**Dart Programming Language: A Beginner’s Tutorial**

Dart is a modern, object-oriented programming language developed by Google, mainly used for building mobile, web, and server applications. It is the primary language for **Flutter**, Google's UI toolkit. This tutorial will introduce you to the basics of Dart, including syntax, data types, control flow, functions, and object-oriented programming (OOP).

**1. Setting Up Dart**

Before you start coding, you need to install Dart. You can:

* Download and install Dart from dart.dev.
* Use DartPad ([dartpad.dev](https://dartpad.dev/" \t "_new)) for online practice.

**2. Basic Syntax**

**Hello World**

A simple Dart program:

dart

CopyEdit

void main() {

print('Hello, Dart!');

}

* void main() is the entry point of the program.
* print() is used to display output.

**3. Variables and Data Types**

Dart supports **strong typing** but also allows type inference using var.

**Declaring Variables**

dart

CopyEdit

void main() {

int age = 25; // Integer

double height = 5.9; // Floating point number

String name = "Dart"; // String

bool isCool = true; // Boolean

var city = "London"; // Dart infers it as String

print("Name: $name, Age: $age, City: $city");

}

**Nullable and Non-Nullable Types (Dart is null-safe)**

By default, variables cannot be null, but you can allow null values using ?.

dart

CopyEdit

String? nullableName; // Can be null

nullableName = "John";

print(nullableName);

**4. Operators**

Dart supports arithmetic, relational, logical, and assignment operators.

dart

CopyEdit

void main() {

int a = 10, b = 5;

print(a + b); // Addition

print(a - b); // Subtraction

print(a \* b); // Multiplication

print(a / b); // Division (returns double)

print(a ~/ b); // Integer division

print(a % b); // Modulus

print(a > b); // Relational operator

}

**5. Control Flow (if-else, loops, switch)**

**if-else Statement**

dart

CopyEdit

void main() {

int num = 10;

if (num > 0) {

print("Positive number");

} else {

print("Negative number");

}

}

**for Loop**

dart

CopyEdit

void main() {

for (int i = 1; i <= 5; i++) {

print("Number: $i");

}

}

**while Loop**

dart

CopyEdit

void main() {

int i = 1;

while (i <= 5) {

print("Value: $i");

i++;

}

}

**switch-case**

dart

CopyEdit

void main() {

int choice = 2;

switch (choice) {

case 1:

print("One");

break;

case 2:

print("Two");

break;

default:

print("Invalid choice");

}

}

**6. Functions**

Functions help in code reuse and modularity.

**Defining Functions**

dart

CopyEdit

void sayHello() {

print("Hello, Dart!");

}

void main() {

sayHello();

}

**Function with Parameters**

dart

CopyEdit

void greet(String name) {

print("Hello, $name!");

}

void main() {

greet("Alice");

}

**Returning Values**

dart

CopyEdit

int add(int a, int b) {

return a + b;

}

void main() {

print(add(10, 5)); // Output: 15

}

**Arrow Functions (Short Syntax)**

dart

CopyEdit

int multiply(int a, int b) => a \* b;

void main() {

print(multiply(4, 3)); // Output: 12

}

**7. Object-Oriented Programming (OOP) in Dart**

Dart is an object-oriented language, meaning it supports **classes, objects, inheritance, polymorphism, and encapsulation**.

**Defining a Class and Creating Objects**

dart

CopyEdit

class Person {

String name;

int age;

// Constructor

Person(this.name, this.age);

// Method

void introduce() {

print("Hi, I am $name and I am $age years old.");

}

}

void main() {

Person p1 = Person("Alice", 30);

p1.introduce();

}

**Encapsulation (Private Variables)**

In Dart, \_ (underscore) makes a variable private.

dart

CopyEdit

class BankAccount {

double \_balance = 1000; // Private variable

void deposit(double amount) {

\_balance += amount;

}

void showBalance() {

print("Balance: $\_balance");

}

}

void main() {

var account = BankAccount();

account.deposit(500);

account.showBalance(); // Balance: 1500

}

**Inheritance**

dart

CopyEdit

class Animal {

void makeSound() {

print("Animal makes a sound");

}

}

class Dog extends Animal {

void bark() {

print("Dog barks");

}

}

void main() {

Dog myDog = Dog();

myDog.makeSound(); // Inherited method

myDog.bark();

}

**Polymorphism (Method Overriding)**

dart

CopyEdit

class Vehicle {

void start() {

print("Vehicle is starting...");

}

}

class Car extends Vehicle {

@override

void start() {

print("Car is starting...");

}

}

void main() {

Vehicle myCar = Car();

myCar.start(); // Output: Car is starting...

}

**Abstract Classes**

Abstract classes cannot be instantiated but can be inherited.

dart

CopyEdit

abstract class Shape {

void draw(); // Abstract method

}

class Circle extends Shape {

@override

void draw() {

print("Drawing a circle");

}

}

void main() {

Circle c = Circle();

c.draw();

}

**8. Collections (Lists, Sets, Maps)**

**Lists (Arrays)**

void main() {

List<int> numbers = [1, 2, 3, 4, 5];

print(numbers[0]); // Accessing elements

numbers.add(6); // Adding element

print(numbers);

}

**Sets (Unique Elements)**

void main() {

Set<int> uniqueNumbers = {1, 2, 3, 3, 4};

print(uniqueNumbers); // Output: {1, 2, 3, 4}

}

**Maps (Key-Value Pairs)**

void main() {

Map<String, int> ages = {'Alice': 30, 'Bob': 25};

print(ages['Alice']); // Output: 30

}

**9. Asynchronous Programming (Future & Async/Await)**

Dart uses Future for asynchronous operations like API calls or file reading.

Future<String> fetchData() async {

await Future.delayed(Duration(seconds: 2)); // Simulating delay

return "Data loaded";

}

void main() async {

print("Fetching data...");

String data = await fetchData();

print(data); // Output after 2 sec: Data loaded

}