presentations.AI

Introduction to Dart Programming

A Comprehensive Overview of Dart Programming Language and Its Applications

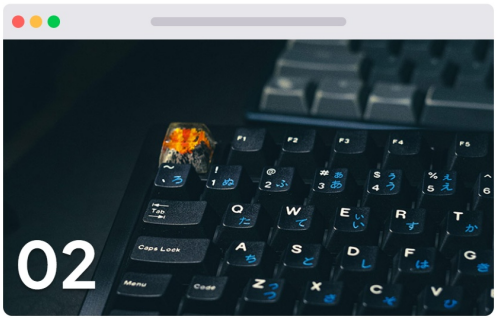
uwihanganye obed
Presenter

Understanding Dart: An Overview

Introduction to the key features of Dart programming language.

Platform Independence

Dart can run on various platforms including web, mobile, and server environments.



Strongly Typed

Dart is statically typed, which helps catch errors during development.

Asynchronous Programming

Dart supports asynchronous programming with features like Future and Stream, making it excellent for handling I/O operations in web applications.

Setting Up the Dart Environment

A step-by-step guide to preparing your Dart programming setup

Use DartPad

Write and execute Dart code directly in your browser at DartPad (https://dartpad.dartlang.org/).

Download Dart SDK

Get the Dart SDK from the official Dart site to start your development.

Install SDK

Follow the provided instructions specific to your operating system: Windows, macOS, or Linux.

Verify Installation

Run `dart --version` in your command line to confirm that Dart is correctly installed.

Use Recommended IDEs

Consider using IDEs such as IntelliJ IDEA, WebStorm, or Visual Studio Code for an enhanced Dart development experience.

Your First Dart Program

Understanding the Basics of Dart Programming

01 Main Function

This is the entry point of every Dart application, where execution begins.

03 Execution Methods

You can run this program using Terminal or an Integrated Development Environment (IDE).

02 Print Function

This built-in function outputs data to the console, crucial for displaying results.

04 Expected Output

The expected output will be 'Hello, World!', a standard for introductory programming.

Dart Syntax Essentials

Key rules for writing Dart code effectively

Single-line comments

Use `//` for single-line comments to annotate code.

01

Multi-line comments

Use `/* ... */` for multi-line comments to explain complex code.

02

Variable declaration

Declare variables using `var`, `int`, `double`, or `String` for different data types.

03

Control structures

Dart supports typical control structures like loops and conditionals for flow control.

04

Data Types and Variables in Dart

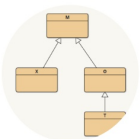
Understanding Dart's Fundamental Data Types for Effective Programming

int

Represents integers used for whole number calculations.

double

Used for floating-point numbers, suitable for decimal values.

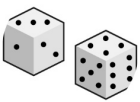


String

Handles textual data, useful for storing characters and sentences.

bool

Represents boolean values, either true or false, for logical operations.



Type Inference

Allows omission of type if the value is clear, simplifying code syntax.

Importance of Data Types

Understanding data types is crucial for effective Dart programming and variable declaration.

Control Flow in Dart

Understanding Conditional Statements, Switch Statements, and Loops

Conditional Statements

if and else statements allow decisions based on conditions.

01

Loops

Dart supports for, while, and do-while loops for repeated execution.

03

Switch Statements

Useful for executing one block of code among many options.

02

Dart Collections: Lists and Maps

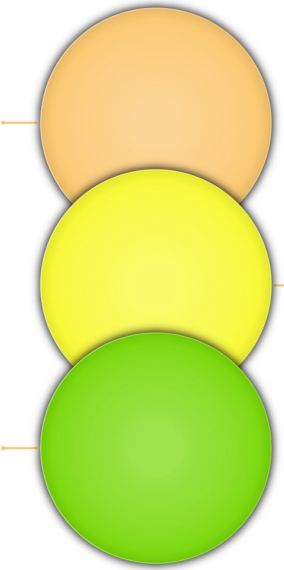
Understanding the Basics of Dart Collections

Lists

Ordered collections of items, allowing for indexed access and manipulation.

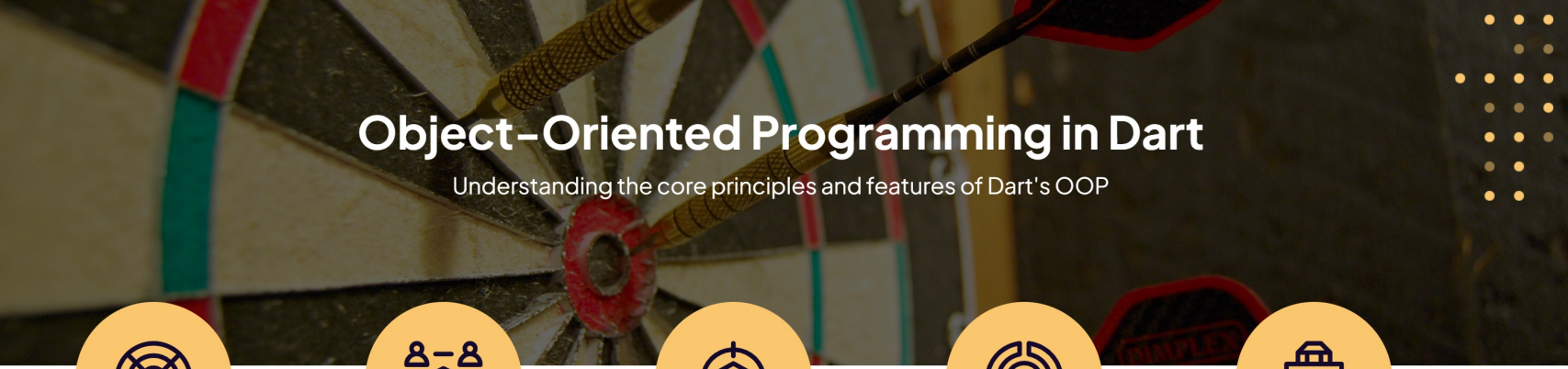
Common Operations

Adding items: Use `add()` for lists and `putIfAbsent()` for maps. Accessing items: Use index for lists and keys for maps.



Maps

Key-value pairs suitable for associating values with keys, enabling quick lookups.



Object-Oriented Programming in Dart

Understanding the core principles and features of Dart's OOP

01 Object-Oriented Language

Dart is an object-oriented language, supporting classes and objects.

02 Class Declaration

Define a class using the `class` keyword, establishing a blueprint for objects.

03 Creating Objects

Instantiate an object from a class to utilize its functionality in an application.

04 Inheritance

Dart supports class inheritance, allowing you to create subclasses that extend the functionality of existing classes.

05 Code Reuse

Utilizing OOP principles enables code reuse and a more organized structure, enhancing maintainability.

Error Handling in Dart

Understanding Error Management Techniques in Dart Programming

Robust Error Handling

Dart provides a robust way to handle errors using try-catch blocks.

Try-Catch Mechanism

Try-Catch: Catch exceptions and handle errors gracefully.

Finally Block

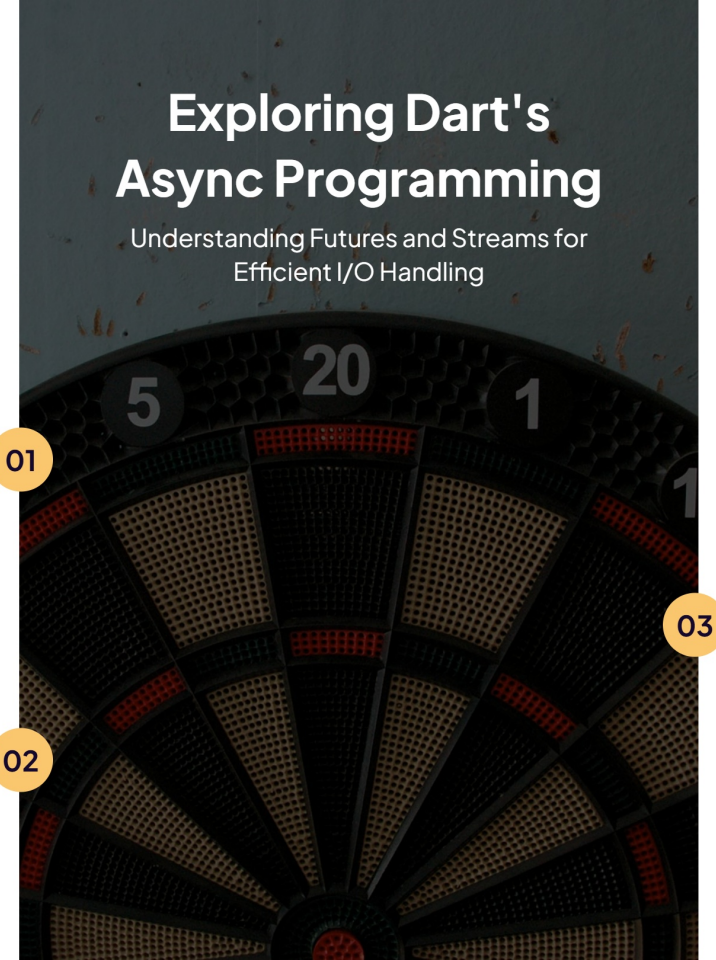
Finally Block: Code that always runs whether an exception occurs or not.

Custom Exceptions

Dart supports custom exceptions for better error management.

Exploring Dart's Async Programming

Understanding Futures and Streams for Efficient I/O Handling



01

Future

Represents a value that may not be available yet but will be at some point in the future.

02

Stream

A sequence of asynchronous events that can handle multiple values over time.

03

Mastering async programming

Allows for efficient handling of I/O operations and makes applications responsive.

Discover the Power of Dart Programming

Unlock your potential in application development with Dart.

Know More