1. Fundamental Data Types in Dart

Dart is a strongly typed language, but it offers a dynamic typing system for flexibility. Here are some of its core data types:

- **int**: Represents integer values, which are whole numbers without a decimal point.
    - **Uses:** Counting, indexing, and any mathematical operation involving whole numbers.
    - **Example:** int age = 30;
- **double**: Represents floating-point numbers, which have a decimal point.
    - **Uses:** Calculations involving fractions, measurements, or currency.
    - **Example:** double price = 19.99;
- **String**: Represents a sequence of characters. Strings can be created using single (') or double (") quotes.
    - **Uses:** Storing text, names, messages, or any textual data.
    - **Example:** String name = 'Alice';
- **bool**: Represents a boolean value, which can only be either true or false.
    - **Uses:** Conditional logic and control flow.
    - **Example:** bool isLoggedIn = true;
- **List**: Represents an ordered collection of values, similar to an array in other languages.
    - **Uses:** Storing a collection of items in a specific order.
    - **Example:** List<String> fruits = ['apple', 'banana', 'orange'];
- **Map**: Represents a collection of key-value pairs, where each key is unique.
    - **Uses:** Storing data in a structured way, like a dictionary or a JSON object.
    - **Example:** Map<String, int> scores = {'Alice': 95, 'Bob': 88};

---

2. Control Structures in Dart

Control structures are used to control the flow of a program's execution.

- **if and else**: These are used for conditional execution. The code block inside if runs if the condition is true, otherwise the else block runs.
- **for loop**: Used to iterate a specific number of times.
- **while loop**: Repeats a block of code as long as a condition is true.
- **switch**: Used to execute different code blocks based on a single variable's value.

---

3. Object-Oriented Programming (OOP) in Dart

Dart is an object-oriented language, and it supports key OOP concepts.

- **Classes**: A class is a blueprint for creating objects. It defines properties (data) and methods (functions) that an object can have.
- **Inheritance**: Allows a new class (subclass) to inherit the properties and methods of an existing class (superclass). This promotes code reusability. In Dart, a subclass uses the extends keyword to inherit from a superclass.
- **Polymorphism**: The ability of an object to take on many forms. A subclass object can be treated as an object of its superclass. Methods can be overridden in subclasses to provide specific implementations.
- **Interfaces**: In Dart, every class implicitly defines an interface. You can implement a class's interface using the implements keyword. This ensures a class has all the methods and properties of the interface.

---

4. Asynchronous Programming in Dart

Asynchronous programming allows your code to perform time-consuming tasks without blocking the main thread, keeping the UI responsive. Dart handles this with Future, async, and await.

- **Future**: Represents a potential value or error that will be available at some point in the future. It's an object that is returned immediately when an asynchronous function is called.
    - **Example:** A function that fetches data from a server returns a Future<String>.

- **async and await**: These keywords make asynchronous code look and behave like synchronous code.
    - async is used to mark a function as asynchronous. This allows it to use the await keyword.
    - await is used to pause the execution of an async function until a Future has completed.
- **Stream**: Represents a sequence of asynchronous events. A Stream can emit multiple values over time, unlike a Future which emits a single value.
    - **Uses:** Handling data from a continuous source, like user input, database queries, or a network connection that continuously sends data.
    - **Example:** A Stream could be used to listen for button clicks or to receive real-time data updates.