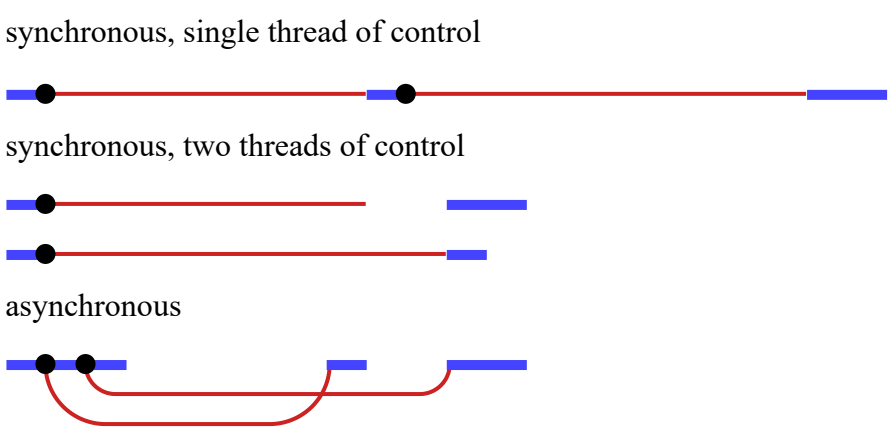
MATERI

Hari-9-Async, Future, Await

Di dalam dunia pemrograman dart terdapat dua cara dalam menjalankan program: Synchronous dan Asynchronous. Synchronous artinya program berjalan secara berurutan sedangkan Asynchronous artinya program berjalan bersama-sama.

Terkadang di dalam program yang kita buat terdapat suatu sintaks yang mengharuskan code pada baris tersebut untuk dijalankan terlebih dahulu sebelum menjalankan sintaks pada baris selanjutnya. Hal ini dikenal dengan istilah **blocking**. Sebaliknya **non-blocking** artinya program berjalan dengan mengeksekusi sintaks dari baris ke baris secara paralel (bersama-sama) .



Perhatikan contoh program di bawah ini:

```
import 'dart:async';


void main (){

  print("saya dijalankan pertama");
  var timer = Timer(Duration(seconds: 3), ()=>print('saya dijalankan terakhir'));
  print("saya dijalankan kedua");

}
```

Jika kita jalankan program di atas, maka yang akan tampil terlebih dahulu di print adalah “saya dijalankan pertama” dan yang kedua adalah coding terakhir dan yang dicetak terakhir adalah var timer, walaupun ditulisa kedua karena proses async yang teradapat pada dart.

Future

Future merupakan salah satu keyword yang disediakan oleh dart yang seperti namanya yaitu masa depan, jadi keyword ini gunanya untuk mengembalikan masa depan atau mengembalikan nilainya untuk waktu yang akan datang

contoh penggunaan future

```
void main() {
  fetchUserOrder();
  print('Ambil Datanya');
}


Future<void> fetchUserOrder() {
  return Future.delayed(Duration(seconds: 2),
      () => print('Selamat datang Peserta bootcamp flutter'));
}
```

Bekerja dengan async dan await

Keyword Async dan Awair menyediakan cara deklaratis untuk mendefinisikan fungsi asynchronous dan menggunakna valuenya. dalam menggunakan async dan await ingat pedoman dasar ini :

1. sebelumn menggunakan asynchronous terlebih dahulu tambahkan keywod async sebelum body
2. keyword await akan berfungsi apabila dia berada dalam fungsi async
3. dan berikut ini merupakan contoh yang mengkonversi dari fungsi synchronous ke dalam asynchronous

```
void main() async {··· }
```

jika fungsi memiliki tipe pengembalian yang di deklarasikan, maka perlu di perbaharui type menjadi Future<T>, dimana T adalah tipe nilai yang dikembalikan oleh fungsi, jika fungsi pengembaliannya tidak secara eksplisit maka tipe data akan menjadi Future<void>

```
Sekarang Anda memiliki fungsi async, Anda dapat menggunakan kata kunci await untuk menunggu hingga Future selesai:
```

```
print(await createOrderMessage());
```

Contoh Synchronous

```
String createOrderMessage() {
  var order = fetchUserOrder();
  return 'Your order is: $order';
}

Future<String> fetchUserOrder() =>
    // Imagine that this function is
    // more complex and slow.
    Future.delayed(
      Duration(seconds: 2),
      () => 'Large Latte',
    );

void main() {
  print('Fetching user order...');
  print(createOrderMessage());
}
```

Contoh Asynchronous

```
Future<String> createOrderMessage() async {
  var order = await fetchUserOrder();
  return 'Your order is: $order';
}

Future<String> fetchUserOrder() =>
    // Imagine that this function is
    // more complex and slow.
    Future.delayed(
      Duration(seconds: 2),
      () => 'Large Latte',
    );

Future<void> main() async {
  print('Fetching user order...');
  print(await createOrderMessage());
}
```

```
contoh asynchronous akan berbeda dalam tiga hal:

Jenis return/kembalian untuk createOrderMessage () berubah dari String ke Future <String>.

keyword async muncul sebelum body fungsi untuk createOrderMessage () dan main ().

keyword await muncul sebelum memanggil fungsi Asynchronous fetchUserOrder () dan createOrderMessage ().
```

Handling Error

untuk menangani handling error function async gunakan, try-catch jadi apabila succes akan masuk ke try dan jika error akan di tangani oleh catch

dan berikut ini merupakan contoh synctaq try catch :

```
try {
  var order = await fetchUserOrder();
  print('Awaiting user order...');
} catch (err) {
  print('Caught error: $err');
}
```

Try-Catch contoh di dalam program

```
    print('Awaiting user order...');
    print(order);
  } catch (err) {
    print('Caught error: $err');
  }
}

Future<String> fetchUserOrder() {
  // Imagine that this function is more complex.
  var str = Future.delayed(
      Duration(seconds: 4),
      () => throw 'Cannot locate user order');
  return str;
}

Future<void> main() async {
  await printOrderMessage();
}
```

Bekerja dengan synchronous dan asynchronous

dengan menggabungkan 2 blok ini kita dapat membuat ui kita menjadi lebih interaktif, shingga lebih memudahkan kita dalam menyajikan data yang interaktif, contoh penggunaan synchronous dan asynchronus sebagai berikut :

```
void main(List<String> args) async{
  var t = Titan(); // initialisasi t = object dari class titan

  print("zeke"); // mencetak zeke
  print(t.name); // akan mencetak string name yang pertama yaitu eren yeger
 await t.getName(); // masuk kedalam async await untuk mendelayed object di dalamnya
 //dan akan mencetak get nem [done]
  print(t.name); //mencetak grisha yeger karena sudah di masukan kedalam async nya
  print("rener"); // akan mecetak rener

  }

class Titan{
  String name = "Eren Yeger"; // mengisi value name eren yeger
  Future<void> getName()async{ // masuk kedalam FUture  dan object get name dan inisialisai async
    await Future.delayed(Duration(seconds: 3)); // memberi delayed pada object selama 3 detik
    name = "grisha"; //set name grisha
    print("get name [done]"); //cetak prin get name done
    }
}
```

Bekerja dengan future dan delayed tanpa async await

dengan menggunakan method future kita juga dapat membuat sebuah asynchrobnous dengan sama sama lebih mudah, berikut ini adalha contoh nya

```
Future delayedPrint(int seconds, String message){
  final duration = Duration(seconds : seconds);
  return Future.delayed(duration).then((value) => message);
}
Run | Debug
main(List<String> args){
print("roger");
  delayedPrint(2,"pirates"). then((status){
    print(status);
  });
  print("is");
}
```

Bekerja dengan async, await, future

```dart
void main(List<String> args) async {
  print("Persiapan. mulai");
  print(await line());
  print(await line2());
}

Future<String> line() async {
  String greeting = "pagiku cerah matahari bersinar...";
  return await Future.delayed(Duration(seconds: 5), () => (greeting));
}
Future<String> line2() async {
  String greeting = "kugendong tas merahaku,  di pundak .......";
  return await Future.delayed(Duration(seconds: 3), () => (greeting));
}
```

Rating - Feedback

Berikan Rating pada posting ini:

0

Berikan kritik dan saran..

Submit