Dart Interview Questions and Answers

Q: What is Dart, and why is it used in Flutter?

Dart is a client-optimized programming language developed by Google for building fast apps on any platform. It's used in Flutter because it supports both Ahead-of-Time (AOT) and Just-in-Time (JIT) compilation, which allows fast development and smooth performance.

Q: What is the difference between var, final, and const in Dart?

`var` allows reassignment; `final` can be assigned only once but determined at runtime; `const` is compile-time constant.

Q: What is a Future in Dart?

A Future represents a potential value or error that will be available at some time in the future. It's used for asynchronous programming.

Q: Explain the difference between List, Set, and Map in Dart.

List: Ordered collection of elements. Set: Unordered collection of unique elements. Map: Collection of key-value pairs.

Q: What are mixins in Dart?

Mixins are a way of reusing code in multiple class hierarchies. They allow a class to include methods and properties from another class without inheritance using the `with` keyword.

Q: What is a factory constructor?

A factory constructor is used when you need more control over instance creation. It can return an existing instance or a new one.

Q: What is the difference between abstract classes and interfaces in Dart?

An abstract class can have method implementations and cannot be instantiated, while an interface is implemented implicitly by any class. Dart allows any class to act as an interface.

Q: Explain async and await in Dart.

`async` marks a function as asynchronous, and `await` pauses execution until the Future completes, allowing for cleaner asynchronous code.

Q: How do you handle exceptions in Dart?

Use try-catch blocks to handle exceptions. Example: try { int result = 10 ~/ 0; } catch (e) { print('Error: $e'); }

Q: What is the difference between == and identical() in Dart?

`==` checks for equality based on the `==` operator override, while `identical()` checks whether two references point to the same object in memory.