# 📘 Flutter Theory Assignments

## 1. Benefits of Using Flutter Over Other Cross-Platform Frameworks

* **Single codebase**: Write once, run on iOS, Android, web, and desktop.
* **Hot reload**: Allows real-time UI and logic updates during development.
* **Custom rendering engine**: Ensures consistent design across platforms.
* **Performance**: Near-native performance thanks to Dart's compiled code.

## 2. Role of Dart in Flutter and Its Advantages

* Dart is Flutter’s official programming language.
* It compiles to **native ARM/x86 code** for high-performance mobile apps.
* Supports both **JIT (Just-in-Time)** for development and **AOT (Ahead-of-Time)** for release.
* Dart has simple and familiar syntax for developers from Java, JavaScript, or C# backgrounds.

## 3. Steps to Set Up a Flutter Development Environment

1. Download and install the **Flutter SDK** from [flutter.dev](https://flutter.dev).
2. Install **Flutter & Dart plugins** in an IDE (e.g., Android Studio or VS Code).
3. Run flutter doctor in the terminal to check for missing dependencies.
4. Create and run your first project using flutter create.

## 4. Basic Flutter App Structure

* **main.dart** is the entry point of a Flutter app.
* The main() function calls runApp() to load the root widget.
* UI is built using a **widget tree**, combining reusable components to form complex interfaces.

## 5. Fundamental Data Types in Dart

* int: Whole numbers (e.g., 5)
* double: Decimal numbers (e.g., 3.14)
* String: Text (e.g., "Hello")
* List: Ordered collections (e.g., [1, 2, 3])
* Map: Key-value pairs (e.g., {'name': 'John'})

## 6. Control Structures in Dart

* **if / else**: Conditional branching
* **for / while**: Looping
* **switch / case**: Multi-branch condition handling

**Example:**

dart

CopyEdit

if (x > 0) print("Positive");

for (var i = 0; i < 5; i++) print(i);

## 7. Object-Oriented Programming in Dart

* **Class**: Blueprint for creating objects.
* **Inheritance**: Allows a class to inherit from another.
* **Polymorphism**: Enables the use of different object types via a common interface.
* Dart lacks formal interfaces but uses **abstract classes** to define them.

## 8. Asynchronous Programming in Dart

* **Future**: Represents a value that will be available later.
* **async/await**: Used to write asynchronous code that looks like synchronous code.
* **Stream**: Handles multiple asynchronous events (like user input or sensor data).

## 9. Difference Between Stateless and Stateful Widgets

| **StatelessWidget** | **StatefulWidget** |
| --- | --- |
| Immutable | Mutable |
| UI doesn’t change | UI can change with state |
| Example: Text("Hello") | Example: Counter app with increment |

## 10. Widget Lifecycle and State Management

* **initState()**: Called when the widget is inserted into the widget tree.
* **build()**: Rebuilds the UI when setState is called.
* **dispose()**: Called when the widget is removed from the tree.

State is managed in the State<T> class and updated using setState().

## 11. Five Common Flutter Layout Widgets

1. **Container** – Box-like widget for styling, margin, padding, and color.
2. **Column** – Vertically arranges widgets.
3. **Row** – Horizontally arranges widgets.
4. **Center** – Centers its child within itself.
5. **Expanded** – Fills remaining space within Column/Row.

## 12. How the Navigator Widget Works in Flutter

* Manages a **stack** of routes (screens).
* Use Navigator.push() to add a new screen to the stack.
* Use Navigator.pop() to go back.
* Mimics browser-like back/forward navigation.

## 13. Concept of Named Routes and Their Advantages

* Define routes with string names like /home, /details.
* Navigate using Navigator.pushNamed(context, '/home').
* Easier to manage, maintain, and reuse across the app.
* Promotes **clean code** and centralized route handling.

## 14. Passing Data Between Screens Using Route Arguments

* Pass arguments using:

dart

CopyEdit

Navigator.pushNamed(context, '/details', arguments: 'data');

* Receive arguments using:

dart

CopyEdit

final data = ModalRoute.of(context)?.settings.arguments;

* Useful for passing values like IDs, strings, or entire objects between screens.