LEARN DART LANGUAGE

IN 40 MINUTES
FOR BEGINNERS

With Hands-on Examples and Practical Projects

YUFENG CHEN

Learn Dart Language

in 40 Minutes for Beginners

with hands-on examples
and practical projects

Yufeng Chen

First published in 2024

by Dynotes Technology Corporation, a Canadian company

https://dynotes.com

ISBN: 978-1-0690323-0-0 (eBook)

Intended Audience

This book is tailored for beginners who are new to Dart but already have a foundational understanding of object-oriented programming (OOP) concepts from languages like C#, Java, Python, or similar. If you’re familiar with the basics of OOP—such as classes, objects, inheritance, and polymorphism—this book will help you quickly adapt those concepts to Dart, enabling you to become proficient in a new, versatile programming language.

Additionally, this book is a valuable resource for educators and students. Instructors teaching programming courses can use it as a structured guide to introduce Dart to their students, while learners can benefit from the clear explanations, practical examples, and step-by-step projects that reinforce their understanding of the language.

Whether you’re transitioning from another OOP language to Dart or seeking to expand your programming repertoire, this book provides the knowledge and tools you need to master Dart with confidence.

The Structure and Philosophy of the Book

This book is structured to facilitate a smooth and progressive learning experience, guiding you from the foundational aspects of Dart programming to more advanced concepts. The philosophy behind this organization is to ensure that every reader, regardless of their prior experience, can build a solid understanding of Dart by following a logical and incremental path.

Foundational Learning: The journey begins with the basics—setting up your environment and writing your first Dart program. These early chapters lay the groundwork, ensuring that you have the necessary tools and understanding to tackle more complex topics.

Incremental Complexity: As you progress, each chapter introduces new concepts while reinforcing the previous ones. This incremental approach helps you build confidence and competence in Dart, with each new topic adding a layer of depth to your knowledge.

Practical Application: The inclusion of practical projects throughout the book is key to the learning philosophy. These projects are not just theoretical exercises but real-world applications that demonstrate how Dart can be used to solve actual problems. By working on these projects, you solidify your understanding and gain hands-on experience that will be invaluable in your development journey.

Advanced Exploration: Once the foundational concepts are well-established, the book delves into more advanced topics such as object-oriented programming, error handling, and Dart's powerful features like null safety and mixins. These sections are designed to challenge you and expand your capabilities, preparing you for more sophisticated programming tasks.

Holistic Development: The book is not just about learning Dart in isolation. It also emphasizes best practices, efficient coding techniques, and problem-solving strategies that are applicable beyond Dart, making you a better programmer overall.

By the end of the book, you will have a comprehensive understanding of Dart, supported by practical experience and a solid grasp of advanced programming techniques. This structured approach ensures that you not only learn Dart but also develop the confidence and skills to apply it effectively in real-world scenarios.

Table of Contents

Introduction

What is Dart?

Dart is a versatile, open-source programming language developed by Google. It is optimized for building fast applications across various platforms such as mobile, web, desktop, and server. Dart is designed to be easy to learn and use, with a focus on productivity and high performance.

Brief History and Purpose

History

Dart was introduced by Google in 2011 as an alternative to JavaScript, primarily for web development. The language has since evolved to support a wide range of applications, from web to mobile, desktop, and backend services. Google uses Dart extensively for its Flutter framework, which enables developers to create natively compiled applications for mobile, web, and desktop from a single codebase.

Purpose

Dart was created with the following goals in mind:

Productivity: Dart offers features that help developers write and maintain code more efficiently, such as a strong type system, rich standard library, and powerful tooling.

Performance: Dart is designed for high performance, both in development (fast compile times) and in production (fast execution).

Portability: Dart can be compiled to native code or JavaScript, making it suitable for a wide range of platforms and environments.

Why Learn Dart?

Single Codebase: With Dart and Flutter, you can write code once and deploy it to multiple platforms, including iOS, Android, web, Windows, macOS, and Linux.

Strong Community and Ecosystem: Dart has a growing community and a rich ecosystem of libraries and packages, which can be found on pub.dev. This makes it easier to find solutions and get support.

Modern Features: Dart includes modern language features such as sound *null* safety, *async/await* for asynchronous programming, and a comprehensive standard library.

Google's Backing: As a Google-developed language, Dart benefits from strong support and continuous improvements from one of the world's leading technology companies. Many high-profile Google applications, such as the AdWords platform, are built using Dart.

Career Opportunities: As the adoption of Flutter grows, the demand for Dart developers is increasing. Learning Dart can open up new career opportunities in mobile and web development.

By learning Dart, you equip yourself with a powerful tool for building high-performance applications efficiently and effectively across multiple platforms. This guide will help you get started on your Dart journey, providing a solid foundation for further exploration and development.

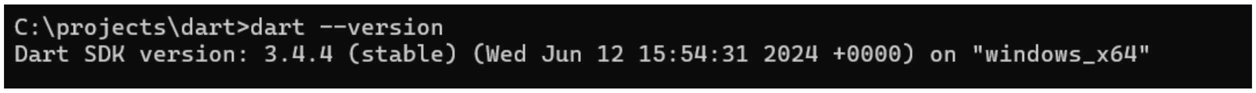
Chapter 1: Getting Started

Setting Up Your Environment

Installing Dart SDK

To start developing with Dart, you need to install the Dart SDK on your machine. Follow the steps for different operating systems at https://dart.dev/get-dart

Verify the installation by running `dart --version`.



```
C:\projects\dart>dart --version
Dart SDK version: 3.4.4 (stable) (Wed Jun 12 15:54:31 2024 +0000) on "windows_x64"
```

Setting up an IDE

Using an Integrated Development Environment (IDE) can greatly enhance your productivity. Visual Studio Code (VS Code) is a popular choice for Dart development.

Installing Visual Studio Code

Download VS Code from the official website and install it.

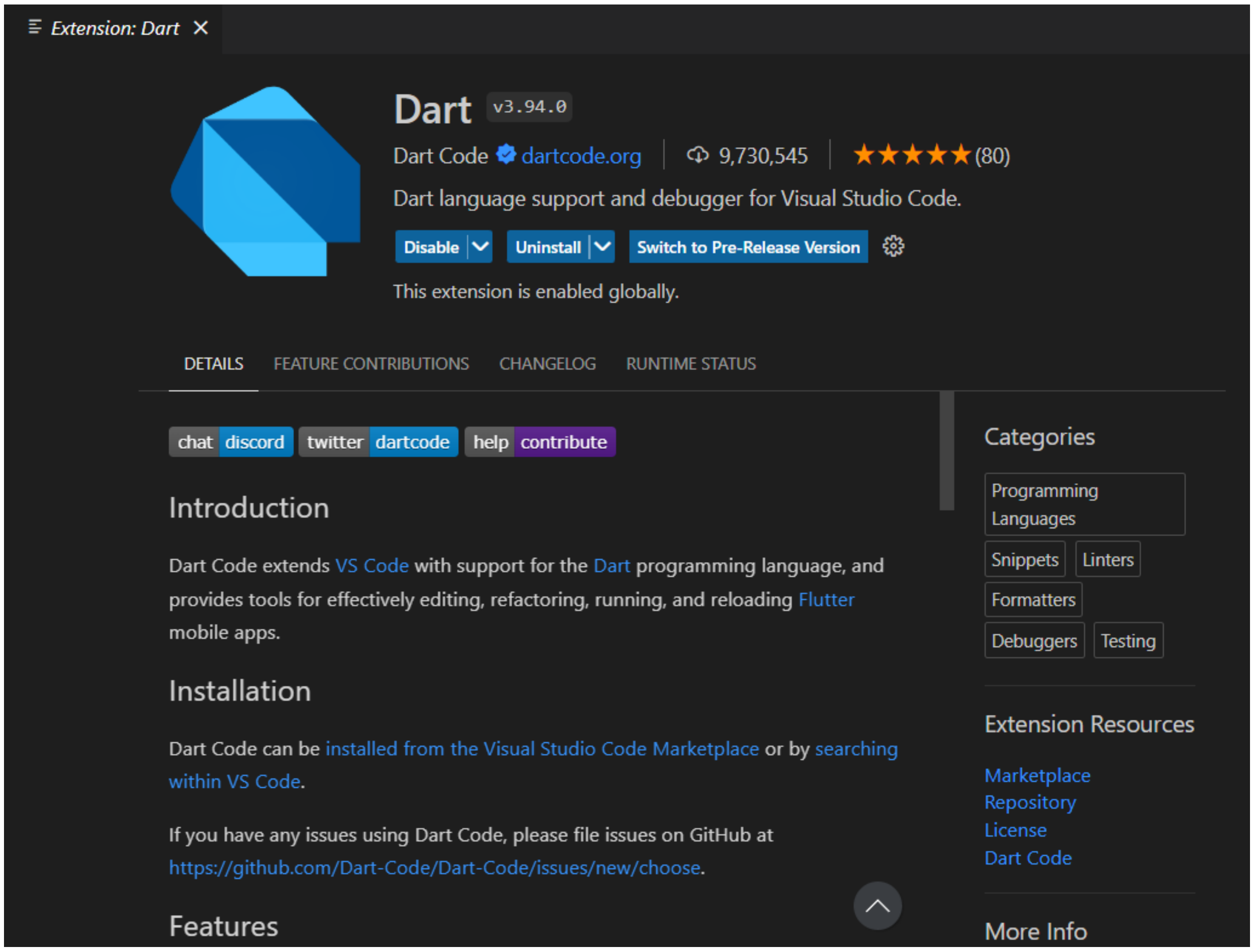
Setting up Dart in VS Code

Open VS Code.

Go to the Extensions view by clicking the square icon in the sidebar or pressing Ctrl+Shift+X.

Search for the "Dart" extension by Dart Code and install it.

After installation, restart VS Code.



Dart language support and debugger for Visual Studio Code.

This extension is enabled globally.

Introduction

Dart Code extends VS Code with support for the Dart programming language, and provides tools for effectively editing, refactoring, running, and reloading Flutter mobile apps.

Installation

Dart Code can be installed from the Visual Studio Code Marketplace or by searching within VS Code.

If you have any issues using Dart Code, please file issues on GitHub at https://github.com/Dart-Code/Dart-Code/issues/new/choose.

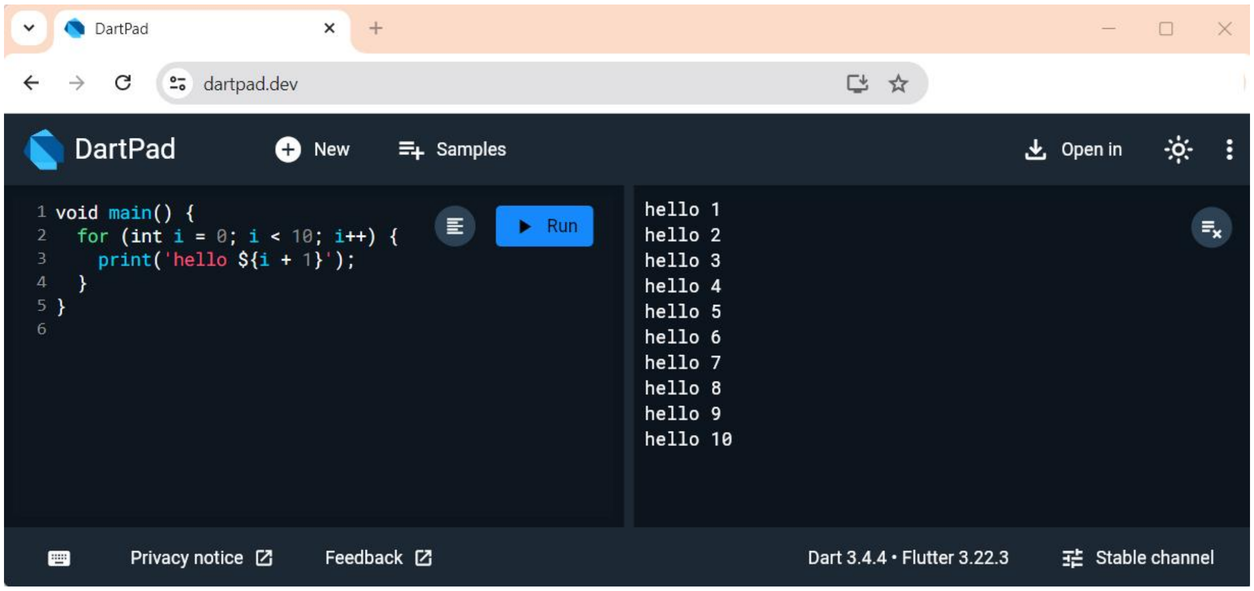
Features

Running App in DartPad

DartPad is an online editor that allows you to write, run, and share Dart code without any setup.

Open DartPad.

You will see a pre-written "Hello World" program. Click the "Run" button to execute the code.



```
void main() {
  for (int i = 0; i < 10; i++) {
    print('hello ${i + 1}');
  }
}
```

```
hello 1
hello 2
hello 3
hello 4
hello 5
hello 6
hello 7
hello 8
hello 9
hello 10
```

Hello World Program

Now that your environment is set up, let's write and run your first Dart program.

Writing and Running Your First Dart Program

Using Dart SDK:

Create a new file named `hello_world.dart`:

```
void main() {
  print('Hello, World!');
}
```

Open your terminal or command prompt and navigate to the directory where your file is saved.

Run the program using the Dart SDK: `dart hello_world.dart`

```
C:\projects\dart\learn_dart_in_40_minutes\chapter_01>dart hello_world.dart
Hello World!
```

You should see the output: `Hello, World!`

You can write and run the above source code using Visual Studio Code and/or DartPad.

By completing this chapter, you have successfully set up your Dart development environment and run your first Dart program. You are now ready to explore more features and capabilities of the Dart language.