



Modul Praktikum **Pemrograman Mobile**



INSTALASI FLUTTER & PENGENALAN DART

DASAR TEORI



FLUTTER

Flutter adalah platform yang digunakan para developer untuk membuat aplikasi multiplatform hanya dengan satu basis coding (codebase). Artinya, aplikasi yang dihasilkan dapat dipakai di berbagai platform, baik mobile Android, iOS, web, maupun desktop. Flutter sudah digunakan oleh banyak apps di seluruh dunia seperti Alibaba dan Ebay. Flutter juga bersifat open source. Bahasa pemrograman yang digunakan oleh Flutter adalah Dart.

Kelebihan Framework Flutter:

1. Pengembangan Aplikasi Lebih Produktif

Flutter menyediakan user interface widgets yang dikembangkan oleh Google dengan kualitas coding yang tinggi. Widgets ini bisa Anda kustomisasi dengan mudah, tanpa perlu menulis coding dari nol. Flutter memiliki fitur hot-reload yang memungkinkan Anda untuk melihat menampilkan hasil coding dengan kilat. Dengan hot-reload, Anda dapat dengan mudah melihat perubahan kode pada perangkat, tanpa perlu menunggu restart dan kehilangan state.

2. UI yang menarik

Anda bisa memanfaatkan koleksi widget untuk tampilan aplikasi, seperti layout, navigasi, animasi, style tema, font, hingga pengaturan tampilan scrolling. Flutter juga dapat mengontrol setiap piksel yang ada di layar, sehingga memudahkan dalam membuat animasi. Flutter menyediakan banyak komponen material design yang dapat berjalan baik pada Android, iOS, dan web.

3. Performa seperti Aplikasi Native

Walaupun aplikasi Flutter bersifat multiplatform, performa aplikasi yang dihasilkan tak kalah dari aplikasi native. Flutter menggunakan graphic engine bernama Skia-2D yang juga digunakan pada Chrome dan Android. Hasilnya,



aplikasi tetap bisa ditampilkan dengan baik di berbagai platform. Tak hanya tampilan desain yang tetap menarik, performa untuk transisi, scrolling, loading data pun tetap optimal di segala platform. Selain memang menyediakan aplikasi dengan performa optimal, Flutter juga loyal membagikan tips dan trik di official youtube channel mereka agar developer mampu membuat aplikasi yang ringan.



Dart

DART

Dart adalah sebuah bahasa pemrograman yang dikembangkan oleh Google dan merupakan bahasa pemrograman resmi untuk Flutter, sebuah UI toolkit dan aplikasi multiplatform dari Google. Dart merupakan bahasa yang wajib Anda kuasai untuk mengembangkan aplikasi Flutter. Dart sangat mirip dengan Bahasa pemrograman Java. Maka dari itu, disarankan sebelum mengambil mata kuliah ini untuk mengambil mata kuliah Pemrograman Basis Obyek dimana didalamnya akan dijelaskan lebih dalam mengenai OOP dan Java.



INSTALASI FLUTTER

1. Sebelum install Flutter, install terlebih dahulu Java JDK dan Android Studio. Unduh zip Flutter dari halaman download Flutter SDK pada link berikut:

<https://docs.flutter.dev/get-started/install/windows>.

Pilih versi terbaru pada website berikut.

The screenshot shows the Flutter documentation website for Windows installation. The page title is "Windows install" and the breadcrumb is "Get started > Install > Windows". The main heading is "System requirements". Below it, a note states: "To install and run Flutter, your development environment must meet these minimum requirements:". The requirements are listed as follows:

- **Operating Systems:** Windows 10 or later (64-bit), x86-64 based.
- **Disk Space:** 1.64 GB (does not include disk space for IDE/tools).
- **Tools:** Flutter depends on these tools being available in your environment.
 - [Windows PowerShell 5.0](#) or newer (this is pre-installed with Windows 10)
 - [Git for Windows 2.x](#), with the **Use Git from the Windows Command Prompt** option.

Below the requirements, a note says: "If Git for Windows is already installed, make sure you can run `git` commands from the command prompt or PowerShell."

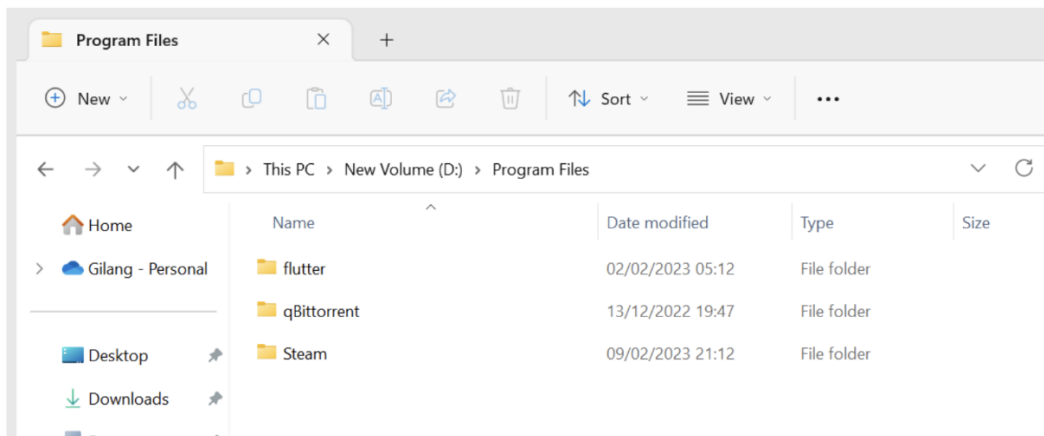
The next section is "Get the Flutter SDK". It contains two steps:

1. Download the following installation bundle to get the latest stable release of the Flutter SDK:
[flutter_windows_3.7.3-stable.zip](#)
2. Extract the zip file and place the contained `flutter` in the desired installation location for the Flutter SDK (for example, `C:\src\flutter`).

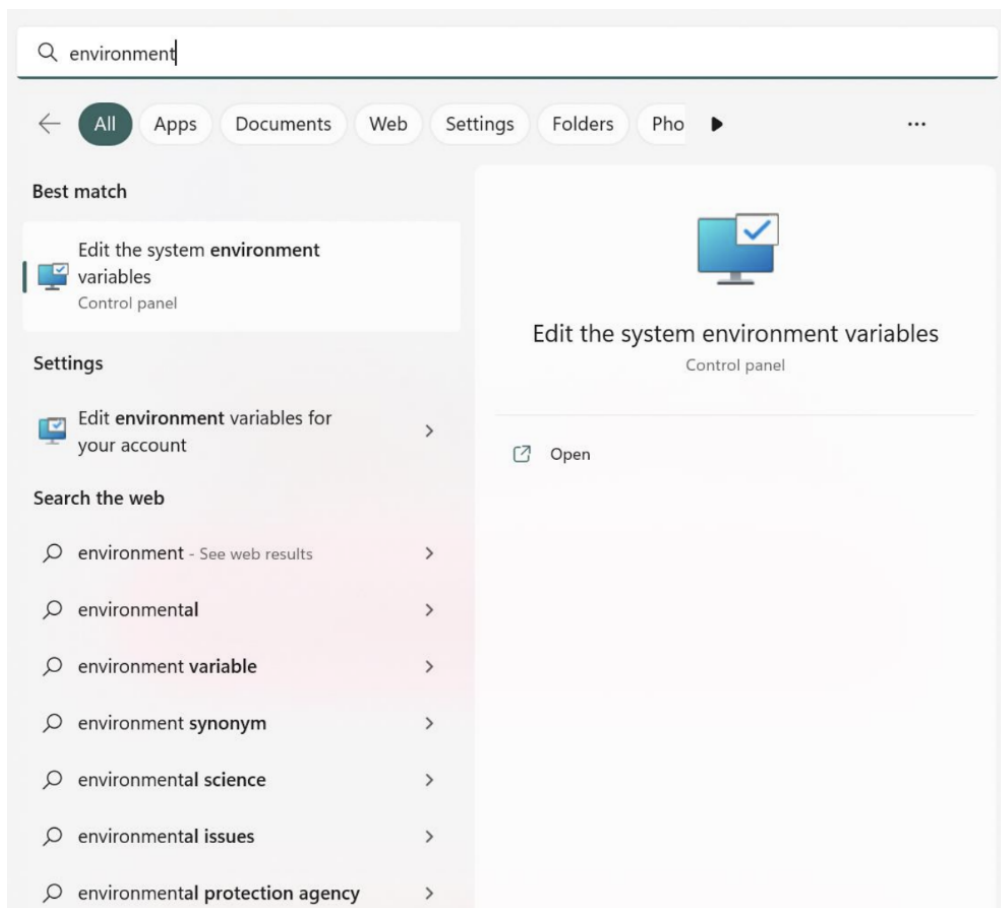
The right sidebar contains a "Contents" section with links to "System requirements", "Get the Flutter SDK", "Update your path", "Run flutter doctor", "Android setup", "Install Android Studio", "Set up your Android device", "Set up the Android emulator", "Agree to Android Licenses", "Windows setup", "Additional Windows requirements", "Web setup", and "Next step".



2. Ekstrak zip dan tempatkan folder Flutter pada lokasi yang anda inginkan. Jangan taruh folder tersebut di dalam *C:\Program Files* atau folder yang memerlukan izin administrator. (Misalnya, Saya tempatkan di *D:\Program Files*)

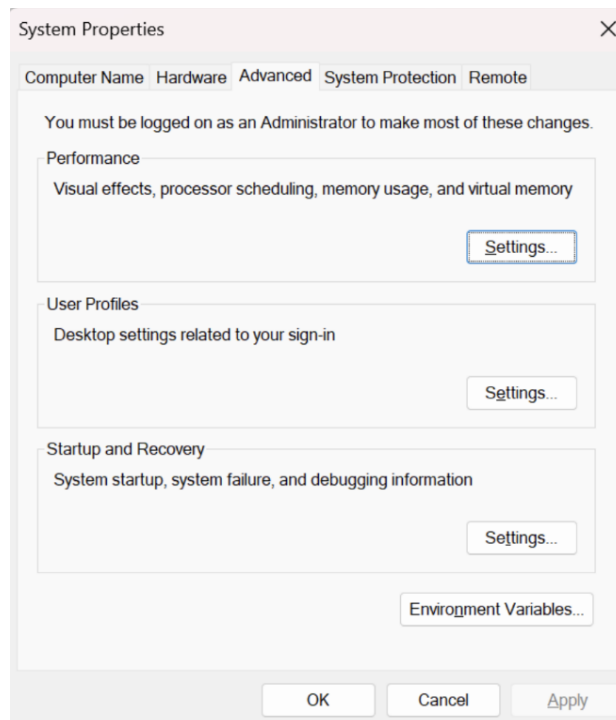


3. Selanjutnya kita masukkan lokasi flutter ke dalam path. Ketikkan pada windows search "environment" dan pilih *Edit the system environment variable*.

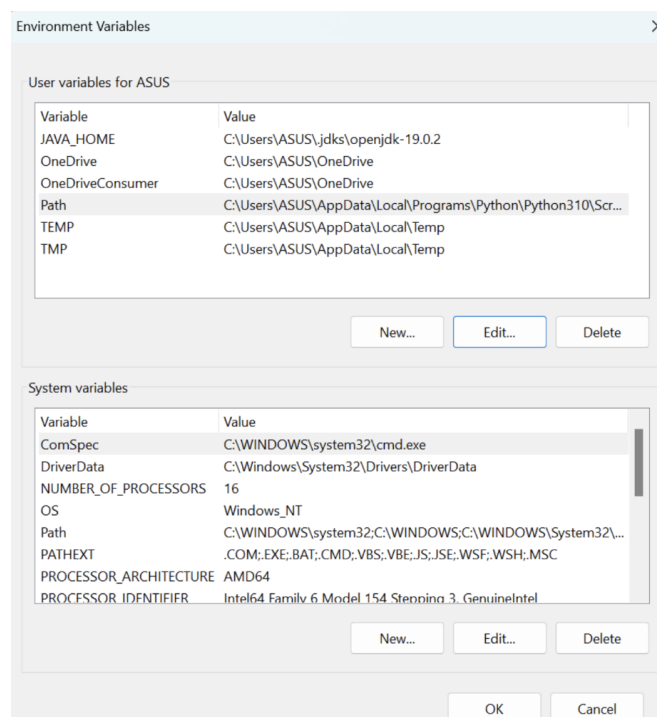




4. Klik tombol *Environment Variables*.

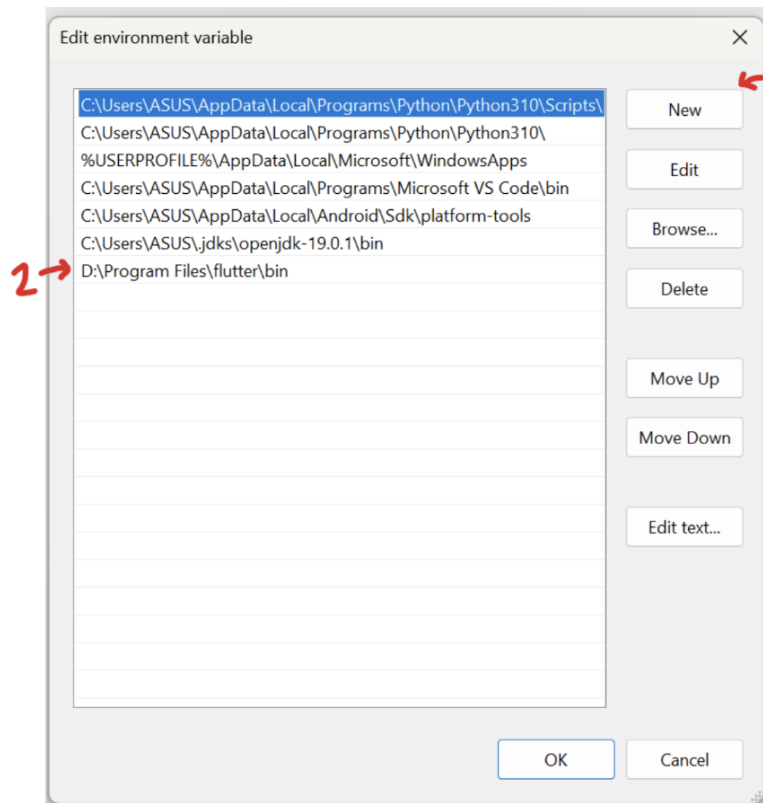


5. Di bagian *user variables* pilih *Path* lalu klik *edit*.





6. Klik *new* lalu paste lokasi flutter/bin.



7. Setelah selesai kita masukkan lokasi flutter ke path. Buka CMD lalu ketikkan command “flutter doctor”. Periksa output dengan cermat untuk perangkat lunak lain yang mungkin perlu Anda instal atau melakukan sesuatu lebih lanjut (ditunjukkan dalam teks tebal) sampai semua hijau/tanda centang kecuali visual studio.

```
C:\Users\ASUS>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.7.1, on Microsoft Windows [Version 10.0.22621.1194], locale en-ID)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 33.0.1)
[✓] Chrome - develop for the web
[X] Visual Studio - develop for Windows
    X Visual Studio not installed; this is necessary for Windows development.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[✓] Android Studio (version 2022.1)
[✓] IntelliJ IDEA Ultimate Edition (version 2022.3)
[✓] VS Code (version 1.75.0)
[✓] Connected device (3 available)
[✓] HTTP Host Availability

! Doctor found issues in 1 category.
```



8. Jika ada prompt untuk menerima android licenses seperti dibawah, ketikkan command "flutter doctor --android-licenses".

```
C:\Users\Kanishka M>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.2.0, on Microsoft Windows [Version 10.0.19042.985], locale en-US)
[!] Android toolchain - develop for Android devices (Android SDK version 30.0.3)
    X Android license status unknown.
      Run 'flutter doctor --android-licenses' to accept the SDK licenses.
      See https://flutter.dev/docs/get-started/install/windows#android-setup for more details.
[✓] Chrome - develop for the web
[!] Android Studio (not installed)
[✓] Connected device (2 available)

! Doctor found issues in 2 categories.
```

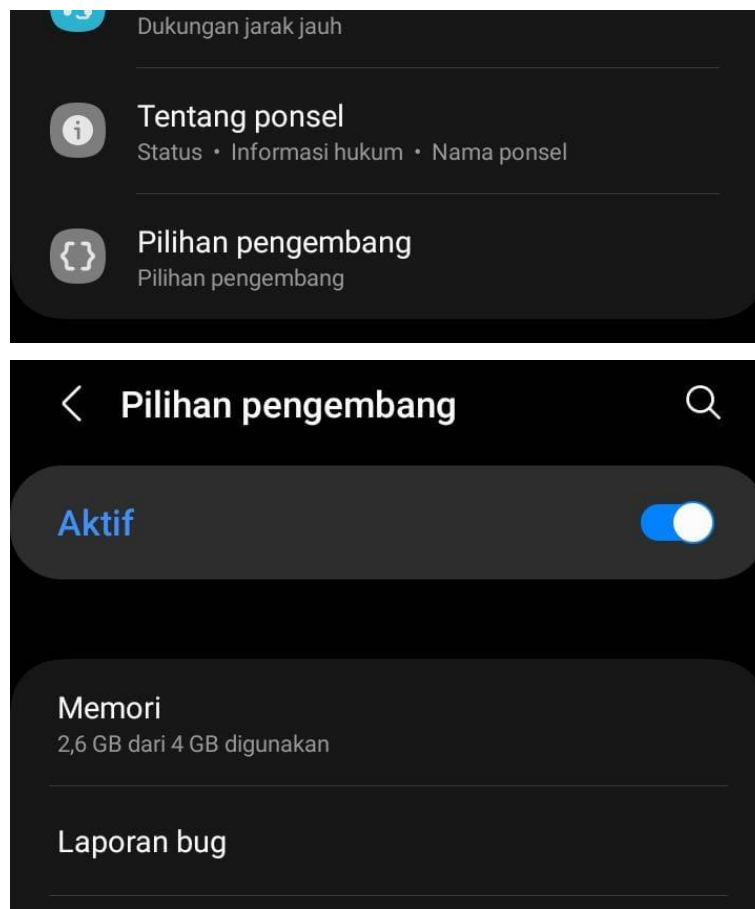



MENGHUBUNGKAN KE PERANGKAT EKSTERNAL

Jika dirasa perangkat pengembang anda tidak mampu menjalankan Android Studio sepenuhnya, anda dapat menggunakan perangkat eksternal seperti handphone untuk menjalankan pengujian dan proses debug melalui koneksi Android Debug Bridge (ADB).

MENYIAPKAN PERANGKAT EKSTERNAL KE MODE “PENGEMBANG”

Di perangkat, buka aplikasi **Setelan**, pilih **Opsi developer***, lalu aktifkan **Proses debug USB** (jika ada).



Pengaktifan mode pengembang pada OneUI

Note:

** setiap perangkat eksternal memiliki cara yang berbeda-beda untuk mengaktifkan opsi developer untuk menjaga data dan privasi anda, silahkan cek pada website untuk langkah lebih detail terkait perangkat anda.*



MENYIAPKAN SISTEM AGAR MENDETEKSI PERANGKAT EKSTERNAL


1. Windows

Instal driver USB untuk ADB (jika ada). Untuk panduan penginstalan dan link ke driver OEM, baca dokumen [Menginstal driver USB OEM](#).

2. macOS

Konfigurasi tambahan tidak diperlukan.

MENGHUBUNGKAN KE PERANGKAT MENGGUNAKAN USB

Setelah Anda siap dan terhubung melalui USB, klik  **Run** di Android Studio untuk [mem-build dan menjalankan aplikasi](#) di perangkat.

Anda juga dapat menggunakan [adb](#) untuk memberikan perintah, seperti berikut:

- Pastikan bahwa perangkat Anda terhubung dengan menjalankan perintah `adb devices` dari direktori `android_sdk/platform-tools/` Anda. Jika sudah terhubung, perangkat Anda akan tercantum pada daftar.
- Jalankan [perintah adb](#) dengan flag `-d` untuk menargetkan perangkat Anda.



PENGENALAN DART

Jika ingin mencoba menulis program Dart tanpa instalasi atau konfigurasi apa pun, maka DartPad adalah pilihan yang tepat: <https://dartpad.dev>

HELLO WORLD

Pertama-tama kita buat terlebih dahulu hello world.

```
void main() {  
  print("Hello World!");  
}
```

Sederhana sekali bukan. Di dalam dart, program dimulai eksekusi pada fungsi main seperti java. Tetapi tidak seperti Java, setiap fungsi atau variabel tidak harus berada di dalam kelas.

VARIABLE & DATA TYPE

Kita lakukan penambahan a + b di dalam dart.

```
void main() {  
  int a = 2;  
  int b = 5;  
  int c = a + b;  
  print("a + b = $c");  
}
```

Hasil:

Console

a + b = 7

Dart merupakan Bahasa *statically typed*. Yang berarti, sebuah variabel memiliki tipe data pada saat *compile time*. Di dalam dart, kita bisa tidak mendeklarasikan tipe data variabel dengan menggunakan keyword var seperti berikut.

```
void main() {  
  var a = "Ini String";  
  print(a.runtimeType);  
}
```



Hasil:

```
Console  
  
String
```

Kita gunakan `.runtimeType` untuk mendapatkan tipe data dari sebuah variabel. Bisa dilihat disini var otomatis akan meng-assign tipe data variabel berdasarkan nilai.

FUNCTION

Kita buat sebuah fungsi pertambahan.

```
int add(int a, int b){  
    return a + b;  
}
```

Sama seperti Java, kita mulai dengan tipe data kembalian fungsi, nama fungsi, parameter lalu kode blok fungsi. Untuk cara penggunaan fungsi adalah sebagai berikut.

```
void main() {  
    print(add(2, 5));  
}
```

Hasil:

```
Console  
  
7
```

Misalnya anda memiliki banyak parameter atau parameter anda sangat panjang, anda bisa tambahkan koma tambahan di dalam penggunaan fungsi agar terformat lebih rapi. Contoh:

```
void introduction(String name, int age, String job) {  
    print("Hello, my Name is $name");  
    print("My Age is $age");  
}
```



```
print("I work as a $job");
}
void main() {
    introduction("Nurhaki Cahya", 21, "Department Store Clerk",);
}
```

Bila kita klik tombol format:

```
1 void introduction(String name, int age, String job) {
2     print("Hello, my Name is $name");
3     print("My Age is $age");
4     print("I work as a $job");
5 }
6
7 void main() {
8     introduction("Nurhaki Cahya", 21, "Department Store Clerk",);
9 }
10
```

Maka line 8 akan otomatis terformat seperti berikut:

```
void introduction(String name, int age, String job) {
    print("Hello, my Name is $name");
    print("My Age is $age");
    print("I work as a $job");
}

void main() {
    introduction(
        "Nurhaki Cahya",
        21,
        "Department Store Clerk",
    );
}
```

NULL SAFETY

Dart memiliki null safety, yaitu fitur dimana variabel harus ditandai dengan '?' (tanda tanya) di depan tipe data agar bisa mengandung nilai null, yang mencegah *exception* yang sangat populer yaitu NPE (*NullPointerException*).

Contoh:

```
void main() {
```



```
String? firstChild;  
print(firstChild);  
}
```

Hasil:

```
Console  
  
null
```

Di dalam dart ada beberapa operator yang menangani variabel null atau nilai null. Pertama yaitu ?? (null-coalescing operators) yang digunakan untuk mengembalikan nilai alternatif jika nilai berisi null.

Contoh:

```
void main() {  
  String? firstChild;  
  print("Nama Anak Pertama: ");  
  print(firstChild ?? "Nggak Punya Anak");  
}
```

Hasil:

```
Console  
  
Nama Anak Pertama:  
Nggak Punya Anak
```

Jika variabel firstChild tidak null, maka akan seperti berikut:

```
void main() {  
  String? firstChild = "Gatot Subroto";  
  print("Nama Anak Pertama: ");  
  print(firstChild ?? "Nggak Punya Anak");  
}
```



Hasil:

```
Console

Nama Anak Pertama:
Gatot Subroto
```

Yang kedua ada '?' (*conditional property access operator*) yang digunakan jika anda ingin memanggil *method* atau *property* dari variabel *nullable*. Jadi jika variabel tersebut null maka kembalikan null jika tidak maka panggil *method* atau *property* tersebut.

Contoh:

```
void main() {
    String? firstChild;
    print("Nama Anak Pertama: ");
    print(firstChild?.toUpperCase());
}
```

Hasil:

```
Console

Nama Anak Pertama:
null
```

Jika variabel `firstChild` tidak null, maka akan seperti berikut:

```
void main() {
    String? firstChild = "Harry Potter";
    print("Nama Anak Pertama: ");
    print(firstChild?.toUpperCase());
}
```

Hasil:



Console

Nama Anak Pertama:
HARRY POTTER

Masih banyak null safety lainnya yang bisa anda baca [disini](#).

NAMED PARAMETER & POSITIONAL PARAMETER

Positional Variable

```
int add(int a, int b)
```

Penggunaan

```
int c = add(1, 2);
```

Named Parameter

```
int add({int? a, int? b})
```

Penggunaan

```
int c = add(a:1, b:2);
```

Dart memiliki 2 cara untuk mendeklarasikan parameter, *Named Parameter* dan *Positional Parameter*. *Positional Parameter* adalah cara pada umumnya untuk mendeklarasikan parameter, sedangkan *Named parameter* adalah parameter yang harus disebutkan namanya pada saat digunakan dan boleh saja tidak digunakan.

Named Parameter ditandai dengan dikurung oleh { } (kurung kurawal). *Named Parameter* bisa digunakan bersamaan dengan *Positional Parameter*. Tujuan *Named Parameter* adalah agar kode lebih jelas dan mudah dibaca serta urutan parameter yang bisa diacak.

Contoh:

```
void introduction(String name, {int? age, String? job}) {  
  print("Hello, my Name is $name");  
  if(age != null){  
    print("My Age is $age");  
  }  
  if(job != null){  
    print("I work as a $job");  
  }  
}  
void main() {
```




```
introduction(  
    "Nurhaki Cahya",  
    job:"Department Store Clerk",  
);  
}
```

Hasil:

Console

```
Hello, my Name is Nurhaki Cahya  
I work as a Department Store Clerk
```

Misalnya, anda ingin mendeklarasikan *Named Parameter* tetapi parameter tersebut harus diisi maka gunakan keyword *required*.

Contoh:

```
void introduction(String name, {required int age, required String job}) {  
    print("Hello, my Name is $name");  
    print("My Age is $age");  
    print("I work as a $job");  
}  
void main() {  
    introduction(  
        "Nurhaki Cahya",  
        age:21,  
        job:"Department Store Clerk",  
    );  
}
```

Hasil:

Console

```
Hello, my Name is Nurhaki Cahya  
My Age is 21  
I work as a Department Store Clerk
```

COLLECTION

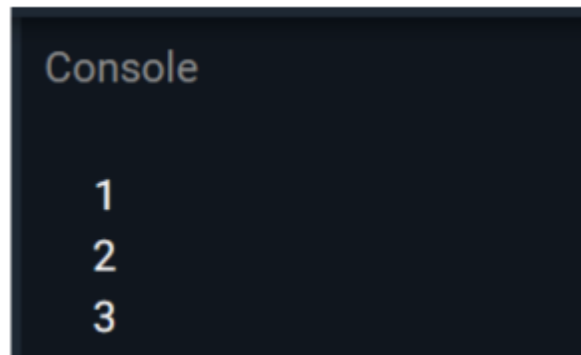


Collections adalah sebuah kumpulan objek yang direpresentasikan sebagai sebuah unit. Beberapa collections yang biasa digunakan di dalam dart yaitu:

- List: kumpulan objek yang terindex. Objek bisa diakses menggunakan indexnya. Contoh:

```
void main() {  
  List<int> nilai = [1, 2, 3];  
  for(int i in nilai){  
    print(i);  
  }  
}
```

Hasil:



```
Console  
  
1  
2  
3
```

- Set: kumpulan objek yang hanya bisa terjadi sekali saja. Contoh:

```
void main() {  
  Set<int> nilai = {1, 2, 2, 3, 4, 4};  
  for (int i in nilai) {  
    print(i);  
  }  
}
```

Hasil:



Console

1
2
3
4

- Map: pemetaan dari keys unik ke nilai yang terkait. Contoh:

```
void main() {  
    Map<int, String> angka = {  
        0:"No1",  
        1:"Satu",  
        2:"Dua",  
        3:"Tiga"  
    };  
    for(int i = 0; i<4;i++){  
        print(angka[i]);  
    }  
}
```

Hasil:

Console

No1
Satu
Dua
Tiga



CLASS, PROPERTY & METHOD

Class adalah cetak biru atau *blueprint* dari object. Class digunakan hanya untuk membuat kerangka dasar. Yang akan kita pakai nanti adalah hasil cetakan dari class, yakni **object**.

Contoh:

```
class Car{  
    // isi dari class  
}
```

Dalam bahasa Dart, penulisan class diawali dengan keyword *class*, kemudian diikuti dengan nama dari class tersebut. Biasanya, nama class ditulis menggunakan *PascalCase* atau *UpperCamelCase*. Yakni setiap kata diawali dengan huruf besar, termasuk kata pertama.

Property (atau disebut juga dengan atribut atau field) adalah data yang terdapat dalam sebuah class.

Contoh:

```
class Car{  
    int? seats;  
    String? name;  
    int? wheels;  
}
```

Method adalah fungsi yang berada di dalam Class.

Contoh:

```
class Car{  
    int? seats;  
    String? name;  
    int? wheels;  
  
    void vroom(){  
        print("Vroom Vroom...");  
    }  
}
```

Di dalam Dart hanya memiliki 2 Access Modifier yaitu **public** dan **private**. Cara mendeklarasikan sebuah class, property atau method sebagai private adalah dengan



menambahkan '_' (underscore) di depan nama class, property atau method tersebut. Class, property atau method otomatis memiliki access modifier public.

Misalnya kita ingin mendeklarasikan property wheels sebagai private maka akan jadi sebagai berikut:

```
class Car{
  int? seats;
  String? name;
  int? _wheels; // wheels menjadi private

  void vroom(){
    print("Vroom Vroom...");
  }
}
```

CONSTRUCTOR

Di dalam Dart, jika anda ingin menginisialisasi property class anda menggunakan constructor. Gunakan *this.{nama property}*, maka dart akan otomatis memasukkan nilai dari constructor ke dalam property. Hal ini hanya bisa dilakukan pada property public.

Contoh:

```
class Car{
  int? seats;
  String? name;
  int? wheels;

  Car({
    required this.seats,
    required this.name,
    required this.wheels,
  });
}
```

Jika anda ingin mengubah nilai parameter sebelum dimasukkan ke dalam property maka anda bisa gunakan member *initializer*. Misalnya, di dalam kelas Car yang kita punyai, kita ingin membuat property wheels menjadi private. Maka kita akan lakukan seperti ini:



```
class Car{
  int? seats;
  String? name;
  int? wheels;

  Car({
    required this.seats,
    required this.name,
    required this.wheels,
  }): this._wheels = wheels;
}
```