Mengapa sangat penting untuk memahami bahasa pemrograman Dart sebelum kita menggunakan framework Flutter ? Jelaskan!

**1. Dasar dari Flutter adalah Dart**

Flutter tidak bisa dipisahkan dari Dart. Semua komponen, mulai dari **widget, state management, routing, hingga async programming**, ditulis dengan Dart. Jika tidak memahami Dart, maka akan kesulitan memahami alur kerja Flutter.

**2. Pemahaman Logika & Sintaks**

Dart menyediakan konsep dasar pemrograman seperti **variabel, tipe data, control flow (if, for, while), functions, OOP (class, inheritance, polymorphism)**.  
Tanpa menguasai ini, kita hanya akan menghafal kode Flutter tanpa benar-benar mengerti fungsinya.

**3. Object-Oriented Programming (OOP)**

Flutter menggunakan pendekatan **OOP** (semua widget adalah class). Dart punya fitur OOP modern seperti:

* **Class & Object**
* **Constructor**
* **Inheritance**
* **Mixins**
* **Generics**  
  Kalau OOP Dart tidak dipahami, sulit sekali menguasai konsep widget tree dan stateful/stateless widget di Flutter.

**4. Asynchronous Programming**

Dalam Flutter banyak proses yang berjalan secara **asynchronous** (misalnya ambil data API, akses database lokal, baca file). Dart menggunakan **async, await, dan Future**. Tanpa paham ini, kita tidak bisa mengelola data dengan benar di Flutter.

**5. Debugging & Problem Solving**

Kalau hanya tahu Flutter tapi tidak paham Dart, ketika error muncul akan sangat sulit mencari solusi. Dengan paham Dart, kita bisa **menelusuri error, membaca stack trace, dan memperbaiki bug** lebih cepat.

**6. Efisiensi Belajar Flutter**

Jika sudah paham Dart, belajar Flutter akan lebih cepat karena kita hanya fokus memahami **framework**-nya, bukan bingung dengan sintaks dasar bahasa.

Rangkumlah materi dari praktikum ini menjadi poin-poin penting yang dapat Anda gunakan untuk membantu proses pengembangan aplikasi mobile menggunakan framework Flutter.

**Poin-Poin Penting Praktikum Dart untuk Flutter**

1. **Functions dalam Dart**
   * Fungsi digunakan untuk memecah program menjadi bagian kecil yang dapat digunakan kembali.
   * Jenis parameter:
     + **Positional Parameter** (parameter biasa, sesuai urutan).
     + **Optional Parameter** (bisa disertakan atau tidak, dengan tanda [] atau {}).
     + **Named Parameter** (lebih jelas dengan nama, biasanya pakai {}).
     + **Default Parameter** (memiliki nilai default).
2. **Functions sebagai First-Class Objects**
   * Fungsi di Dart bisa diperlakukan seperti variabel: disimpan dalam variabel, dikirim sebagai parameter, atau dikembalikan dari fungsi lain.
3. **Anonymous Functions (Lambda/Closure)**
   * Fungsi tanpa nama yang biasanya digunakan untuk callback atau operasi singkat.
   * Contoh: (x, y) => x + y;
4. **Lexical Scope & Closures**
   * **Lexical Scope**: variabel hanya bisa diakses dalam lingkup tempat ia didefinisikan.
   * **Closure**: fungsi bisa “mengingat” variabel di scope luar walaupun scope itu sudah selesai dieksekusi.
5. **Return Multiple Values**
   * Dart tidak mendukung return banyak nilai secara langsung, tapi bisa menggunakan:
     + **List**
     + **Map**
     + **Class/Record (Dart 3)** untuk mengembalikan beberapa nilai sekaligus.

**Manfaat untuk Flutter**

* **Functions & Parameter** → membantu dalam membuat widget yang reusable dan konfigurasi widget lebih fleksibel.
* **First-Class Functions & Anonymous Functions** → dipakai pada event handler seperti onPressed, onChanged.
* **Closures** → penting untuk state management dan callback.
* **Async/Await (Future & Stream)** → sangat krusial saat mengambil data API, membaca database lokal, atau operasi I/O di Flutter.
* **Return Multiple Values** → memudahkan passing data antar widget atau pengelolaan data kompleks.