LEMBAR TUGAS

Pemrograman Mobile

P-2

Yonatan Efrassetyo (27) - 3C

Nama : Yonatan Efrassetyo

NIM : 2241720063

Kelas : 3C

Tugas

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

1. Pengantar Bahasa Pemrograman Dart Bagian 1

Bahasa Dart adalah inti dari framework Flutter. Kerangka kerja modern seperti Flutter membutuhkan bahasa modern tingkat tinggi agar bisa memberikan pengalaman terbaik kepada pengembang, serta memungkinkan untuk membuat aplikasi seluler yang luar biasa. Memahami Dart adalah dasar untuk bekerja dengan Flutter; pengembang perlu mengetahui asal-usul bahasa Dart, bagaimana komunitas mengerjakannya, kelebihannya, dan mengapa itu adalah bahasa pemrograman yang dipilih untuk Flutter.

Intinya mempelajari suatu dasar adalah hal penting karena akan memudahkan kita kedepannya dan tidak akan membuat kita bingung.

2. Getting started with Dart

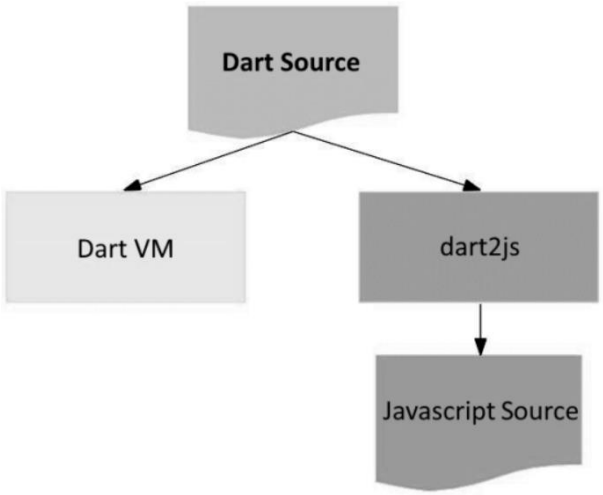
Dart bertujuan untuk menggabungkan kelebihan-kelebihan dari sebagian besar bahasa tingkat tinggi dengan fitur-fitur bahasa pemrograman terkini, antara lain sebagai berikut:

- **Productive tooling**: merupakan fitur kakas (*tool*) untuk menganalisis kode, plugin IDE, dan ekosistem paket yang besar.
- **Garbage collection**: untuk mengelola atau menangani dealokasi memori (terutama memori yang ditempati oleh objek yang tidak lagi digunakan).
- **Type annotations (opsional)**: untuk keamanan dan konsistensi dalam mengontrol semua data dalam aplikasi.
- **Statically typed**: Meskipun *type annotations* bersifat opsional, Dart tetap aman karena menggunakan fitur *type-safe* dan *type inference* untuk menganalisis *types* saat *runtime*. Fitur ini penting untuk menemukan *bug* selama kompilasi kode.
- **Portability**: bahasa Dart tidak hanya untuk web (yang dapat diterjemahkan ke JavaScript) tetapi juga dapat dikompilasi secara *native* ke kode **Advanced RISC Machines (ARM)** dan x86.

3. The evolution of Dart

Diluncurkan pada tahun 2011, Dart telah berkembang sejak saat itu. Dart merilis versi stabilnya pada tahun 2013, dengan perubahan besar termasuk dalam rilis Dart 2.0 menjelang akhir 2018

4. Introducing the structure of the Dart language



Kode Dart dapat dieksekusi pada lingkungan yang mendukung bahasa Dart. Lingkungan yang mendukung bahasa Dart perlu memperhatikan fitur-fitur penting seperti berikut:

- *Runtime systems*
- *Dart core libraries*
- *Garbage collectors*

5. Hands-on with Dart

- Object orientation

  Seperti kebanyakan bahasa modern, Dart dirancang untuk **object-oriented (OO)**. Secara singkat, Bahasa OOP didasarkan pada konsep **objek** yang menyimpan kedua data (disebut **fields**) dan kode (disebut **methods**). Objek-objek ini dibuat dari cetak biru yang disebut **class** yang mendefinisikan *field* dan *method* yang akan dimiliki oleh sebuah objek
- Dart Operators

  Di Dart, operator tidak lebih dari *method* yang didefinisikan dalam *class* dengan sintaks khusus. Jadi, ketika Anda menggunakan operator seperti `x == y`, seolah-olah Anda sedang memanggil `x.==(y)` metode untuk melakukan perbandingan kesetaraan.

- Arithmetic Operators

  `+` untuk tambahan.

  `-` untuk pengurangan.

  `*` untuk perkalian.

  `/` untuk pembagian.

  `~/` untuk pembagian bilangan bulat. Di Dart, setiap pembagian sederhana dengan `/` menghasilkan nilai *double*. Untuk mendapatkan nilai bilangan bulat, Anda perlu membuat semacam transformasi (yaitu, *typecast*) dalam bahasa pemrograman lain; namun Dart sudah mendukung untuk operasi ini.

  `%` untuk operasi modulus (sisa bagi dari bilangan bulat).

  `-expression` untuk negasi (yang membalikkan suatu nilai).

- Increment and Decrement Operators

  `++var` atau `var++` untuk menambah nilai variabel `var` sebesar 1

  `--var` atau `var--` untuk mengurangi nilai variabel `var` sebesar 1

- Equality and relational operators

  `==` untuk memeriksa apakah operan sama

  `!=` untuk memeriksa apakah operan berbeda

  `>` memeriksa apakah operan kiri lebih besar dari operan kanan

  `<` memeriksa apakah operan kiri lebih kecil dari operan kanan

  `>=` memeriksa apakah operan kiri lebih besar dari atau sama dengan operan kanan

  `<=` memeriksa apakah operan kiri kurang dari atau sama dengan operan kanan

- Logical Operators

  `!expression` negasi atau kebalikan hasil ekspresi—yaitu, `true` menjadi `false` dan `false` menjadi `true`.

  `||` menerapkan operasi logika `OR` antara dua ekspresi.

  `&&` menerapkan operasi logika `AND` antara dua ekspresi.