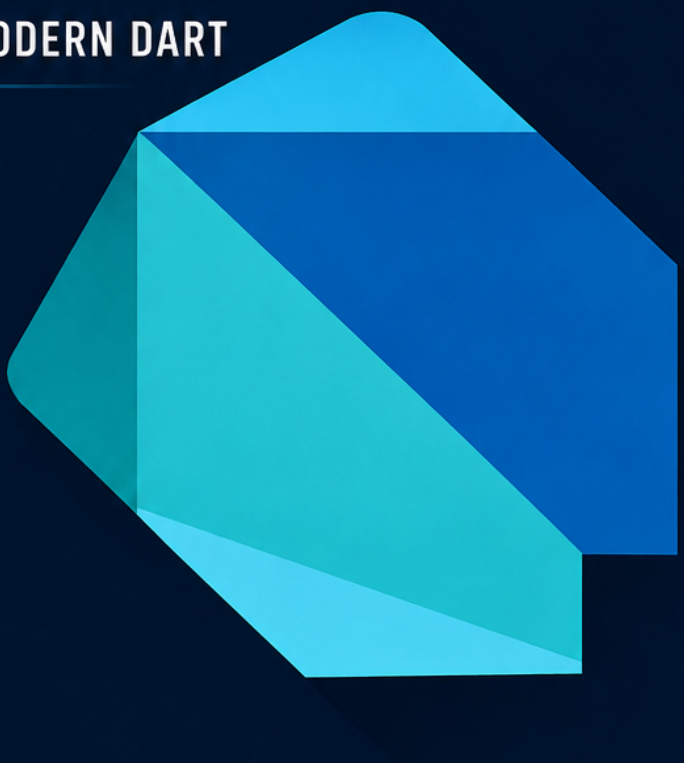
MASTERING DART PROGRAMMING

FROM FUNDAMENTALS TO ADVANCED PATTERNS IN MODERN DART

- DART 3.X & NULL SAFETY
- RECORDS & PATTERNS
- ASYNCHRONOUS PROGRAMMING
- ISOLATES & CONCURRENCY
- METAPROGRAMMING & MIXINS
- WEB, SERVER, CLI & FLUTTER



JAMES LOGAN

Mastering Dart Programming

From Fundamentals to Advanced Patterns in Modern Dart

Steve Publications

This book is available at https://leanpub.com/masteringdartprogramming

This version was published on 2026-07-26

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

From Fundamentals to Advanced Patterns in Modern Dart

This book is a complete guide to the Dart programming language, taking you from your first line of code through advanced patterns used in professional software development. Whether you are new to programming or an experienced developer exploring Dart for Flutter, web, server, or CLI applications, this book provides thorough explanations, production-quality examples, and deep insight into how Dart works under the hood. The content reflects modern Dart (3.x) with sound null safety enforced, covering records, patterns, isolates, asynchronous programming, metaprogramming, and interoperability with other languages.

Introduction: Why Dart Matters

In October 2011, Google engineers Lars Bak and Kasper Lund walked onto the stage at a GOTO conference in Aarhus, Denmark, and announced something that would become one of the most controversial decisions in modern web development. They had built a new programming language called Dart, designed to replace JavaScript for large-scale web applications.

The room did not react well. Neither did the internet.

The criticism was fierce and immediate. Developers saw Dart as yet another corporate attempt to fracture the open web. A leaked internal memo describing the project's ambition to "replace JavaScript as the lingua franca of Web development" only fueled the fire. Blog posts declared Dart dead on arrival. Conference talks mocked it. The language seemed destined for the graveyard of well-intentioned failures alongside ActionScript, CoffeeScript, and a dozen others that tried and failed to unseat JavaScript.

But here is what happened next, and it is the story that matters: Dart did what good software does when the market tells you that your initial strategy is wrong. It adapted.

Instead of trying to kill JavaScript, Dart learned to compile to it. Instead of fighting the browser ecosystem, Dart joined it. And then, almost quietly, something remarkable happened. Google built Flutter, a UI framework that needed a language fast enough for smooth animations, expressive enough for complex widget trees, and safe enough for millions of users. Dart fit that need perfectly.

By 2024, Flutter had become one of the most widely adopted cross-platform frameworks in the world, and Dart rode along with it, gaining legitimacy not through marketing but through millions of developers shipping real applications to real users. More importantly, Dart proved itself outside of Flutter as well: server-side services handling thousands of requests per second, command-line tools distributed across platforms, web applications compiled to both JavaScript and WebAssembly.

The language that was supposed to replace JavaScript instead found its own identity. It became something JavaScript never could be: a language where null

pointer exceptions are impossible by design, where the type system catches bugs before they reach production, where async code reads like synchronous code, and where a single compilation target can serve both development-time hot reload and production-grade native performance.

This book exists because Dart has earned serious attention. It is no longer a curiosity or a Flutter accessory. It is a mature, thoughtfully designed language whose features work together in ways that make large-scale software development genuinely easier. The goal here is not to convince you that Dart is the best language for everything. No language is. The goal is to give you a deep, practical understanding of Dart so that you can use it effectively wherever it fits your needs, whether that is building Flutter applications, writing server-side services, creating CLI tools, or any combination thereof.

Throughout this book, we will build a single application from scratch and evolve it as we learn new concepts. You will see how individual language features fit into real projects, not just isolated code snippets. By the end, you will not just know Dart syntax; you will understand how to design, structure, and optimize Dart applications the way experienced Dart developers do.

Let us begin with the ecosystem that makes all of this possible.

What This Book Covers

This book is structured as a progressive journey from beginner to expert-level Dart programming. The early chapters establish foundations: installation, tooling, basic syntax, variables, types, control flow, functions, collections, and object-oriented programming. Once those fundamentals are solid, the book moves into advanced territory: null safety, generics, records, patterns, extensions, mixins, enums, asynchronous programming with Futures and Streams, concurrency with isolates, error handling, annotations, library design, testing, performance optimization, code generation, and interoperability with C and JavaScript.

Every concept is explained with attention to the underlying principles and design rationale. You will learn not only how to write Dart code but why Dart works the way it does, what common pitfalls to avoid, and what patterns professional Dart developers use in real-world projects. All code examples are complete, compilable, and annotated to explain how and why they work.

Who This Book Is For

This book assumes you already know how to program in at least one other language. You should be comfortable with basic concepts like variables, loops, conditionals, functions, and objects. If those concepts are new to you, consider starting with a general programming introduction before diving in here. No prior Dart experience is required, and no Flutter knowledge is assumed, though the Dart skills you acquire will serve you well if you later explore Flutter development.

How to Use This Book

Read the book sequentially from beginning to end. Each chapter builds on concepts introduced earlier, and jumping ahead too quickly can leave gaps in your understanding. Take time to study the code examples, type them out by hand if that helps, and experiment with variations. The best way to learn a language is to write code in it, so use the examples as starting points for your own exploration.

All code examples target modern Dart with sound null safety enabled, which has been mandatory since Dart 3 (released May 2023) [3]. If you encounter older Dart code elsewhere that does not use null safety, treat it as legacy and understand that this book teaches the current, recommended way to write Dart.

The Philosophy Behind Dart's Design

Before diving into syntax, it is worth understanding the philosophy that shaped Dart. The language was designed with several core goals in mind:

- **Structured yet flexible**: Dart scales from small scripts to large enterprise applications, supporting optional type annotations for lightweight development and full static typing when you need rigor [4].
- **Familiar and natural**: The syntax draws from C, Java, and JavaScript traditions so that most developers feel at home quickly.
- **High performance**: Dart supports both JIT compilation for fast development cycles (including hot reload in Flutter) and AOT compilation for production deployments with native-speed execution [2].

- **Modern by default**: Features like sound null safety, pattern matching, records, and first-class async support are not afterthoughts but integral parts of the language.

These design goals are reflected throughout the book. When we encounter a particular syntax choice or language behavior, we will often trace it back to one of these principles, because understanding the why makes the how easier to remember and apply.

A Note on Versions and Features

Dart follows semantic versioning with stable releases approximately every three months, beta releases roughly monthly, and development builds twice weekly [5]. Language features are introduced through a language versioning system: each package specifies a minimum SDK version in its pubspec.yaml file, and the compiler enables or disables features based on that constraint. This allows Dart to evolve rapidly without breaking existing code.

This book targets Dart 3.x with a focus on features available from Dart 3.0 onward, which includes records, patterns, class modifiers, switch expressions, and enforced sound null safety [6]. Where a feature was introduced in a specific minor version (for example, digit separators in 3.6 or null-aware collection elements in 3.8), the text notes that requirement.

Your First Glimpse of Dart

To give you a taste of what is ahead, here is a complete Dart program that demonstrates several core concepts:

```dart
// A simple command-line application demonstrating basic Dart syntax
void main() {
  // Variables with type inference
  var name = 'Dart';
  final version = '3.12';

  // A function with named parameters
  printGreeting(target: name, edition: version);

  // Working with collections
  final features = ['null safety', 'patterns', 'records'];
  for (final feature in features) {
    print('  - $feature');
  }
}

// Function with required positional and named parameters
void printGreeting({required String target, String edition = 'latest'}) {
  print('Hello from $target ($edition)!');
  print('Key features:');
}
```

This short program already shows type inference with `var` and `final`, string interpolation, named parameters, collection iteration, and the standard entry point pattern. Over the coming chapters, we will unpack every one of these concepts in detail and then move far beyond them.

Let us begin.

Chapter 1: The Dart Ecosystem

Dart does not exist in a vacuum. It is part of a broader ecosystem that includes compilers, runtimes, tools, libraries, and frameworks. Understanding this ecosystem is essential for making informed decisions about how to develop, test, deploy, and maintain Dart applications. This chapter provides an overview of where Dart fits in the modern development landscape.

Origins and Design Philosophy

Dart was created by Lars Bak and Kasper Lund at Google, two engineers with deep expertise in virtual machines and language design. Bak had previously co-created the V8 JavaScript engine used in Chrome, and both he and Lund had worked on the HotSpot JVM. Their combined experience informed many of Dart's architectural choices [7].

The language was publicly announced on October 10, 2011, at the GOTO conference. In his announcement blog post, Lars Bak outlined Dart's design goals:

> "Create a structured yet flexible language for Web programming. Make Dart feel familiar and natural to programmers and thus easy to learn. Support high performance on all devices and servers." [8]

At the time, JavaScript was the only viable language for client-side web development, but it had significant limitations for large-scale applications: inconsistent typing, unpredictable performance across browsers, and a lack of mature tooling. Dart aimed to address these issues while remaining accessible to developers already comfortable with C-style syntax.

The initial reaction was polarized. Some praised Dart's ambition; others criticized it as unnecessary or even hostile to the open web. A leaked internal Google memo describing Dart (then called "Dash") as intended to "replace JavaScript as the lingua franca of Web development" fueled controversy [9]. Eventually, the approach shifted. Rather than trying to replace JavaScript entirely, Dart learned to compile to it and coexist with the existing ecosystem.

This pragmatic evolution continued. In 2018, Dart 2.0 introduced a sound type system. In 2021, Dart 2.12 brought sound null safety, the largest language change since 2.0. In 2023, Dart 3.0 added records, patterns, and class modifiers while making null safety mandatory for all code [6]. Each major release has strengthened Dart's position as a serious, production-ready language.

Major Version Milestones

The following table summarizes key releases in Dart's history:

Version	Date	Key Features
1.0	November 2013	First stable release; ECMA standardization began
1.12	August 2015	Language stabilized; major bug fixes
2.0	February 2018	Sound type system; breaking changes from 1.x
2.3	May 2019	Spread operator, collection if/for
2.7	December 2019	Extension methods
2.12	March 2021	Sound null safety (opt-in); FFI stable
2.15	December 2021	Constructor tear-offs (function pointers)
2.17	May 2022	Enhanced enums with members and constructors
3.0	May 10, 2023	Records, patterns, class modifiers, switch expressions; null safety mandatory
3.1	August 2023	Performance improvements; WebAssembly bug fixes
3.2	November 2023	Type promotion on private final fields; if-case fixes; build hooks (experimental)
3.3	February 2024	Extension types; abstract getters promotable; dart:js_interop improvements
3.4	May 2024	Type analysis improvements across conditionals and switch expressions
3.5	August 2024	No new language features; type inference refinements

Version	Date	Key Features
3.6	December 2024	Digit separator underscores in number literals; pub workspaces
3.7	February 2025	Wildcard variables (_); dart format rewrite tied to language version; macros development halted
3.8	May 2025	Null-aware collection elements (?expr)
3.9	August 2025	Null safety assumptions in type promotion; Flutter upper version bound respected
3.10	November 2025	Dot shorthands; build hooks stable; analyzer plugin system; removed IA32 support
3.11	February 2026	Windows Unix domain sockets; workspace globbing; dart pub cache gc
3.12	May 2026	Private named parameters via initializing formals

This rapid release cadence (approximately every three months) reflects Dart's commitment to continuous improvement while maintaining backward compatibility through language versioning. Each release is announced on the official Dart Blog, and the complete language evolution timeline is documented at dart.dev/resources/language/evolution.

Where Dart Runs: Compilers and Runtimes

One of Dart's defining strengths is its multi-target compilation model. The same Dart source code can run on different platforms through different compilation paths. Understanding these paths is crucial for choosing the right development workflow and deployment strategy.

The Dart VM and JIT Compilation

The Dart Virtual Machine (VM) is the primary runtime for developing and running Dart applications on native platforms. It features a just-in-time (JIT)

compiler that translates Dart code into machine code at runtime, enabling fast startup and incremental recompilation [2].

JIT compilation is particularly valuable during development. It allows features like hot reload in Flutter, where changes to source code are reflected in the running application almost instantly without losing application state. The JIT compiler also supports rich debugging, profiling, and live metrics collection that power the Dart DevTools suite.

When you run a Dart program using the `dart run` command during development, the Dart VM's JIT compiler is doing the work. You do not need to compile your code ahead of time.

AOT Compilation for Production

For production deployments on native platforms (mobile, desktop, server), Dart uses ahead-of-time (AOT) compilation to produce optimized machine code. The AOT compiler translates all Dart code, including dependencies, into a single executable binary before the application runs [2].

AOT-compiled code has several advantages:

- **Fast startup**: No runtime compilation overhead means the application starts immediately.
- **Consistent performance**: There is no "warm-up" period where the JIT compiler optimizes hot paths. Performance is predictable from the first execution.
- **Smaller memory footprint**: The resulting binary does not need to carry the JIT compiler.
- **No runtime reflection overhead**: In many cases, AOT compilation eliminates the need for runtime type information.

The `dart compile exe` command produces self-contained executables for Windows, Linux, or macOS. For mobile platforms like Android and iOS (via Flutter), the AOT compiler generates platform-specific machine code that is packaged into the application bundle.

Web Compilation: JavaScript and WebAssembly

For web applications, Dart offers two compilation targets.

The traditional target is JavaScript via the `dart2js` compiler (now accessed through `dart compile js`). This produces optimized JavaScript bundles that run in any modern browser. The output can be minified and tree-shaken for production deployments. For development on the web, Dart also provides the Dart Development Compiler (DDC), which prioritizes fast incremental compilation and accurate error messages over output size [2].

A newer target is WebAssembly via `dart compile wasm`. WebAssembly offers near-native performance in browsers and represents the future of high-performance web applications. Dart's WebAssembly support is still evolving but provides a promising path for computationally intensive web applications that need to go beyond what JavaScript alone can deliver.

Platform Summary

The following table summarizes the compilation targets available in the Dart ecosystem:

Target Platform	Development Mode	Production Mode
Native (CLI, server, desktop)	Dart VM with JIT	AOT-compiled executable
Mobile (via Flutter)	Dart VM with JIT + hot reload	AOT-compiled machine code
Web	DDC (fast incremental) or dart2js	dart2js (optimized JS) or WebAssembly

Flutter, Web, Server, and CLI

Dart is used in four primary application domains. While Flutter has brought Dart the most visibility, the language is fully capable outside of UI development.

Flutter Applications

Flutter is Google's UI toolkit for building natively compiled applications for mobile, web, desktop, and embedded devices from a single codebase. Dart

is the sole programming language for Flutter development. The combination of Dart's JIT compilation (for hot reload during development) and AOT compilation (for production performance) makes it an excellent fit for Flutter's workflow [2].

Flutter apps are written entirely in Dart. The framework provides its own rendering engine, widget system, and tooling. If you are learning Dart primarily to build Flutter applications, this book will give you the language foundation you need. However, understanding Dart as a standalone language is valuable even if your primary focus is Flutter.

Web Applications

Dart can be used to build web applications independently of Flutter. You might choose this path if you want a strongly typed alternative to JavaScript for backend-heavy web applications or if you are building a single-page application that does not require Flutter's widget ecosystem.

Web compilation produces JavaScript (or WebAssembly) that runs in the browser. Dart's tooling, including the `dart create` command, can scaffold web projects with appropriate configuration. The same language features available in native Dart are available for web development.

Server-Side Applications

Dart is a capable server-side language. Its event-loop architecture, async-first design, and garbage-collected runtime make it well suited for I/O-bound server applications such as REST APIs, microservices, and real-time backends.

The `dart:io` library provides networking, file system access, HTTP server capabilities, and more. Frameworks like Shelf and Angel provide higher-level abstractions for building web servers. Dart's AOT compilation produces efficient server binaries with fast startup times.

Command-Line Tools

Dart is increasingly popular for building command-line tools. The `dart create --template=cli` command scaffolds a CLI application with argument parsing, help text generation, and structured logging. Because Dart can

compile to self-contained executables on all major platforms, distributing Dart-based CLI tools is straightforward.

Many developers who first encounter Dart through Flutter discover that it is also an excellent choice for build tools, data processing scripts, automation utilities, and developer tooling.

The Dart Toolchain Overview

The Dart SDK includes a comprehensive set of command-line tools that cover the entire development lifecycle. These tools are accessed through the `dart` command and are designed to work together seamlessly.

Core Commands

- **`dart create`**: Generates new projects with templates for CLI apps, web apps, server packages, and more.
- **`dart run`**: Runs a Dart program using the VM's JIT compiler. Ideal for development.
- **`dart compile`**: Compiles Dart code to executables (`exe`), JavaScript (`js`), WebAssembly (`wasm`), or AOT snapshots.
- **`dart test`**: Runs tests written with the `test` package.
- **`dart analyze`**: Performs static analysis on your code, catching errors and style violations.
- **`dart format`**: Formats code according to the official Dart style guide.
- **`dart pub`**: Manages packages and dependencies (get, add, remove, upgrade, publish).
- **`dart doc`**: Generates API documentation from source code comments.
- **`dart fix`**: Automatically applies recommended code fixes.

Dart DevTools

Dart DevTools is a suite of debugging and profiling tools distributed with the SDK. It provides views for:

- **Debugger**: Step through code, inspect variables, set breakpoints.
- **Logging**: View application logs in real time.

- **CPU Profiler**: Identify performance bottlenecks with flame graphs.
- **Memory Profiler**: Detect memory leaks and analyze heap usage.
- **Network Inspector**: Monitor HTTP requests and responses.
- **Performance**: Analyze frame rendering and timeline events (for Flutter apps).

DevTools is launched automatically when you run `dart run --observe` for command-line applications or through IDE integrations for Flutter development [10].

Release Channels

Dart uses three release channels to balance stability with innovation:

- **Stable**: Production-ready releases published approximately every three months. Use this channel for production applications.
- **Beta**: Preview releases published roughly monthly. Features are mostly complete but may still have bugs. Good for testing new features before they reach stable.
- **Dev**: Cutting-edge builds published about twice weekly. Features are experimental and may change or be removed. Intended for language contributors and early adopters who want to test upcoming changes [5].

Most developers should use the stable channel. The beta and dev channels are valuable for staying informed about Dart's evolution but carry the risk of instability.

Summary

Dart is a mature, multi-platform programming language with a rich ecosystem. Its origins as a web-focused alternative to JavaScript have evolved into a broader role as the foundation for Flutter and a capable standalone language for server, CLI, and desktop development. The dual JIT/AOT compilation model supports both fast development cycles and production-grade performance. Unified tooling through the `dart` command and DevTools provides a cohesive developer experience across all platforms.

In the next chapter, we will set up your Dart development environment and write your first programs.

Chapter 2: Getting Started with Dart

Before writing Dart code, you need a working development environment. This chapter walks through installing the Dart SDK, choosing a development environment, creating your first project, and understanding the basic workflow for running and testing Dart programs.

Installing Dart on Your System

The Dart Software Development Kit (SDK) contains everything needed to develop Dart applications: the Dart VM, compilers, command-line tools, and core libraries. Installation methods vary by operating system.

Using Package Managers (Recommended)

The easiest way to install Dart is through your operating system's package manager. These methods make updates straightforward and integrate well with existing development workflows.

macOS with Homebrew:

```
brew install dart
```

Windows with winget:

```
winget install --id Dart.DartSDK -e
```

Windows with Chocolatey:

```
choco install dart-sdk
```

Linux (Debian/Ubuntu):

```
sudo apt-get update
sudo apt-get install dart
```

After installation, verify that Dart is working:

```
dart --version
```

This should output the installed SDK version, such as `Dart SDK version: 3.12.2`.

Alternative Installation Methods

If package managers are not available or you need a specific version, consider these alternatives:

- **ZIP Archive**: Download a specific SDK version from the Dart SDK archive at https://dart.dev/get-dart/archive. Extract the archive and add the `bin` directory to your system PATH.
- **Docker**: Use official Dart Docker images for containerized development. This is useful for CI/CD pipelines or ensuring consistent environments across teams.
- **Build from Source**: For language contributors who want to work on Dart itself, the SDK can be built from source using the instructions at https://github.com/dart-lang/sdk.

Flutter Users

If you are primarily developing Flutter applications, installing the Flutter SDK is sufficient. The Flutter SDK includes the Dart SDK in its `bin` directory. When you run `flutter --version`, it reports both the Flutter and Dart versions. You do not need a separate Dart installation unless you want to work on non-Flutter Dart projects independently [5].

Development Environments and Editors

Dart has excellent support across multiple editors and IDEs. The choice depends on your preferences, existing workflow, and the type of applications you plan to build.

Visual Studio Code

VS Code with the Dart extension is one of the most popular development environments for Dart. The official Dart extension (published by the Dart team) provides:

- IntelliSense code completion
- Syntax highlighting
- Inline error reporting from the Dart analyzer
- Debugging with breakpoints and variable inspection
- Formatting with `dart format`
- Test running and coverage
- Package management integration

Install the extension from the VS Code Marketplace, then open a Dart project folder. The extension automatically detects Dart files and activates its features.

IntelliJ IDEA and Android Studio

JetBrains' IDEs support Dart through the official Dart plugin. Both IntelliJ IDEA (Ultimate edition) and Android Studio include Dart support. Features are similar to VS Code but integrated into the JetBrains ecosystem:

- Advanced code navigation and refactoring
- Run configurations for different platforms
- Integrated terminal and version control
- Profiling tools

The Dart plugin is also available for other JetBrains IDEs like PyCharm and WebStorm.

Command-Line Development

For CLI applications, server projects, or when working in constrained environments, Dart can be developed entirely from the command line. The `dart` tool provides all necessary functionality: running, compiling, testing, analyzing, and formatting. Many experienced Dart developers do most of their work in a terminal with a text editor like Neovim or Emacs.

DartPad

For quick experiments, learning exercises, or sharing code snippets, DartPad (https://dartpad.dev) is an online Dart playground. It runs in the browser with no installation required and provides basic editing, running, and error reporting. DartPad is excellent for trying out language features but not suitable for serious development work.

Creating Your First Project

Dart projects are organized as packages. A package is a directory containing Dart code along with a `pubspec.yaml` file that describes the project's metadata and dependencies. The `dart create` command generates new projects with appropriate structure.

CLI Application Template

To create a command-line application:

```
dart create --template=console-simple my_first_app
cd my_first_app
```

This creates a minimal project with the following structure:

```
my_first_app/
├── bin/
│   └── my_first_app.dart    # Entry point
├── lib/                     # Library code (empty for simple template)
├── test/                    # Test files
├── pubspec.yaml             # Package configuration
└── README.md
```

The entry point is `bin/my_first_app.dart`. Here is what the default code looks like:

```dart
void main(List<String> arguments) {
  print('Hello world: ${arguments.join(', ')}!');
}
```

Run the application:

```
dart run
```

Output:

```
Hello world: !
```

The `main` function receives command-line arguments as a list of strings. Pass arguments when running:

```
dart run -- Dart Rocks
```

Output:

```
Hello world: Dart, Rocks!
```

Note the double dash (`--`). It separates options for the `dart` command from arguments passed to your program. Without it, Dart would interpret `Dart` and `Rocks` as its own flags.

Web Application Template

To create a web application:

```
dart create --template=web-server my_web_app
cd my_web_app
```

This generates a project with an HTTP server serving HTML that includes compiled Dart code. Run it with:

```
dart run
```

Then open the provided URL in your browser.

Package Template

To create a library package (code intended to be used by other projects):

```
dart create --template=package-simple my_library
cd my_library
```

This creates a project focused on library code under `lib/` with tests under `test/`. Library packages do not have a `bin/` directory because they are not meant to run as standalone applications.

Running, Testing, and Publishing Basics

Running Dart Code

There are several ways to execute Dart code:

- **`dart run bin/app.dart`**: Runs the specified file using the Dart VM. This is the primary way to run during development.
- **`dart run`**: Runs the default entry point defined in `pubspec.yaml` (typically `bin/<package_name>.dart`).
- **`dart compile exe bin/app.dart -o myapp`**: Compiles to a native executable, then run with `./myapp` (or `myapp.exe` on Windows).

For development, always prefer `dart run` over compiling. The JIT compiler provides faster iteration and better error messages. Compile only when you need a distributable binary.

Analyzing Your Code

The Dart analyzer performs static analysis to catch errors, warnings, and style issues before runtime:

```
dart analyze
```

Example output:

```
Analyzing my_first_app...
No issues found!
```

If there are problems:

```
Analyzing my_first_app...

  error - bin/my_first_app.dart:3:5 - The variable 'x' must be assigned before
   ↪  it can be used.
          Try initializing the variable or assigning it before use. -
          ↪  not_assigned_potentially_used_variables
   info  - bin/my_first_app.dart:1:1 - The imported package 'unused' is not
   ↪  used.
          Try removing unused imports. - unused_import

2 issues found.
```

Issues are categorized as:

- **error**: Code will not run correctly. Fix required.
- **warning**: Potential problem or deprecated usage.
- **info**: Style suggestion or best practice recommendation.

Run `dart analyze` regularly during development and in your CI pipeline to maintain code quality.

Formatting Your Code

The Dart community uses a single, opinionated code style enforced by the `dart format` tool:

```
dart format .
```

This reformats all Dart files in the current directory tree according to the official style guide. There are no configuration options. The philosophy is that consistent formatting eliminates debates about style and lets developers focus on logic.

Example of what `dart format` does:

```
// Before formatting
functionWithALongName(
    argumentOne,argumentTwo,
    argumentThree);

// After formatting
functionWithALongName(argumentOne, argumentTwo, argumentThree);
```

Integrate `dart format` into your editor or run it before committing code.

Testing Your Code

Dart includes a built-in testing framework accessed through `dart test`. Tests are written using the `test` package and placed in the `test/` directory. Here is a simple example:

```
// test/my_first_app_test.dart
import 'package:test/test.dart';

void main() {
  test('adds numbers correctly', () {
    expect(2 + 2, equals(4));
  });
}
```

Run all tests:

```
dart test
```

Output:

```
00:00 +1: All tests passed!
```

The `test` command discovers and runs all files matching `test/**/*.dart`. More on testing in Chapter 13.

Publishing Packages

If you create a library package that others might find useful, you can publish it to pub.dev, the official Dart package repository. Before publishing:

1. Ensure your `pubspec.yaml` includes required fields: name, version, description.
2. Run `dart pub publish --dry-run` to validate.
3. Fix any reported issues.
4. Run `dart pub publish` to publish for real.

Package names must be unique across pub.dev and use only lowercase letters, digits, and underscores [11]. Versioning follows semantic versioning (semver): MAJOR.MINOR.PATCH.

Understanding the pubspec.yaml File

Every Dart package has a `pubspec.yaml` file that defines its metadata and dependencies. Here is an annotated example:

```yaml
name: my_first_app
description: A simple command-line application demonstrating Dart basics.
version: 1.0.0

# The environment constraint specifies which Dart SDK versions this package
↪  supports.
environment:
  sdk: '^3.12.0'

# Dependencies required to run the application.
dependencies:
  http: ^1.2.0      # An HTTP client library from pub.dev

# Dependencies only needed during development (testing, tooling).
dev_dependencies:
  test: ^1.25.0
  lints: ^4.0.0     # Recommended lint rules

# Executables that users can run with `dart run <name>`
executables:
  my_first_app:
```

Key fields explained:

- **name**: The package identifier. Must be unique on pub.dev.
- **version**: Semantic version number. Required for published packages.
- **environment.sdk**: Specifies the minimum and maximum supported Dart SDK versions. The caret (^) allows compatible updates. `^3.12.0` means ">=3.12.0 and <4.0.0".
- **dependencies**: Libraries your code imports and uses at runtime.
- **dev_dependencies**: Libraries used only during development (tests, code generation, linting).
- **executables**: Maps executable names to entry-point scripts in `bin/`.

The `dart pub get` command downloads and installs all dependencies listed in `pubspec.yaml` and creates a `pubspec.lock` file that records the exact versions resolved. Commit the lock file to version control for reproducible builds.

Summary

You now have a working Dart development environment, understand how to create projects, run code, analyze it, format it, test it, and publish packages. The foundation is in place. In the next chapter, we dive into Dart's language fundamentals: syntax, variables, types, operators, and the basic building blocks of every Dart program.

Chapter 3: Language Fundamentals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Program Structure and Imports

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Entry Point: main()

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Libraries and Imports

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Comments and Documentation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Variables, Constants, and Finality

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Declaring Variables with var

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Explicit Type Annotations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Constants: final vs const

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Nullability: The Default Behavior

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Uninitialized Variables and late

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Core Data Types and Literals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Numbers: int and double

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Strings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Booleans

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Records

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Functions as Values

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Core Types Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

null, Object, dynamic, and void

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Introducing TaskFlow: A Running Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The First Version: Basic Syntax in Action

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Operators and Expressions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Arithmetic Operators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Comparison and Relational Operators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Logical Operators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Assignment Operators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Null-Aware Operators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Cascade Notation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Operator Precedence

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chapter 4: Control Flow and Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Conditional Logic and Switch Statements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

if and else

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Ternary Conditional Operator

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Switch Statements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

if-case Statements (Dart 3+)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Loops and Iteration Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

for Loops

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

for-in Loops

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

while and do-while Loops

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Breaking and Continuing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Functions as First-Class Citizens

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Declaring Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Arrow Syntax for Single-Expression Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Function Parameters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Anonymous Functions and Lambdas

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Closures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

TaskFlow v0.2: Functions and Control Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Tear-offs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Generators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chapter 5: Collections in Depth

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Lists: Ordered Collections and Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Creating Lists

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Accessing and Modifying Elements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Iterating Over Lists

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Transforming Lists with Higher-Order Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Unmodifiable Lists

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Choosing the Right Collection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Sets: Uniqueness and Set Algebra

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Creating Sets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Set Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Maps: Key-Value Associations and Transformations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Creating Maps

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Accessing and Modifying Maps

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Transforming Maps

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Collection Literals, For-Loops, and Ifs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Spread Operators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Collection Ifs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Collection For-Loops

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Null-Aware Elements (Dart 3.8+)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

TaskFlow v0.3: Collections in Practice

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Common Pitfalls and Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chapter 6: Object-Oriented Programming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Classes, Objects, and Instances

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Defining a Class

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Instance Variables and Getters/Setters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Final Instance Variables

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Static Members

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Type Checking and Casting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Constructors and Initialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Default Constructor

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Generative Constructors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Named Constructors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Constant Constructors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Factory Constructors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Constructor Initialization Order

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Redirecting Constructors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Inheritance, Superclasses, and Polymorphism

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Extending Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Overriding Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Abstract Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Interfaces and implements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Polymorphism

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Encapsulation and Access Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Library Privacy with Underscore

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Public API Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

TaskFlow v0.4: Object-Oriented Refactoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Class Modifiers and Sealed Types (Dart 3+)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Class Modifier System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Four Access Modifiers Explained

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Modifier Comparison Table

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Combining Modifiers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Practical Example: API Design with Modifiers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Guidelines for Choosing Modifiers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chapter 7: Null Safety and Modern Dart

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Null Safety Model Explained

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Problem: Null Reference Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Solution: Non-Nullable by Default

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Soundness: What It Means

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Benefits of Null Safety

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Nullable Types and the Bang Operator

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Declaring Nullable Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Working with Nullable Values

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Type Promotion and Flow Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Null-Aware Operators and Coalescing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Null Coalescing Operator (??)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Null-Aware Access Operator (?.)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Null-Aware Assignment Operator (??=)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Operator Summary Table

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Late Initialization and Lazy Evaluation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The late Keyword

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Lazy Initialization Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

late final for Constructor Parameters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

TaskFlow v0.5: Null Safety in the Domain Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Common Pitfalls and Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chapter 8: Advanced Type System Features

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Generics: Type Parameters and Constraints

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Generic Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Generic Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Type Bounds with extends

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Generic Collection Literals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Reified Generics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Records and Pattern Matching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Records: Lightweight Data Structures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Pattern Matching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Enums with Members and Constructors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Basic Enums

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Enhanced Enums (Dart 2.17+)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Sealed Classes and Exhaustive Pattern Matching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Extensions and Mixins for Code Reuse

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Extension Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Mixins

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

TaskFlow v0.6: Sealed Classes, Patterns, and Records

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chapter 9: Asynchronous Programming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Event Loop and Microtask Queue

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Visualizing the Event Loop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Why Two Queues?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Futures and the Async/Await Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Creating Futures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Using await

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chaining Futures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Future API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Returning Futures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Error Handling in Async Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Futures vs Streams: When to Use Which

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Streams: Reactive Data Pipelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Consuming Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Creating Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Transforming Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Stream Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

TaskFlow v0.7: Async File Persistence and Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chapter 10: Concurrency with Isolates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Why Single-Threaded Is Not Enough

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Creating and Communicating Between Isolates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Isolate Memory Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Using Isolate.run for Short-Lived Tasks (Dart 3.3+)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Using Isolate.spawn for Long-Lived Workers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Message Passing Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Compute API and Worker Pools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The compute Function

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Worker Pool Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Shared Memory and Message Passing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

TaskFlow v0.8: Isolates for Batch Processing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Common Pitfalls and Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chapter 11: Error Handling and Annotations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Exception Types and the Try/Catch Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Exception Hierarchy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Try/Catch/Finally

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Rethrowing Exceptions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Designing Domain-Specific Exceptions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Simple Custom Exceptions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Exception Design Guidelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Assertions and Defensive Programming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The assert Statement

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Defensive Programming Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

TaskFlow v0.9: Domain-Specific Error Handling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Annotations and Metadata for Tooling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Built-in Annotations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Package:meta Annotations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Custom Annotations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chapter 12: Libraries, Packages, and Modules

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Library Directives and Visibility

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Every File Is a Library

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Importing Libraries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Controlling Visibility

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Library Privacy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Exporting Libraries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Package Structure and pubspec.yaml

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

What Is a Package?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Standard Directory Layout

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The pubspec.yaml File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Dependency Management and Versioning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Version Constraints

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Resolving Dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Dependency Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Dependency Conflicts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Publishing and Consuming Packages

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Preparing to Publish

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Publishing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Consuming Packages

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Package Naming Conventions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

TaskFlow v1.0: Refactoring into a Proper Package Structure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Before: Single-File Structure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

After: Professional Package Layout

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Main Library Export

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Model Files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Storage Abstraction Layer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The CLI Entry Point

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The pubspec.yaml

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Why This Structure Matters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chapter 13: Testing in Dart

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Test Package and Unit Testing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Setting Up Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Writing Your First Test

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Assertions with expect

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Grouping Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

setUp and tearDown

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Testing Asynchronous Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Mocking, Fakes, and Stubs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Types of Test Doubles

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Using Fakes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Using Mocktail for Mocking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Integration Testing Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Running Integration Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Testing with Real Dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Test Organization and Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Structure Large Test Suites

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Writing Good Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Running Tests with Coverage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

TaskFlow: Comprehensive Testing Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chapter 14: Performance and Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Understanding Dart's Runtime Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

JIT vs AOT Performance Characteristics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Web Compilation Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Common Performance Characteristics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Memory Management and Allocation Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Garbage Collection in Dart

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Allocation Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Memory Leak Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Profiling Tools and Flame Graphs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Dart DevTools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Using Timeline Events

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Interpreting Flame Graphs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Optimization Techniques and Anti-Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Effective Optimizations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Anti-Patterns to Avoid

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Deep Dive: The Dart Garbage Collector

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Generational Collection Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Promotion and Floating Garbage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Practical GC Optimization Tips

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Advanced Profiling Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Flame Graph Deep Dive

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Memory Profiling Workflow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Timeline Events for Custom Instrumentation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

TaskFlow v1.0: Performance Profiling and Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Scenario

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Problem 1: Slow Tag Filtering

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Problem 2: UI Freeze During Batch Analysis

Problem 3: Memory Leak from Uncancelled Subscriptions

Measuring the Impact

Optimization Principles Demonstrated

Real-World Optimization Case Studies

Case Study 1: Optimizing a Large List View

Case Study 2: Reducing Startup Time for a CLI Tool

Case Study 3: Optimizing Network Request Handling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Compilation Target Performance Comparison

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chapter 15: Interoperability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Foreign Function Interface (FFI)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

How FFI Works

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Basic FFI Usage: A Complete Working Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Working with C Data Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Structs and Unions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Function Pointers and Callbacks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Using ffigen for Binding Generation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Build Hooks for Native Assets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

FFI Case Study: Integrating a Cryptography Library

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

JavaScript Interop for Web Targets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The dart:js_interop Library: Complete Working Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Extension Types for Type Safety

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Interfacing with Third-Party JavaScript Libraries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

package:web for Browser APIs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Legacy JS Interop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Chapter 16: Metaprogramming and Build Tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Dart Build System Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

build_runner Overview

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

source_gen for Annotation-Based Generation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Popular Code Generators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Writing a Custom Builder

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Part Files and Generated Code Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

The Fate of Dart Macros

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Augmentations: The Alternative Direction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Practical Metaprogramming Today

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Compile-Time Constants and Mirrors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Compile-Time Constants

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Mirrors (Deprecated)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Conclusion: The Path Forward

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Key Takeaways from the Journey

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Emerging Features and Dart's Future

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Building Professional Dart Applications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Language Specifications and Standards

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Historical and Design Documents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Official Dart Documentation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Tools and Libraries Documentation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Runtime and Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Third-Party Packages

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.

Glossary of Terms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringdartprogramming.