Learn DART for Flutter | What is Dart Programming? | Dart Programming Language

Introduction

Dart is a powerful programming language developed by Google, primarily used for building mobile, desktop, and web applications. Before diving into Flutter development, it's crucial to understand Dart because Flutter relies entirely on Dart for its development. This article will introduce you to the basics of Dart, its importance in Flutter, and some essential features that make it a suitable language for modern app development.

1. What is Dart?

- Dart is an open-source, object-oriented, and strongly-typed language developed by Google in 2011.
- It's designed to be a client-optimized language for fast apps on any platform.

- Dart can be compiled to native code or JavaScript, making it versatile for web and mobile applications.

2. Why Learn Dart Before Flutter?

- **Core Language for Flutter:** Flutter uses Dart as its programming language. Understanding Dart is essential for Flutter development.
- **Unified Language:** Unlike other frameworks that require multiple languages for front-end and back-end development, Flutter and Dart offer a single language for both UI and logic.
- **Simplifies Learning Curve:** By mastering Dart, you can focus solely on Flutter's widgets and tools without worrying about integrating multiple languages.

3. Key Features of Dart

- **Object-Oriented:** Dart supports all Object-Oriented Programming (OOP) concepts, including classes, objects, inheritance, and polymorphism.
- **Strongly-Typed:** Variables in Dart are strongly typed, meaning you need to declare the data type of variables during their declaration.
- **Asynchronous Programming with `async` and `await`:** Dart provides built-in support for asynchronous programming, allowing you to write non-blocking code that is easy to read and maintain.
- **Ahead-of-Time (AOT) and Just-in-Time (JIT) Compilation:** Dart supports both AOT and JIT compilation, providing fast execution and development cycles.

4. Understanding Dart's Object-Oriented Nature

Dart is an object-oriented language, meaning everything in Dart is an object, even functions. This allows you to create complex data structures and classes to manage the logic of your applications.

- **Classes and Objects:** Classes are blueprints for creating objects. An object is an instance of a class.

```dart
// Example of a simple Dart class
class Animal {
  String name;
  int age;

  // Constructor
  Animal(this.name, this.age);

  // Method
  void displayInfo() {
    print("Name: $name, Age: $age");
  }
}

void main() {
  Animal dog = Animal("Buddy", 5);
  dog.displayInfo(); // Output: Name: Buddy, Age: 5
}
```

5. Strong Typing in Dart

Dart is a strongly typed language, which means you must specify the type of a variable during its declaration. This helps catch errors early in the development process.

```dart
// Example of strongly typed variables
int age = 25;
String name = "Alice";

// This will throw an error because Dart is strongly typed
// age = "Twenty-five";
```

6. Asynchronous Programming with `async` and `await`

Dart provides powerful support for asynchronous programming, which is essential for tasks like fetching data from the internet, reading files, or performing time-consuming operations without freezing the user interface.

- **Async and Await:**
  - `async` is used to declare a function that will return a `Future`.
  - `await` is used to wait for a `Future` to complete and return the result.

```dart
// Example of async and await
Future<String> fetchUserOrder() async {
  // Imagine that this function fetches user orders from an API
  return Future.delayed(Duration(seconds: 3), () => "User Order: Latte");
}

void main() async {
  print("Fetching user order...");
  String order = await fetchUserOrder();
  print(order); // Output after 3 seconds: User Order: Latte
}
```

7. Compilation Techniques: AOT and JIT

Dart supports both Ahead-of-Time (AOT) and Just-in-Time (JIT) compilation:

- **AOT Compilation:** Used for production builds. It compiles the code before execution, which improves startup times and performance.
- **JIT Compilation:** Used during development. It compiles the code at runtime, allowing features like hot reload in Flutter, where changes in code are reflected instantly without restarting the application.
- **Example:**
  - AOT helps in deploying fully compiled applications with optimized performance.
  - JIT allows developers to see changes immediately during development, improving productivity.

8. Dart and JavaScript Interoperability

Dart can be compiled into JavaScript, enabling it to run in any web browser. This makes it a powerful tool for web development, alongside Flutter's capability to build web applications.

```dart
// Simple Dart code that can be compiled to JavaScript
void main() {
  print("Hello, Dart to JavaScript!");
}
```

9. Unified Development with Dart in Flutter

In Flutter, Dart is used for everything—UI layout, business logic, state management, and more. This unification simplifies the development process, as you don’t need to learn multiple languages or frameworks to build a complete application.

10. Benefits of Using Dart in Flutter

- **Hot Reload:** Dart's JIT compilation enables Flutter’s hot reload feature, allowing you to see changes instantly without restarting the app.
- **Cross-Platform:** Dart allows you to write code once and run it on multiple platforms—iOS, Android, web, and desktop.
- **Optimized for UI Development:** Dart’s syntax and features are optimized for UI development, making it easier to build responsive and dynamic user interfaces.

Conclusion

Dart is a versatile, powerful language that is essential for Flutter development. Whether you are a beginner or an experienced developer, learning Dart will not only streamline your Flutter development process but also enhance your overall programming skills. Start with Dart, and you'll find the transition to Flutter development smooth and intuitive.

In the upcoming sessions, we will dive deeper into Dart programming, covering essential topics like variables, functions, classes, asynchronous programming, and more, with practical examples and exercises. This foundational knowledge will set you on the path to mastering Flutter development.

This article should provide a structured overview of Dart and its significance in Flutter development, with clear points, code examples, and explanations to help beginners get started.