MAKESKILLED

Innovations | Hackathons | Startups

Introduction to Dart: A Developer's Overview

Welcome to our presentation on Dart, Google's versatile language. It's built for modern app development. Dart excels across platforms. It powers the popular Flutter UI framework. Prepare to explore its core features and benefits.

by Makeskilled Team

Dart Basic Structure: Your First Steps

Entry Point

Every Dart program starts with **void main()**.

Statement Endings

Each statement must end with a semicolon.

Code Blocks

Curly braces define code blocks. This organises your logic.

Understanding these fundamentals is key. They form the backbone of all Dart applications. It's a clean, readable syntax.

MAKESKILLED
Innovations | Hackathons | Startups

Variables in Dart: Flexible Data Storage

Declaration

var uses type inference. Explicit types are also supported.

Immutability

final and **const** declare immutable variables. Use them for fixed values.

Dart's flexible variable declaration suits different needs. **camelCase** is the standard naming convention. This ensures code readability and consistency.

Choosing the right declaration improves performance. It also enhances maintainability. Always consider variable scope.

MAKESKILLED
Innovations | Hackathons | Startups

Datatypes in Dart: Ensuring Type Safety

Core Types

Includes **int**, **double**, **String**, **bool**, **List**, **Map**.

Type Safety

Dart provides static type checking. This prevents common errors.

Null Safety

Null safety is enabled by default. This avoids null reference issues.

Dart's robust type system ensures reliable code. Understanding each datatype is crucial. It supports strong, predictable programming.

These types are fundamental building blocks. They help you structure complex data. Always leverage them for clarity.

Operators in Dart: Logic and Computation

MAKESKILLED
Innovations | Hackathons | Startups

Arithmetic

Perform basic calculations: **+, -, *, /, ~/**.

Comparison

Evaluate relationships: **==, !=, >, <, >=, <=**.

Logical

Combine conditions: **&&, ||, !**.

Assignment

Assign values: **=, +=, -=**.

Null-Aware

Handle nulls safely: **??, ??=, ?.**.

Dart offers a comprehensive set of operators. They enable diverse programming tasks. Master them for efficient code.

Conditional Statements: Controlling Flow

MAKESKILLED
Innovations | Hackathons | Startups

If-Else Logic

Use **if**, **else if**, **else** for decision-making.

Switch-Case

Handle multiple conditions with **switch-case** statements.

Ternary Operator

Concise conditional assignments: **condition ? expr1 : expr2**.

Conditional statements are vital for dynamic applications. They dictate program behaviour. Choose the best structure for clarity.

These constructs ensure your app responds intelligently. They adapt to varying inputs. Practice using them effectively.

Loops in Dart: Efficient Iteration

Basic Loops

for, **while**, and **do-while** are standard.

Collection Iteration

for-in simplifies iterating over collections.

Loop Control

break exits loops; **continue** skips iterations.

Loops are fundamental for repetitive tasks. They streamline data processing. Select the appropriate loop type for efficiency.

MAKESKILLED
Innovations | Hackathons | Startups

Functions in Dart: Modular Code

Definition

Define with **returnType functionName()**.

Arrow Syntax

Concise functions use **() => expr**.

Parameters

Support optional and named parameters.

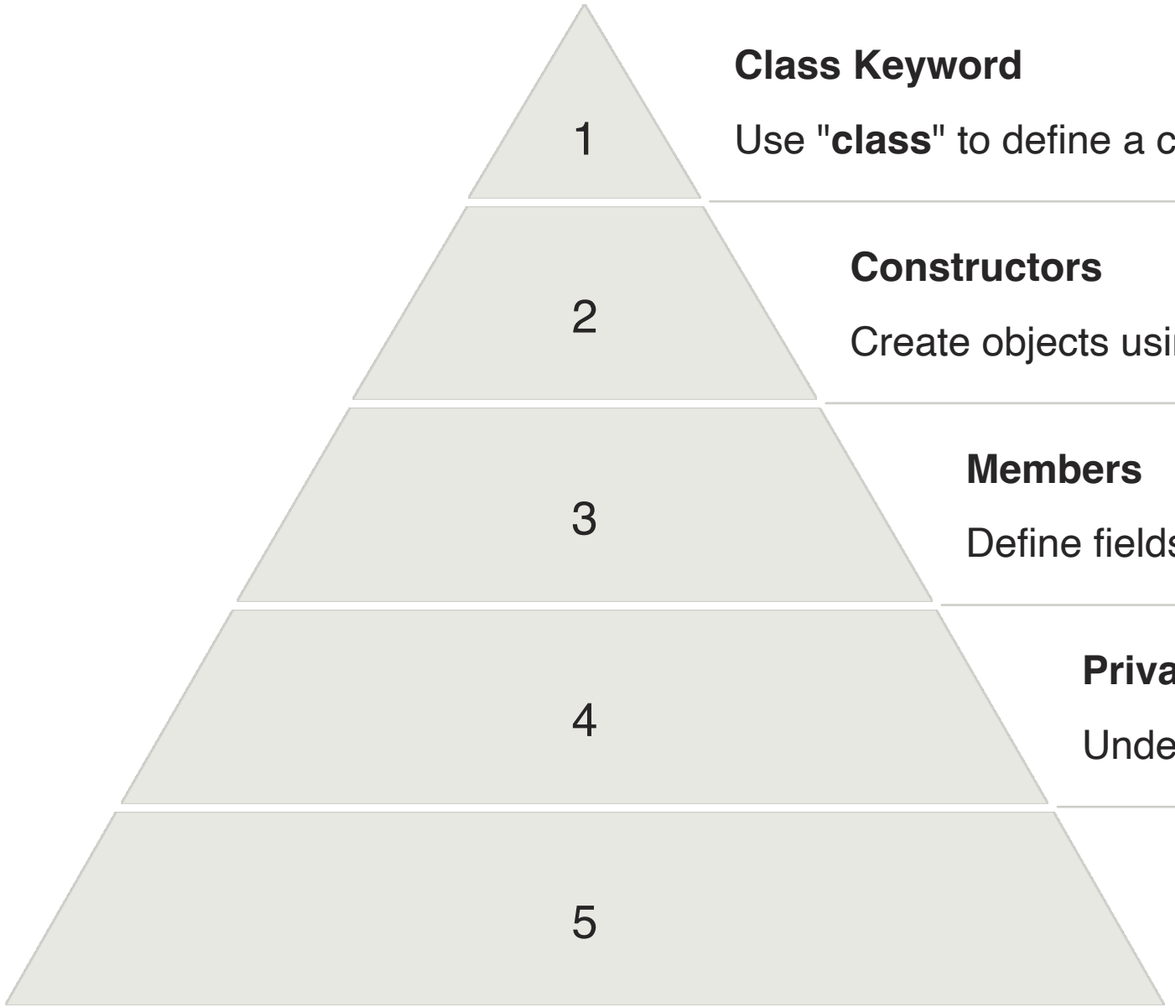
Anonymous Functions

Also known as lambdas, for inline use.

Functions promote code reusability. They enhance maintainability and readability. Master them for organised development.

Classes and Objects: Object-Oriented Dart

MAKESKILLED
Innovations | Hackathons | Startups



Class Keyword

Use "**class**" to define a class.

Constructors

Create objects using constructors.

Members

Define fields, methods, getters, setters.

Private

Underscore prefix makes members private.

Static

Static variables and methods supported.

Dart's object-oriented features enable powerful designs. They structure complex applications. Embrace them for scalability.

Additional Dart Syntax Essentials

MAKESKILLED
Innovations | Hackathons | Startups

Comments

// for single-line, **/* */** for multi-line.

String Interpolation

Embed variables: **"Hello, $name"**.

Collection Literals

Easy lists **[]** and maps **{}**.

Enums

Define fixed sets of values.

Imports

Use **import** for modular code.

These elements enhance Dart's expressiveness. They make code cleaner and more efficient. Incorporate them into your practice.