Flutter & Dart Interview Questions (Basic to Advanced)

1. What is Dart?

Dart is a client-optimized language developed by Google for building fast apps on multiple platforms.

2. What is Flutter?

Flutter is an open-source UI framework by Google for building natively compiled apps for mobile, web, and desktop from a single codebase.

3. Difference between Dart and Flutter?

Dart is the programming language; Flutter is the framework built using Dart.

4. What is a Widget?

A widget is the basic building block of Flutter’s UI. Everything in Flutter is a widget.

5. Difference between StatelessWidget and StatefulWidget?

StatelessWidget has immutable state; StatefulWidget maintains a mutable state.

6. What is the widget tree?

The widget tree is the hierarchy of widgets that defines the structure of the UI.

7. What is BuildContext?

It is the handle to the location of a widget in the widget tree, used for accessing theme, media query, or navigation.

8. Explain the Flutter rendering process.

Flutter converts widgets into elements, then render objects, and finally paints them via the Skia engine.

9. What is Hot Reload?

It allows developers to update code and instantly see changes without restarting the app, preserving the state.

10. What is Hot Restart?

It restarts the entire app and resets the state, unlike hot reload.

11. Explain the role of Dart’s async/await.

Used to write asynchronous code that looks synchronous, improving readability.

12. What are Futures in Dart?

Futures represent potential values or errors that will be available at some time in the future.

13. What are Streams in Dart?

Streams deliver a sequence of asynchronous data events.

14. What is an Isolate?

An isolate is an independent thread of execution that runs code in parallel without shared memory.

15. What is the compute() function in Flutter?

A helper that runs expensive functions in a separate isolate for background processing.

16. What is the use of Keys in Flutter?

Keys preserve the state of widgets when the widget tree rebuilds.

17. What is InheritedWidget?

It efficiently passes data down the widget tree and rebuilds only dependent widgets.

18. What is setState() used for?

It notifies Flutter that the state of a widget has changed and the UI should be rebuilt.

19. Explain the widget lifecycle.

initState() → build() → didUpdateWidget() → dispose().

20. What is GlobalKey?

A key that uniquely identifies a widget across rebuilds and allows access to its state.

21. What is Navigator in Flutter?

Navigator manages a stack of routes (screens) for navigation.

22. Difference between Navigator.push and pushReplacement?

push adds a new route; pushReplacement removes the current one and pushes a new route.

23. What is Provider?

A state management library based on InheritedWidget that allows easy dependency injection and reactivity.

24. What are Streams used for in Flutter?

To listen to real-time data changes, such as Firebase updates.

25. What is Bloc?

A pattern for managing state using streams and reactive programming principles.

26. What is Riverpod?

A newer, compile-time safe state management library that improves upon Provider.

27. What is GetX?

A lightweight Flutter package for state management, navigation, and dependency injection.

28. What is MobX?

A reactive state management solution that tracks observable state changes automatically.

29. What are Slivers?

Portions of scrollable areas that can change size or shape dynamically.

30. What is CustomPainter?

Used to draw custom graphics or shapes directly on the canvas.

31. Explain AnimationController.

A controller that manages animation timing and triggers rebuilds when animation values change.

32. What are implicit animations?

Animations that automatically animate property changes (e.g., AnimatedContainer).

33. What are explicit animations?

Animations controlled manually with AnimationController and Tween.

34. What is a Ticker?

A ticker calls a callback on each animation frame, driving animations.

35. Explain Hero animations.

They animate shared elements between two routes for smooth transitions.

36. What is LayoutBuilder used for?

It builds widgets based on parent constraints.

37. Explain MediaQuery.

It provides device information like screen size, orientation, and text scaling.

38. What is a RepaintBoundary?

Prevents unnecessary repaints by isolating the subtree for performance.

39. Explain Global vs Local Context.

Global context is accessible anywhere; local context is specific to a widget’s build method.

40. What are Performance Optimization Techniques in Flutter?

Use const widgets, avoid rebuilding, use RepaintBoundary, and profile with DevTools.

41. What are Platform Channels?

They enable communication between Dart and native Android/iOS code.

42. What is Tree Shaking?

Removes unused code during build to reduce app size.

43. What are Flutter build modes?

Debug, Profile, and Release.

44. What is the Flutter Engine?

It handles rendering, text layout, and platform communication using Skia.

45. Explain StatelessWidget and its lifecycle.

It is immutable and only has a build() method.

46. Explain StatefulWidget and its lifecycle.

It maintains a mutable state via createState(), initState(), build(), dispose().

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

const is compile-time constant; final is runtime constant.

48. What is late keyword in Dart?

Allows late initialization of non-nullable variables.

49. What is a mixin?

A class that provides reusable code to other classes using ‘with’ keyword.

50. What is an abstract class?

A class that cannot be instantiated directly; used as a blueprint.

51. What is factory constructor?

Returns existing or custom instances instead of new objects.

52. What are extension methods?

They add new functionality to existing classes without inheritance.

53. Explain the cascade operator (..).

Allows multiple operations on the same object sequentially.

54. What is operator overloading?

Allows custom behavior for operators like + or ==.

55. What are Generics?

Enable type safety and code reusability for collections and classes.

56. What is late initialization error?

Thrown when accessing a late variable before initialization.

57. What are records in Dart 3?

A new way to group multiple values without creating a class.

58. What is pattern matching?

A Dart 3 feature for destructuring and matching object shapes.

59. What is a sealed class?

Restricts subclassing to the same library for controlled inheritance.

60. Explain difference between extends and implements.

extends inherits functionality; implements forces method overrides.

61. Explain difference between StreamBuilder and FutureBuilder.

StreamBuilder listens to multiple events; FutureBuilder handles one-time async results.

62. What are Keys used for?

To maintain widget identity during rebuilds.

63. Explain Flutter’s rendering pipeline.

Layout → Paint → Compose → Rasterize.

64. What is Flutter DevTools used for?

Performance profiling, widget inspection, and memory tracking.

65. Explain widget rebuilding.

Occurs when state changes and Flutter re-runs the build() method.

66. What are immutable widgets?

Widgets whose properties cannot change once built.

67. What is ValueNotifier?

A class that holds a single value and notifies listeners when it changes.

68. What is ValueListenableBuilder?

A widget that rebuilds when ValueNotifier’s value changes.

69. What is dependency injection?

Providing required objects from external sources rather than creating them inside.

70. What is JSON serialization?

Converting Dart objects to JSON and vice versa.

71. Explain difference between Navigator 1.0 and 2.0.

Navigator 1.0 is imperative; 2.0 is declarative using Router API.

72. What is declarative UI?

UI is rebuilt entirely when state changes, ensuring predictable rendering.

73. What is composition vs inheritance?

Composition uses widgets within widgets; inheritance extends base classes.

74. What is asynchronous programming?

Programming where tasks execute independently without blocking the main thread.

75. What is StreamController?

Used to manage stream input, output, and state.

76. Explain Zone in Dart.

An execution context for async operations to handle logging or error tracking.

77. Difference between microtasks and event queue.

Microtasks have higher priority than event queue tasks.

78. What is a closure?

A function that captures and remembers variables from its scope.

79. What are const constructors?

Create compile-time constant objects.

80. Explain const object vs const reference.

Const object is immutable; const reference points to const instance.

81. Explain difference between dynamic, Object?, and var.

dynamic skips type checks; Object? is base class; var infers type.

82. What is type inference?

Dart automatically deduces variable types.

83. What are enums with values?

Enums that hold fields and methods for more complex logic.

84. Explain sound null safety.

Prevents null errors through compile-time enforcement.

85. Explain what a BuildContext is not.

It’s not the widget itself, nor global—it’s a handle to widget location.

86. Explain how Flutter handles UI updates.

Rebuilds the widget tree reactively when state or data changes.

87. Explain render tree vs widget tree.

Widget tree defines configuration; render tree handles layout and painting.

88. Explain what happens when setState() is called.

Marks widget as dirty → triggers rebuild → updates UI.

89. How do you reduce rebuild frequency?

Use const, Keys, memoization, or ValueListenableBuilder.

90. How do you persist data in Flutter?

Using SharedPreferences, Hive, SQLite, or secure storage.

91. Explain Flutter plugin architecture.

Plugins connect Dart code to native Android/iOS APIs using platform channels.

92. What is async gap in build()?

Build method is synchronous; async operations must complete before build.

93. Explain difference between StatelessWidget rebuilds and StatefulWidget rebuilds.

StatelessWidget rebuilds fully; StatefulWidget reuses State object.

94. Explain widget immutability.

Widgets are immutable; updates trigger rebuilds rather than mutation.

95. Explain what Skia does in Flutter.

Handles rendering and drawing graphics efficiently.

96. Explain the difference between paint() and layout() phases.

Layout defines size and position; paint draws pixels on screen.

97. Explain why Flutter is cross-platform.

It compiles Dart to native ARM code for each platform using its own engine.

98. What is hot reload limitation?

Cannot change class hierarchy or generic type parameters dynamically.