

## Q1. What are the primary design goals of Dart?

**A:** Dart was designed with specific priorities around performance, safety, or developer productivity depending on its domain. These goals shape its syntax, type system, and standard library choices.

## Q2. How does Dart handle type checking: static or dynamic?

**A:** Dart uses its type system to catch errors at compile time (static) or at runtime (dynamic). This affects refactoring safety, tooling support, and the kinds of bugs that reach production.

## Q3. How does Dart manage memory?

**A:** Dart uses one of three approaches: manual allocation/deallocation, a garbage collector that periodically reclaims unreachable objects, or automatic reference counting. Each has different latency and throughput tradeoffs.

## Q4. What is Dart's concurrency model?

**A:** Dart supports concurrency through threads, coroutines, `async/await`, or an actor model. The choice determines how I/O-bound and CPU-bound workloads are scaled and how shared state is safely accessed.

## Q5. How are packages and dependencies managed in Dart?

**A:** Dart uses a package manager and a manifest file to declare dependencies and version constraints. A lock file pins exact versions to ensure reproducible builds across environments.

## Q6. How does Dart implement object-oriented or functional programming?

**A:** Dart supports classes and inheritance for OOP, or first-class functions and immutability for FP, or both. Many modern Dart programs blend both paradigms depending on the problem.

## Q7. How does Dart handle errors?

**A:** Dart surfaces errors via exceptions, `Result/Either` types, or error return values. The choice impacts whether errors are checked at compile time and how calling code is structured.

## Q8. What testing tools are commonly used in Dart?

**A:** Dart has a standard or widely adopted test runner and assertion library. Tests are typically organized into unit, integration, and end-to-end layers with coverage reporting built in or via plugins.

## Q9. What makes Dart well-suited for its target domain?

**A:** Dart excels in its domain due to a combination of performance characteristics, ecosystem maturity, language ergonomics, and community support that makes common tasks simple.

## Q10. What are common performance pitfalls in Dart?

**A:** Typical performance issues in Dart include excessive allocations, blocking the main thread or event loop,  $O(n^2)$  data structures, and missing caching. Profiling and benchmarking tools are available to diagnose them.