Bagian II: Dasar-Dasar Dart

Pada bagian ini, kita akan membahas dasar-dasar bahasa pemrograman Dart, yang merupakan bahasa utama yang digunakan dalam pengembangan aplikasi Flutter. Dart dirancang untuk menjadi bahasa yang mudah dipelajari dan efisien dalam membuat aplikasi modern. Dengan Dart, Anda akan dapat mengembangkan aplikasi yang berjalan cepat dan dapat digunakan di berbagai platform. Memahami dasar-dasar Dart sangat penting, karena Flutter bergantung sepenuhnya pada Dart untuk pengembangan aplikasi.

Dart memiliki sintaksis yang mirip dengan bahasa pemrograman lain seperti Java dan JavaScript, sehingga memudahkan pemrogram yang sudah berpengalaman dengan bahasa lain untuk beradaptasi dengan cepat. Di bagian ini, Anda akan mempelajari tentang tipe data, variabel, struktur kontrol, koleksi, serta cara menangani error dengan Dart. Pengetahuan ini akan memberi dasar yang kuat untuk membangun aplikasi yang lebih kompleks dengan Flutter dan memastikan kode yang Anda tulis dapat berjalan secara efisien dan tanpa masalah.

2.1 Pengantar Bahasa Pemrograman Dart

Dart adalah bahasa pemrograman yang dikembangkan oleh Google, dirancang khusus untuk membangun aplikasi modern yang berjalan cepat dan efisien di berbagai platform. Bahasa ini digunakan secara eksklusif dalam pengembangan aplikasi Flutter, memberikan para pengembang kontrol penuh terhadap aplikasi yang mereka bangun. Dart dipilih karena kemudahan penggunaannya, kecepatan eksekusi, dan

kemampuannya untuk mendukung aplikasi dengan antarmuka yang kaya dan performa tinggi.

Salah satu fitur utama Dart adalah kemampuannya untuk menjalankan kode secara asinkron, yang sangat penting dalam pengembangan aplikasi mobile untuk memastikan responsifitas aplikasi. Dart juga mendukung paradigma pemrograman berorientasi objek (OOP), yang memungkinkan pengembang untuk menulis kode yang terstruktur dan mudah dikelola. Dengan Dart, pengembang dapat membuat aplikasi yang mudah dipelihara, dengan sintaksis yang jelas dan kemampuan untuk menangani berbagai kebutuhan aplikasi dengan efisien.

2.2 Sejarah dan Fitur Dart

Dart pertama kali diperkenalkan oleh Google pada tahun 2011 dengan tujuan untuk menggantikan JavaScript dalam pengembangan aplikasi web. Namun, seiring berkembangnya teknologi, Dart juga mulai dipergunakan untuk pengembangan aplikasi mobile melalui Flutter. Pada awalnya, Dart dirancang untuk menjadi lebih cepat dan lebih efisien daripada JavaScript, serta untuk mempermudah pengembangan aplikasi web dan mobile.
Seiring dengan perkembangan Flutter, Dart menjadi semakin populer karena kemampuannya untuk menyediakan pengalaman pengembangan yang lebih baik. Salah satu fitur utama Dart adalah **Just-in-Time (JIT)** dan **Ahead-of-Time (AOT)** compilation, yang memungkinkan aplikasi berjalan cepat pada waktu eksekusi dan memiliki startup yang cepat. Dart juga dilengkapi dengan **Garbage Collection** otomatis, yang membantu pengelolaan memori dengan cara yang lebih efisien.

Selain itu, Dart juga mendukung **Null Safety**, yang memberikan kemampuan untuk mencegah error terkait null, sebuah masalah umum dalam pemrograman. Ini memungkinkan pengembang untuk menulis kode yang lebih aman dan bebas dari bug terkait nilai null. Fitur lainnya seperti **async/await** memungkinkan pengembangan aplikasi asinkron yang lebih efisien, sementara **Future** dan **Stream** digunakan untuk menangani operasi yang berjalan dalam latar belakang tanpa mengganggu responsivitas aplikasi.

2.3 Tipe Data dan Variabel di Dart

Dart memiliki berbagai tipe data yang digunakan untuk mendeklarasikan variabel dalam pemrograman. Sebagai bahasa yang berorientasi objek, Dart mendukung penggunaan tipe data primitif seperti angka, teks, dan nilai boolean, serta tipe data kompleks seperti koleksi dan objek. Pemahaman tentang tipe data dan cara mendeklarasikan variabel adalah langkah awal yang penting dalam pengembangan aplikasi menggunakan Dart.

Berikut adalah beberapa tipe data dasar yang sering digunakan dalam Dart:

1. **String**: Digunakan untuk menyimpan teks.
   ```
   String name = 'Flutter Developer';
   ```
2. **int**: Digunakan untuk menyimpan bilangan bulat.
   ```
   int age = 30;
   ```
3. **double**: Digunakan untuk menyimpan angka desimal.
   ```
   double height = 175.5;
   ```
4. **bool**: Digunakan untuk menyimpan nilai boolean (benar atau salah).
   ```
   bool isActive = true;
   ```

5. **var**: Dart juga memungkinkan penggunaan tipe data **var**, yang memungkinkan kompilator Dart untuk secara otomatis menentukan tipe data dari nilai yang diberikan.

```
var temperature = 36.6; // Tipe data
akan otomatis menjadi double
```

6. **dynamic**: Tipe data ini memungkinkan variabel untuk menyimpan nilai dengan tipe yang dapat berubah selama runtime.

```
dynamic message = 'Hello, Dart';
message = 2025; // Bisa diubah menjadi
tipe data lain
```

Contoh Penggunaan Variabel dan Tipe Data

```
void main() {
  // Mendeklarasikan variabel dengan tipe data
  String name = 'John Doe';
  int age = 25;
  double salary = 5000.50;
  bool isEmployed = true;

  // Menampilkan hasil
  print('Name: $name');
  print('Age: $age');
  print('Salary:
\$${salary.toStringAsFixed(2)}');
  print('Is Employed: $isEmployed');
}
```

Penjelasan:

- Pada kode di atas, variabel name, age, salary, dan isEmployed mendeklarasikan tipe data yang sesuai dengan nilai yang disimpan.
- Penggunaan tipe data yang tepat penting dalam memastikan aplikasi berjalan dengan efisien dan meminimalkan bug.

Dengan pemahaman yang kuat tentang tipe data dan cara mendeklarasikan variabel, Anda dapat mulai membuat program Dart yang lebih kompleks dan berguna dalam konteks pengembangan aplikasi Flutter.

2.4 Operasi Dasar dengan Dart

Dart menyediakan berbagai operasi dasar untuk bekerja dengan variabel, seperti operasi aritmatika, perbandingan, dan logika. Memahami operasi dasar ini sangat penting karena mereka sering digunakan dalam perhitungan, logika aplikasi, serta dalam pengolahan data di aplikasi Flutter. Di bawah ini adalah beberapa jenis operasi dasar yang sering digunakan dalam Dart.

1. Operasi Aritmatika

Operasi aritmatika digunakan untuk melakukan perhitungan matematis seperti penjumlahan, pengurangan, perkalian, pembagian, dan modulus.

```
int a = 10;
int b = 5;

print(a + b); // Penjumlahan: 15
print(a - b); // Pengurangan: 5
print(a * b); // Perkalian: 50
print(a / b); // Pembagian: 2.0
print(a % b); // Modulus: 0
```

2. Operasi Perbandingan

Operasi perbandingan digunakan untuk membandingkan dua nilai dan menghasilkan nilai boolean (true atau false). Ini berguna dalam struktur kontrol seperti if dan while.

```
int x = 10;
int y = 20;

print(x == y); // Sama dengan: false
print(x != y); // Tidak sama dengan: true
print(x > y);  // Lebih besar: false
```

```
print(x < y);  // Lebih kecil: true
print(x >= y); // Lebih besar atau sama
dengan: false
print(x <= y); // Lebih kecil atau sama
dengan: true
```

3. Operasi Logika

Operasi logika digunakan untuk memanipulasi nilai boolean, biasanya digunakan dalam pengkondisian dan loop.

```
bool isAdult = true;
bool hasPermission = false;

print(isAdult && hasPermission); // AND: false
print(isAdult || hasPermission); // OR: true
print(!isAdult); // NOT: false
```

Operasi penugasan digunakan untuk memberikan nilai kepada variabel.

```
int score = 10;
score += 5; // Sama dengan score = score + 5;
score -= 3; // Sama dengan score = score - 3;
score *= 2; // Sama dengan score = score * 2;
score ~/= 3; // Pembagian bulat (integer
division): score = score ~/ 3;
print(score); // Hasil akhir: 8
```

Contoh Penggunaan Operasi Dasar dalam Program

```
void main() {
  int num1 = 15;
  int num2 = 4;

  // Operasi Aritmatika
  print('Penjumlahan: ${num1 + num2}');
  print('Pembagian: ${num1 / num2}');
  print('Modulus: ${num1 % num2}');

  // Operasi Perbandingan
  print('Apakah num1 lebih besar dari num2?
${num1 > num2}');
```

```
  // Operasi Logika
  bool isAvailable = true;
  bool isDiscounted = false;
  print('Apakah produk tersedia dan diskon?
${isAvailable && isDiscounted}');
}
```

Penjelasan:

- **Operasi Aritmatika**: Digunakan untuk operasi dasar seperti penjumlahan dan pembagian.
- **Operasi Perbandingan**: Berguna untuk memeriksa apakah dua nilai sama atau lebih besar/kecil dari yang lainnya.
- **Operasi Logika**: Digunakan untuk memanipulasi nilai boolean, misalnya untuk mengecek dua kondisi apakah benar bersamaan atau tidak.
- **Operasi Penugasan**: Berguna untuk memberikan atau memodifikasi nilai variabel dengan lebih ringkas.

Dengan memahami dan menggunakan operasi dasar ini, Anda dapat dengan mudah menangani perhitungan dan logika dalam aplikasi Flutter yang sedang dikembangkan.

2.5 Struktur Kontrol

Struktur kontrol digunakan untuk mengendalikan alur eksekusi program berdasarkan kondisi tertentu. Dalam Dart, ada beberapa jenis struktur kontrol yang umum digunakan, yaitu **kondisional** dan **perulangan**. Dengan memahami struktur kontrol ini, Anda dapat menulis kode yang lebih dinamis dan responsif sesuai dengan kebutuhan aplikasi.

1. Kondisional (if, else, switch)

Struktur kontrol kondisional memungkinkan Anda untuk mengevaluasi ekspresi boolean dan menentukan jalur eksekusi program berdasarkan hasil evaluasi tersebut.

If-Else

Struktur if digunakan untuk menjalankan blok kode jika kondisi yang diberikan bernilai true. Anda juga dapat menambahkan blok else untuk menangani kasus lain jika kondisi tidak terpenuhi.

```
int age = 18;

if (age >= 18) {
  print('Anda sudah dewasa');
} else {
  print('Anda masih di bawah umur');
}
```

Else If

Anda dapat menggunakan else if untuk memeriksa lebih dari satu kondisi.

```
int score = 85;

if (score >= 90) {
  print('Nilai A');
} else if (score >= 80) {
  print('Nilai B');
} else {
  print('Nilai C');
}
```

Switch

switch digunakan untuk membandingkan nilai variabel dengan beberapa kondisi yang mungkin. Ini lebih efisien dibandingkan penggunaan banyak if-else ketika Anda memeriksa nilai variabel terhadap beberapa kondisi yang berbeda.

```
String day = 'Minggu';

switch (day) {
  case 'Senin':
    print('Hari pertama kerja');
    break;
```

```
  case 'Minggu':
    print('Hari libur');
    break;
  default:
    print('Hari biasa');
}
```

2. Pengulangan (Loops)

Dart mendukung beberapa jenis struktur pengulangan untuk menjalankan blok kode secara berulang-ulang berdasarkan kondisi tertentu. Pengulangan yang paling umum adalah for, while, dan do-while.

For Loop

for loop digunakan untuk mengulang kode dengan jumlah iterasi yang sudah diketahui sebelumnya.

```
for (int i = 1; i <= 5; i++) {
  print('Iterasi ke-$i');
}
```

While Loop

while loop digunakan untuk mengulang kode selama kondisi yang diberikan bernilai true.

```
int i = 1;
while (i <= 5) {
  print('Iterasi ke-$i');
  i++;
}
```

Do-While Loop

do-while loop mirip dengan while, tetapi kode akan dieksekusi setidaknya satu kali sebelum memeriksa kondisi.

```
int i = 1;
do {
  print('Iterasi ke-$i');
  i++;
} while (i <= 5);
```

3. Break dan Continue

- **Break** digunakan untuk keluar dari loop secara paksa sebelum loop selesai.
- **Continue** digunakan untuk melewati satu iterasi dalam loop dan melanjutkan dengan iterasi berikutnya.

```
for (int i = 1; i <= 5; i++) {
  if (i == 3) {
    break; // Menghentikan loop saat i = 3
  }
  print('Iterasi ke-$i');
}

for (int i = 1; i <= 5; i++) {
  if (i == 3) {
    continue; // Melewati iterasi saat i = 3
  }
  print('Iterasi ke-$i');
}
```

Contoh Penggunaan Struktur Kontrol dalam Program

```
void main() {
  int age = 20;

  // Kondisional if-else
  if (age >= 18) {
    print('Anda sudah cukup umur untuk membuat
akun.');
  } else {
    print('Anda terlalu muda untuk membuat
akun.');
  }

  // Switch case
  String day = 'Sabtu';

  switch (day) {
    case 'Senin':
      print('Mulai kerja!');
      break;
```

```
    case 'Sabtu':
      print('Waktunya bersantai!');
      break;
    default:
      print('Hari biasa.');
  }

  // For Loop
  print('Loop dengan for:');
  for (int i = 1; i <= 3; i++) {
    print('Iterasi ke-$i');
  }

  // While Loop
  print('Loop dengan while:');
  int i = 1;
  while (i <= 3) {
    print('Iterasi ke-$i');
    i++;
  }

  // Do-While Loop
  print('Loop dengan do-while:');
  int j = 1;
  do {
    print('Iterasi ke-$j');
    j++;
  } while (j <= 3);
}
```

Penjelasan:

- **Kondisional** (if, else, switch) digunakan untuk memeriksa kondisi dan mengeksekusi kode berdasarkan kondisi tersebut.
- **Looping** (for, while, do-while) digunakan untuk menjalankan blok kode berulang kali dengan berbagai cara, tergantung pada kebutuhan.

- **Break dan Continue** memberikan kontrol tambahan dalam loop untuk menghentikan atau melanjutkan iterasi sesuai kondisi tertentu.

Dengan memahami struktur kontrol ini, Anda dapat menulis program yang lebih dinamis dan fleksibel, yang dapat beradaptasi dengan berbagai kondisi yang ada pada aplikasi Flutter yang sedang Anda kembangkan.