense at all. When the listing starts in this stream the data gets fetched and provided in chunks.

Broadcast streams:
This stream is meant for the individual messages that can be handled one at a time. These types of streams are commonly used for mouse events in a browser. You can list this type of stream at any time. Multiple listeners can listen at a time and also you have a chance to listen after the cancellation of the previous subscription

---

54.What is a Flutter inspector?

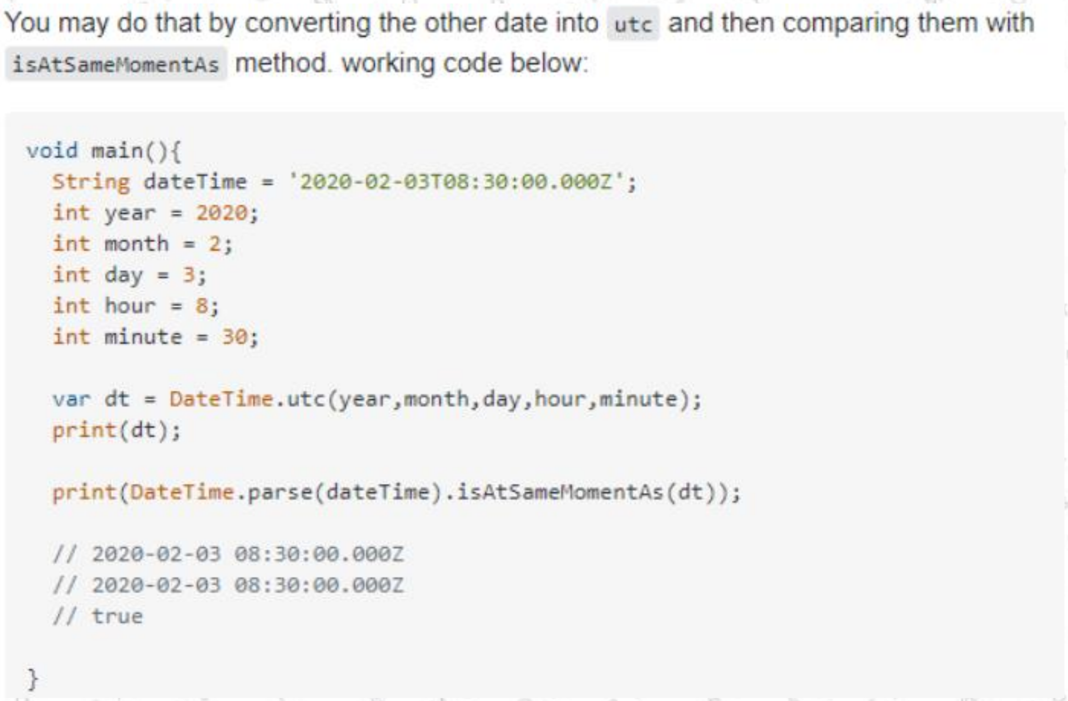
Flutter inspector is a tool that helps in visualizing and exploring the widget trees. It helps in understanding the present layout and diagnoses various layout issues

---

55.Stream vs Future?

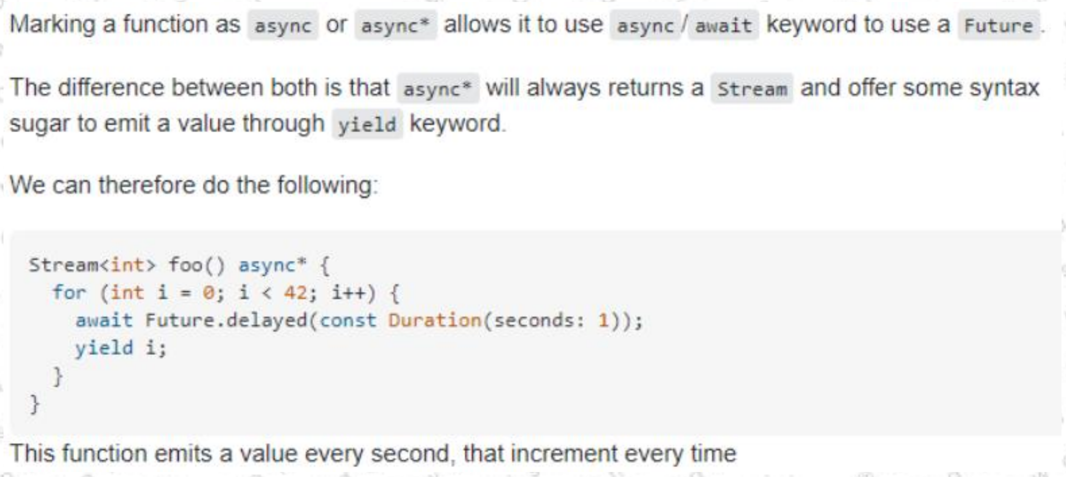
The difference is that Futures are about one-shot request/response (I ask, there is a delay, I get a notification that my Future is ready to collect, and I'm done!) whereas Streams are a continuous series of responses to a single request (I ask, there is a delay, then I keep getting responses until the stream dries up or I decide to close it and walk away)

56.How to compare two dates that are constructed differently in Dart?


You may do that by converting the other date into `utc` and then comparing them with `isAtSameMomentAs` method. working code below:

```
void main(){
  String dateTime = '2020-02-03T08:30:00.000Z';
  int year = 2020;
  int month = 2;
  int day = 3;
  int hour = 8;
  int minute = 30;

  var dt = DateTime.utc(year,month,day,hour,minute);
  print(dt);

  print(DateTime.parse(dateTime).isAtSameMomentAs(dt));

  // 2020-02-03 08:30:00.000Z
  // 2020-02-03 08:30:00.000Z
  // true

}
```

57.What's the difference between async and async* in Dart?



Marking a function as `async` or `async*` allows it to use `async` / `await` keyword to use a `Future`.

The difference between both is that `async*` will always returns a `Stream` and offer some syntax sugar to emit a value through `yield` keyword.

We can therefore do the following:

```dart
Stream<int> foo() async* {
  for (int i = 0; i < 42; i++) {
    await Future.delayed(const Duration(seconds: 1));
    yield i;
  }
}
```

This function emits a value every second, that increment every time

---

58.Debug vs Profile mode?

In debug mode, the app is set up for debugging on the physical device, emulator, or simulator.

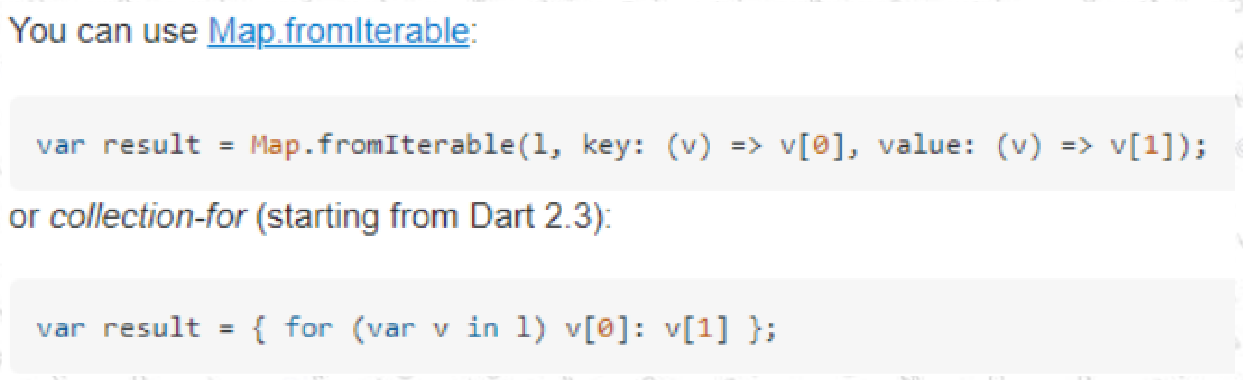
Debug

Assertions are enabled. Service extensions are enabled. Compilation is optimized for fast development and run cycles (but not for execution speed, binary size, or deployment). Debugging is enabled, and tools supporting source level debugging (such as DevTools) can connect to the process.

Profile In profile mode, some debugging ability is maintained—enough to profile your app’s performance. Profile mode is disabled on the emulator and simulator, because their behavior is not representative of real performance. On mobile, profile mode is similar to release mode, with the following differences:

Some service extensions, such as the one that enables the performance overlay, are enabled. Tracing is enabled, and tools supporting source-level debugging (such as DevTools) can connect to the process.

59.How to convert a List into a Map in Dart?



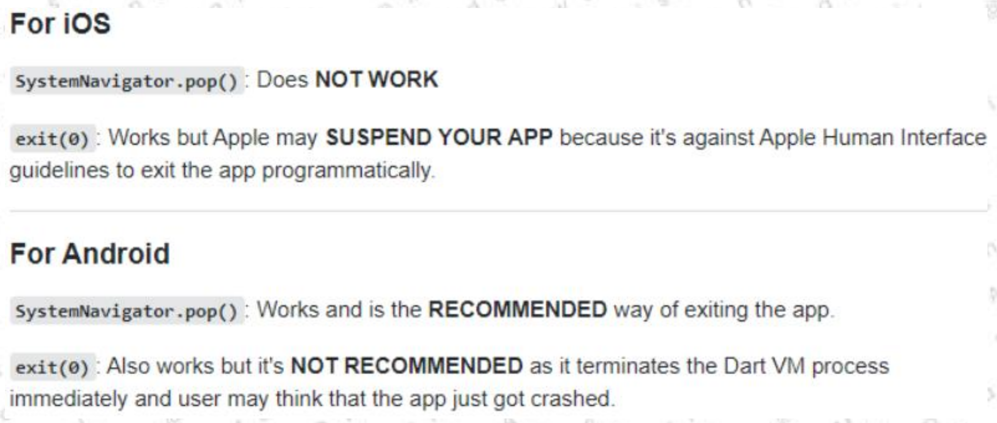
You can use Map.fromIterable:

```dart
var result = Map.fromIterable(l, key: (v) => v[0], value: (v) => v[1]);
```

or *collection-for* (starting from Dart 2.3):

```dart
var result = { for (var v in l) v[0]: v[1] };
```

62.Why is exit(0) not preferred for closing an app?


For iOS

`SystemNavigator.pop()` : Does **NOT WORK**

`exit(0)` : Works but Apple may **SUSPEND YOUR APP** because it's against Apple Human Interface guidelines to exit the app programmatically.

For Android

`SystemNavigator.pop()` : Works and is the **RECOMMENDED** way of exiting the app.

`exit(0)` : Also works but it's **NOT RECOMMENDED** as it terminates the Dart VM process immediately and user may think that the app just got crashed.

63.What is the difference between main function and the runApp() function in Flutter?

In Dart, main() acts as the entry point for the program whereas runApp() attaches the given widget to the screen.

64.What is Dart and why does Flutter use it?

Dart is AOT (Ahead Of Time) compiled to fast, predictable, native code, which allows almost all of Flutter to be written in Dart. This not only makes Flutter fast, virtually everything (including all the widgets) can be customized.

Dart can also 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 makes it easier to create smooth animations and transitions that run at 60fps. Dart can do object allocation and garbage collection without locks. And like JavaScript, Dart avoids preemptive scheduling and shared memory (and thus locks). Because Flutter apps are compiled to native code, they do not require a slow bridge between realms (e.g., JavaScript to native). They also start up much faster.

Dart allows Flutter to avoid the need for a separate declarative layout language like JSX or XML, or separate visual interface builders, because Dart's declarative, programmatic layout is easy to read and visualize. And with all the layout in one language and in one place, it is easy for Flutter to provide advanced tooling that makes layout a snap.
Developers have found that Dart is particularly easy to learn because it has features that are familiar to users of both static and dynamic languages

65.Where are the layout files? Why doesn’t Flutter have layout files?

In the Android framework, we separate an activity into layout and code. Because of this, we need to get references to views to work on them in Java. (Of course Kotlin lets you avoid that.) The layout file itself would be written in XML and consist of Views and ViewGroups.

Flutter uses a completely new approach where instead of Views, you use widgets. A View in Android was mostly an element of the layout, but in Flutter, a Widget is pretty much everything. Everything from a button to a layout structure is a widget. The advantage here is in customisability. Imagine a button in Android. It has attributes like text which lets you add text to the button. But a button in Flutter does not take a title as a string, but another widget. Meaning inside a button you can have text, an image, an icon and pretty much anything you can imagine without breaking layout constraints. This also lets you make customised widgets pretty easily whereas in Android making customised views is a rather difficult thing to do

---

66.What is the difference between final and const in Flutter?

final means single-assignment: A final variable or field must have an initializer. Once assigned a value, a final variable's value cannot be changed. final modifies variables.

const has a meaning that's a bit more complex and subtle in Dart. const modifies values. You can use it when creating collections, like const [1, 2, 3], and when constructing objects (instead of new) like const Point(2, 3). Here, const means that the object's entire deep state can be determined entirely at compile time and that the object will be frozen and completely immutable.

Const objects have a couple of interesting properties and restrictions:

They must be created from data that can be calculated at compile time. A const object does not have access to anything you would need to calculate at runtime. 1 + 2 is a valid const expression, but new DateTime.now() is not.

They are deeply, transitively immutable. If you have a final field containing a collection, that collection can still be mutable. If you have a const collection, everything in it must also be const, recursively.

They are canonicalized. This is sort of like string interning: for any given const value, a single const object will be created and re-used no matter how many times the const expression(s) are evaluated.