

# Modul Praktikum Flutter

## Bagian Dua: UI & Routing (part 1)



Yudi Wibisono ([yudi@upi.edu](mailto:yudi@upi.edu))  
Ilmu Komputer, Universitas Pendidikan Indonesia ([cs.upi.edu](http://cs.upi.edu))

versi: Beta; Feb 2024



<http://creativecommons.org/licenses/by-nc-sa/3.0/>

Modul ini bebas di-copy, didistribusikan, ditransmit dan diadaptasi/modifikasi/diremiks dengan syarat tidak untuk komersial, pembuat asal tetap dicantumkan dan hasil modifikasi dishare dengan lisensi yang sama.

# Pendahuluan

Flutter adalah framework atau SDK yang dapat digunakan untuk mengembangkan aplikasi di berbagai platform seperti Android, iOS, Web, Windows, Linux, MacOS. Pengembang utama Flutter adalah Google dan pertama kali di-release pada tahun 2017.

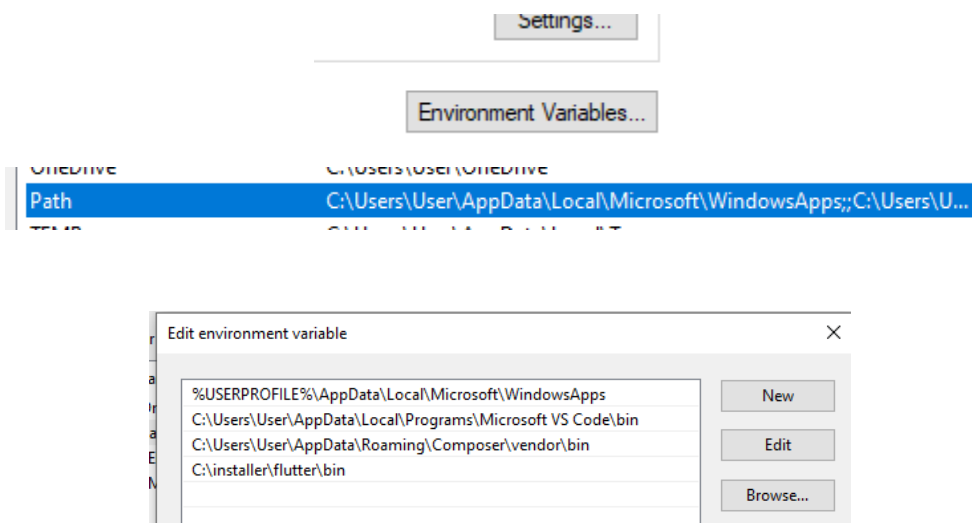
Beberapa alasan memilih Flutter untuk pengembangan multiplatform adalah:

1. Satu codebase untuk semua platform (Android, iOS, Web). Beberapa framework lain memerlukan tambahan code yang spesifik untuk setiap platform.
2. Framework yang relatif baru muncul sehingga tidak terbebani dengan arsitektur masa lalu dan dapat menggunakan teknik terbaik seperti declarative UI dan hot reload.
3. Mendapat dukungan Google. Selama dukungan ini tetap ada, Flutter akan tetap ada. Tapi ini sekaligus menjadi salah satu titik lemah Flutter.
4. Memiliki kinerja tinggi karena menggunakan Dart. Code Dart di-compile ke native code sehingga memiliki kinerja tinggi. Hot reload membantu programmer saat pengembangan.
5. Flutter menggambar sendiri UI-nya pixel demi pixel. Ini meningkatkan kinerja dan memberi kebebasan untuk menggambar komponen UI (widget) tanpa terikat API dari platform.
6. Widget dapat dikustomisasi.

## Instalasi

Flutter memerlukan SDK (Software Development Kit) yang perlu diinstall, ikuti petunjuk di: <https://docs.flutter.dev/get-started/install> sesuai dengan sistem operasi masing-masing.

Jika menggunakan Windows, download dan install flutter SDK pada <https://docs.flutter.dev/get-started/install/windows> (untuk OS Windows). Ekstrak SDK, lalu set path di environment variable ke tempat SDK ini diekstrak (gambar bawah)



Close semua terminal (command prompt), lalu panggil `flutter doctor` untuk memeriksa kelengkapan.

Contoh hasilnya akan seperti ini:

```
PS C:\Users\stei> flutter doctor
Doctor summary (to see all details, run
[✓] Flutter (Channel stable, 2.8.1, on
[✗] Android toolchain - develop for And
  ✗ Unable to locate Android SDK.
    Install Android Studio from: http
    On first launch it will assist yo
    (or visit https://flutter.dev/doc
    If the Android SDK has been insta
    `flutter config --android-sdk` to
[✓] Chrome - develop for the web
[!] Android Studio (not installed)
[✓] VS Code (version 1.63.2)
[✓] Connected device (2 available)
```

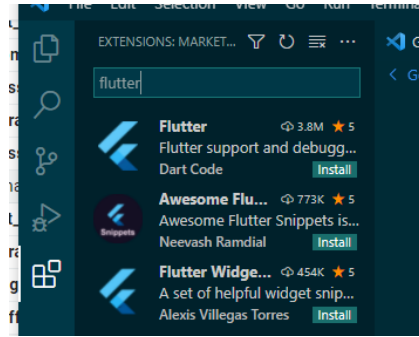
Untuk komputer Mac, contoh hasilnya (ada tambahan Xcode)

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.7.3, on macOS 13.1 22C65 darwin-arm64, loca
le en-ID)
[✓] Android toolchain - develop for Android devices (Android SDK version 3
3.0.0)
[✗] Xcode - develop for iOS and macOS
  ✗ Xcode installation is incomplete; a full installation is necessary f
or iOS development.
    Download at: https://developer.apple.com/xcode/download/
    Or install Xcode via the App Store.
    Once installed, run:
      sudo xcode-select --switch /Applications/Xcode.app/Contents/Develo
per
      sudo xcodebuild -runFirstLaunch
  ✗ CocoaPods not installed.
    CocoaPods is used to retrieve the iOS and macOS platform side's pl
ugin code that responds to your plugin usage on the Dart side.
    Without CocoaPods, plugins will not work on iOS or macOS.
    For more info, see https://flutter.dev/platform-plugins
    To install see https://guides.cocoapods.org/using/getting-started.ht
ml#installation-for-instructions.
[✓] Chrome - develop for the web
[✓] Android Studio (version 2022.1)
[✓] VS Code (version 1.75.1)
[✓] Connected device (2 available)
[✓] HTTP Host Availability

! Doctor found issues in 1 category.
```

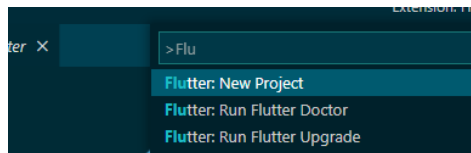
Pastikan minimal untuk Chrome sudah dapat berjalan. Untuk sekarang abaikan dulu Android Toolchain, Android Studio dan XCode karena kita hanya akan menjalankan di web.

Setelah itu install extension untuk Dart di Visual Studio Code. Pilih extension, ketik "Flutter" dan pilih extension Flutter (gambar bawah). Lewatkan langkah ini jika sebelumnya sudah menginstall saat mempelajari Dart di modul sebelumnya.

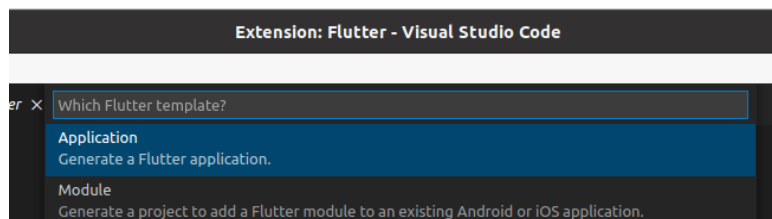


## Project Pertama

Kita akan buat program sederhana. Masuk ke command palette (ctrl+shift+P atau command+shift+P untuk Mac), ketik "Flutter" lalu "Flutter: New Project."

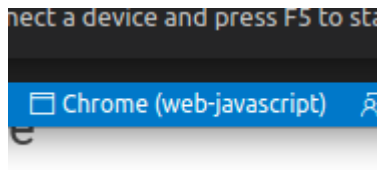


Kemudian pilih application



Pilih direktori tempat projet akan disimpan, dan Flutter akan membuatkan contoh project.

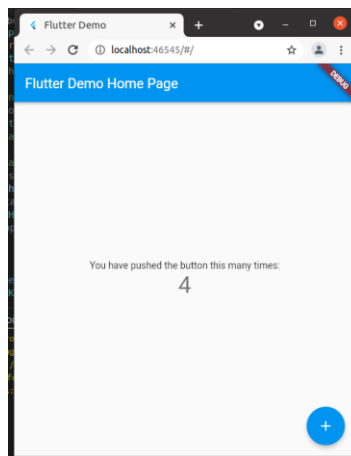
Pastikan di kanan bawah device Chrome sudah terhubung (gambar di bawah). Jika belum, klik dan pilih Chrome.



Jalankan program dengan F5 atau klik "run" di atas method main

```
Run | Debug | Profile
void main() {
  runApp(const MyApp());
}
```

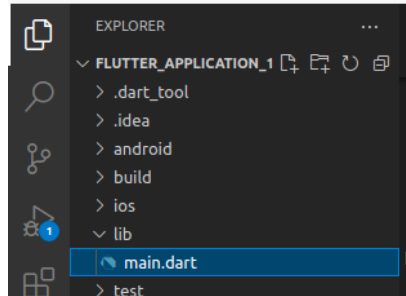
Hasilnya akan muncul seperti gambar ini



Catatan: App juga dapat dijalankan melalui command line, masuk ke direktori code lalu jalankan: `flutter run -d chrome`

Catatan: Jika saat dijalankan hanya muncul layar kosong di Chrome, jalankan melalui command line seperti di atas. Jika berhasil, berarti masalahnya ada pada extension Visual Studio Code. Masuk ke extension, dan pilih versi "pre-release".

Sekarang kita akan coba kemampuan hot reload Dart. Pilih explore lalu main.dart. Code app ada di file ini.



Scroll sekitar baris 99 (gambar bawah).

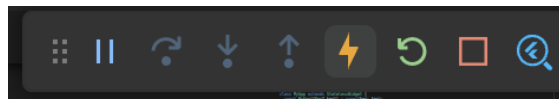
```

95 // horizontal).
96 mainAxisAlignment: MainAxisAlignment.center,
97 children: <Widget>[
98   const Text(
99     'You have pushed the button this many times:',
100   ), // Text
101   Text(

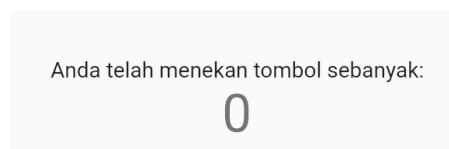
```

"You have pushed" dengan bahasa Indonesia misalnya.

Tekan **ctrl-S** untuk menyimpan, jika aplikasi tidak otomatis di-reload, tekan ctrl-F5 atau icon petir

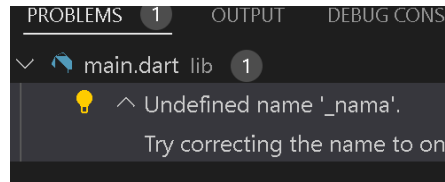


Dapat dilihat aplikasi langsung diupdate sebagai berikut (sebaiknya window VSC dan Chrome diletakkan berdampingan untuk memudahkan debug):



Ini adalah fitur hot reload yang dimungkinkan karena Dart mendukung teknik kompilasi JIT (Just in Time). JIT membuat perubahan code dapat langsung dilihat secara instan oleh developer. Ini akan sangat membantu terutama pada pembuatan user interface, karena perubahan dapat langsung dilihat dengan cepat. Sedangkan pada saat deploy, fitur AOT (ahead of time) membuat aplikasi berukuran kecil sehingga dapat di-load dan dijalankan dengan cepat.

Jika ada kesalahan, maka pada tab "PROBLEMS" akan muncul pesan kesalahan. Contoh pesan kesalahan dapat dilihat pada bagian bawah:



Pada main.dart ini terlihat bagaimana Class MyApp yang berisi title, theme, lalu class MyHomePage berisi layout (method build), floating button dan event (\_incrementCounter) yang dipanggil jika button ini diklik.

## Project Hello World

Sesuai tradisi, kita akan membuat aplikasi Helloworld yang paling sederhana. **Hapus** semua isi main.dart dan paste dengan code berikut.

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Hello App',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Hello'),
        ),
        body: const Center(
          child: Text('Hello World!'),
        ),
      ),
    );
  }
}
```

Tekan Ctrl-S untuk menyimpan dan jalankan.

Berikut penjelasan tiap bagian

Statement `import 'package:flutter/material.dart'` mengimpor berbagai widget untuk Material Design. Material Design adalah bahasa rancangan visual yang dibuat Google untuk Android, Flutter dan Web.

Catatan: bagi pembaca yang tertarik dengan UI/UX dapat melihat penjelasan lebih rinci tentang Material Design di <https://material.io>

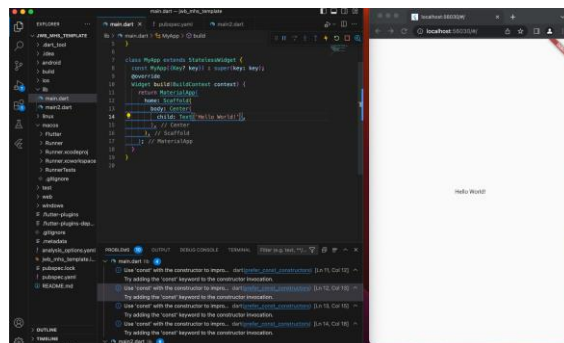
Method `runApp` pada `main()` berfungsi untuk meng-inflate widget `MyApp` (bayangkan boneka yang baru terbentuk saat ditiup) dan "menempelkannya" di layar. Widget berisi deskripsi user interface, sehingga perlu "ditiup" dan ditempelkan ke layar untuk menjadi komponen user interface yang dapat dilihat dan dimanipulasi user. Teknik ini disebut declarative UI.

Berikutnya `class MyApp extends StatelessWidget`. Ada dua jenis widget pada Flutter, `StatelessWidget` dan `StatefulWidget`. Sesuai namanya, `stateless` tidak dapat diubah setelah di-create (statik) sedangkan `statefull` dapat diubah setelah di-create. Karena widget ini hanya menampilkan "hello world" lalu selesai, maka `stateless` lebih tepat digunakan.

Widget tersusun secara hirarkis dalam bentuk pohon (widget tree). Method `build(BuildContext context)` dipanggil saat widget akan dimasukkan ke dalam widget tree. Di dalamnya, terdapat widget `MaterialApp` untuk membungkus aplikasi yang menggunakan komponen Material Design. Di dalamnya lagi kita tambahkan `Scaffold` untuk menampung widget `AppBar` (bar di bagian atas) dan `Body`. `Body` diisi dengan widget `Center` yang akhirnya berisi widget `Text`. Dapat dilihat struktur hirarkinya. Dari contoh sederhana ini, muncul julukan bahwa di Flutter *"every thing drawn on a screen is a widget"*

**Catatan:** bagi pengguna Ubuntu, untuk memudahkan debug dengan satu monitor, gunakan workspace. Satu workspace berisi visual studio berdampingan dengan window Chrome berisi app. Workspace yang lain berisi browser untuk membuka modul ini atau lainnya. Gunakan `ctrl-alt-panah kiri` dan `ctrl-alt-panah kanan` untuk berpindah antar workspace.

Bagi pengguna Mac: masuk ke mission control (swipe 3 atau 4 jari ke atas) dan tambahkan desktop ke-dua (button tambah di kanan atas) dan drag VSC dan Chrome ke dekstop tersebut (gambar bawah)





Bagi pengguna Windows: <https://support.microsoft.com/en-us/windows/multiple-desktops-in-windows-36f52e38-5b4a-557b-2ff9-e1a60c976434>

## Project Hello World [NAMA]

Berikutnya kita akan menambahkan fitur agar user dapat memasukkan nama, menekan tombol dan mengeluarkan pesan "Halo [Nama]"

Untuk project ini akan ada beberapa widget terkait interaksi dengan user: label input dan label output, textbox dan button. Widget Column digunakan untuk menampung ketiga widget ini:

```
Center(  
  child: Column(  
    mainAxisAlignment: MainAxisAlignment.min,  
    children: [  
      Text('Masukan Nama :'),  
      TextField( . . . ),  
      ElevatedButton(. . .),  
      Text('Halo $_nama'),  
    ],  
  )), //column center
```

Pada app sebelumnya kita menggunakan steless widget, atau widget tanpa state. State adalah informasi yang digunakan saat widget di-build dan informasi ini dapat berubah. Jadi widget berubah akibat perubahan state di dalam widget. Setiap state berubah, widget terkait harus melakukan rebuild untuk merefleksikan state yang baru di user interface. Hal ini disebut **react style of programming**.

Berbeda dengan dengan StatelessWidget yang hanya perlu meng-override build(), pada StatefulWidget kita perlu meng-override createState(). Kita juga perlu membuat class turunan State yang berfungsi menyimpan state (variabel-variabel yang terkait UI). Strukturnya sebagai berikut:

```
class MyApp extends StatefulWidget {  
  @override  
  MyAppState createState() {  
    return MyAppState(); //class state  
  }  
}
```

```
//class state untuk menyimpan state
class MyAppState extends State<MyApp> {
  String _nama = ""; //state berupa variabel yang disimpan

  @override
  Widget build(BuildContext context) { //build pindah ke sini
    return MaterialApp(
      ...
    );
  }
}
```

Pada build() kita tambahkan widget TextField untuk menerima input dari user dan ElevatedButton untuk tombol yang nantinya di-tap.

Isian user ditampung di state (MyAppState) dan ditampilkan saat tombol ditekan. Berikut potongan codenya:

Setiap user mengetikkan huruf, **onChanged** akan dipanggil dan \_nama akan terisi.

```
TextField(
  onChanged: (text) {
    _nama = text; //update state
  },
),
```

Pada saat widget button ditekan, **onPressed** dan **setState()** dipanggil. Method **SetState()** akan membuild ulang widget dengan state terbaru (melakukan refresh).

```
ElevatedButton(
  onPressed: () { //button ditap
    setState(({})); //refresh, rebuild widget untuk menampilkan state
  },
),
```

Code lengkapnya adalah sebagai berikut:

```
import 'package:flutter/material.dart';
void main() {
  runApp(const MyApp());
}

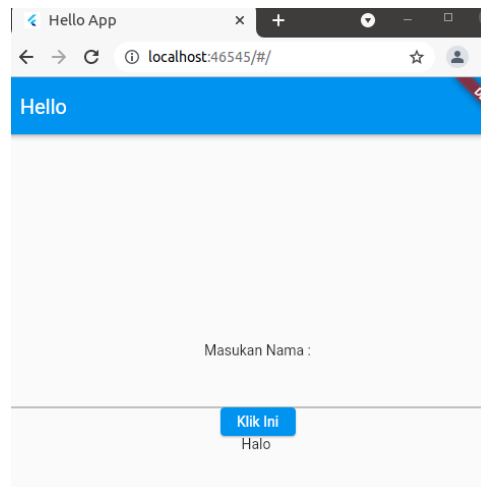
class MyApp extends StatefulWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  MyAppState createState() {
    return MyAppState();
  }
}
```

```

class MyAppState extends State<MyApp> {
  String _nama = "";
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Hello App',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Hello'),
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.min,
            children: [
              Text('Masukan Nama :'),
              TextField(
                onChanged: (text) {
                  _nama = text;
                },
              ),
              ElevatedButton(
                onPressed: () {
                  setState( () {} ); //refresh
                },
                child: const Text('Klik Ini'),
              ),
              Text('Halo $_nama'), //hasil disini
            ],
          ), //column center
        ), //Scaffold
      ); //Material APP
    }
  }
}

```

Hasilnya akan seperti ini, coba masukkan nama dan tap buttonnya.



Pendekatan ini tidak efisien dalam kasus ini karena setiap user mengetikkan karakter maka onChanged dipanggil, padahal kita tidak memerlukan penanganan per karakter

(latihan: pada kondisi apa kita perlu penanganan event update per karakter?)

Solusinya adalah menggunakan Controller. Controller dapat digunakan untuk mengakses nilai dari sebuah Widget. TextEditingController digunakan untuk mengakses nilai TextEditing

Tambahkan TextEditingController pada class state

```
class MyAppState extends State<MyApp> {  
  final textEditController = TextEditingController();  
  String _nama = "";  
  
  @override  
  void dispose() {  
    // controller dibersihkan saat widget dibuang  
    textEditController.dispose();  
    super.dispose();  
  }  
  ...  
}
```

Hubungkan textfield dengan controllernya, event onChanged **dapat dihapus**. Hati-hati jangan lupa koma setelah nilai atribut.

```
TextField(  
  controller: textEditController, //controller  
) ,
```

Setiap button ditekan, maka update variabel **\_nama** dilakukan melalui controller

```
ElevatedButton(  
  onPressed: () {  
    setState(() {  
      _nama = textEditController.text; //akses text  
    }); //refresh  
  },  
  child: const Text('Klik Ini'),  
) ,
```

Code lengkap: <https://pastebin.com/raw/a2hTeyJa>

### Latihan 1:

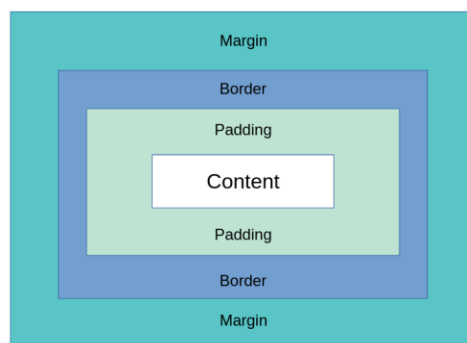
Buat app yang menerima panjang dan lebar persegi panjang, kemudian menampilkan luas dan keliling persegi panjang tersebut.

Petunjuk: gunakan `int.parse(string)` atau `double.parse(string)` untuk mengkonversi string menjadi integer atau double.

## Padding

Pada app sebelumnya, jarak antar widget masih terlalu berdekatan dan font belum memiliki style.

Padding adalah salah satu cara menambahkan jarak antar widget. Hubungan antara padding, border dan margin dapat dilihat pada gambar di bawah.

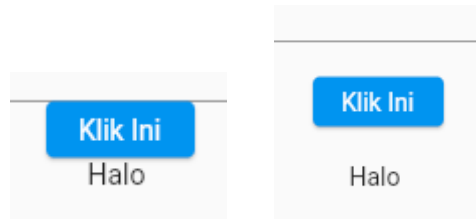


Padding pada Flutter juga merupakan Widget. Sebagai contoh, untuk button pada app sebelumnya, kita akan menambahkan padding 20 pixel ke semua arah (atas, bawah, kiri, kanan) dengan menggunakan `EdgeInsets` :

```
...
TextField(
  controller: textEditingController, //controller
),
Padding(
  padding: EdgeInsets.all(20), //20 pixel ke semua arah
  child:
    ElevatedButton(
      ...
    ),
),
...
```

Pada code di atas, button diletakkan sebagai child widget `Padding`.

Bandingkan hasilnya sebelum dan sesudah:



Selain padding ke semua arah, `EdgeInsets` dapat diatur ke arah kiri, kanan, atas, bawah saja. Berikut contoh penggunaan `EdgeInsets` untuk arah tertentu:

```
EdgeInsets.only(right:50);
EdgeInsets.only(left:10);
EdgeInsets.only(top:5);
EdgeInsets.only(bottom:10);
```

Kombinasi dapat digunakan sebagai berikut

```
EdgeInsets.only(top: 50, bottom: 50)
```

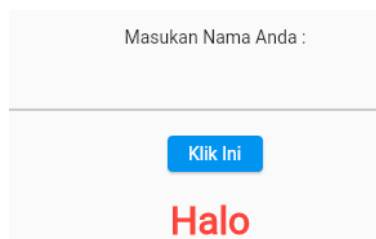
Untuk menangani ukuran layar yang beragam, Flutter menggunakan logical pixel, bukan physical pixel.

## Formatting Text

Warna, format ukuran dapat diatur melalui widget `TextStyle`. Kita akan ubah style pada font output sebagai berikut

```
Text(
  'Halo $_nama',
  style:
    const TextStyle(
      fontSize: 30,
      fontWeight: FontWeight.bold,
      color: Colors.red),
),
```

Hasilnya:



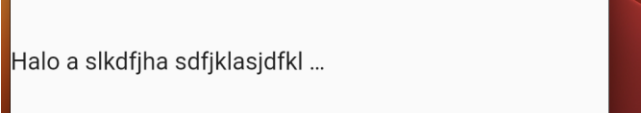
Silakan coba berbagai property TextStyle lainnya.

Untuk penanganan overflow text, tersedia atribut `overflow`

Sebagai contoh, jika teksnya panjang dan overflow di-set `ellipsis`

```
Text( 'Halo a slkdfjha sdfjklasjdfkl ajsdklfjaslk; dfjkl;asdjfkla sjdkl;fj  
askldfjalksd jfklasdjfklasdjlf kjasdlk;fj;aslj fal; skdjf$_nama',  
      overflow: TextOverflow.ellipsis,  
      style: TextStyle(fontSize: 20),  
    ),
```

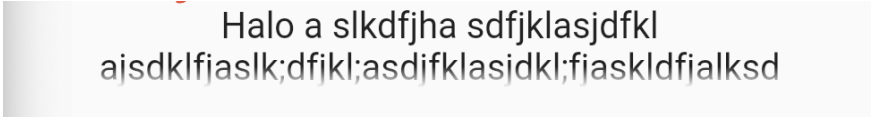
maka hasilnya teks akan dipotong dan ditambahkan tiga titik "...":



Terdapat atribut `maxLines` untuk mengatur jumlah maksimum baris.

Selain `TextOverflow.ellipsis`, tersedia `TextOverflow.clip`, `fade` dan `visible`. Sebagai contoh kombinasi `fade` dan `maxLines`. Tambahkan `textAlign` agar teks berada di tengah

```
overflow: TextOverflow.fade,  
maxLines: 2,  
textAlign: TextAlign.center
```



Code lengkap: <https://pastebin.com/raw/wqFaw4FQ> d

## Latihan 2:

Tambahkan formatting text dan padding pada latihan 1 sesuai dengan selera anda.

## Formatting Button

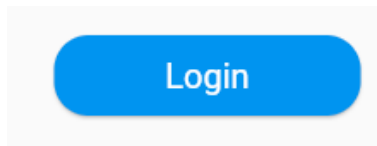
Format button dapat diatur melalui atribut `style`. Sebagai contoh untuk membuat button dengan ujung yang membulat.

```

ElevatedButton(
  onPressed: () {},
  child: const Text('  Login  '),
  style: ElevatedButton.styleFrom(
    textStyle: const TextStyle(fontSize: 20),
    padding:
      const EdgeInsets.symmetric(horizontal: 50, vertical: 20),
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(20.0),
    )),
)

```

Hasilnya adalah sebagai berikut:





# Built-In UI Widget

Setelah mencoba sekilas Flutter dengan app HelloWorld dan berapa widget, berikutnya kita akan mencoba lebih jauh berbagai widget terkait user interface.

Demo widget UI berbasis Material Design 3 :

[https://flutter.github.io/samples/web/material\\_3\\_demo/](https://flutter.github.io/samples/web/material_3_demo/)

Selain text label, button, text edit, padding seperti yang telah kita coba sebelumnya, Flutter menyediakan berbagai built-in widget untuk user interface. Berikut beberapa widget yang umum digunakan.

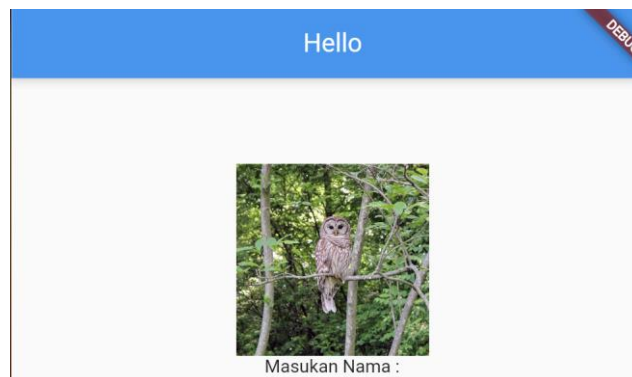
## Image Widget

Widget ini digunakan untuk menampilkan gambar dengan format: jpeg, bmp, gif, webp

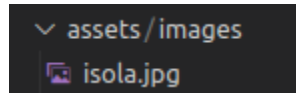
Image.network digunakan untuk menambahkan gambar dari internet. Sebagai contoh kita akan menambahkan image di code sebelumnya di atas teks "Masukan Nama":

```
children: [  
  Image.network(  
    'https://flutter.github.io/assets-for-api-docs/assets/widgets/owl-2.jpg',  
    height: 150,  
  ),  
  . . .
```

Hasilnya:



Gambar juga dapat diambil dari assets lokal. Pada direktori project flutter kita, letakkan sebuah file image di assets/images (buat directory-nya terlebih dulu). Letakkan file image di sana.



Untuk me-load image tersebut gunakan method `asset()`:


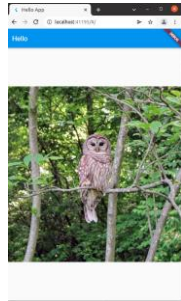


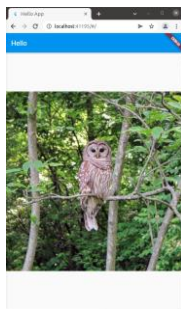
```
Image.asset(
  "images/isola.jpg", //pastikan file ada di [project]/assets/images
  height: 150,
),
```

Widget Image memiliki atribut `fit` untuk meng-align image di dalam batas. Ini berguna saat kita ingin gambar mengisi tempat yang disediakan. Atribut `fit` dapat berisi berbagai nilai. Tabel berikut memperlihatkan perbedaannya untuk code berikut (silakan dicoba):

```
body: Center(
  child: Image.network(
    "https://flutter.github.io/assets-for-api-docs/assets/widgets/owl-2.jpg",
    fit: BoxFit.cover,
    width: double.infinity,
    height: double.infinity,
  ),
),
```

Berikut hasil dari berbagai nilai atribut `fit`.

Nilai Atribut "fit"	Deskripsi	Hasil
<code>BoxFit.cover</code>	Sekecil mungkin tapi menutupi semua kotak target.	
<code>BoxFit.fill</code>	Mengisi kotak target dengan menyesuaikan aspect ratio (walaupun terdistorsi).	

<code>BoxFit.contain</code>	Sebesar mungkin tapi tetap berada di kotak target.	
<code>BoxFit.scaleDown</code>	Align image di dalam kotak target (center) dan jika diperlukan scale down untuk memastikan image cukup.	
<code>BoxFit.fitHeight</code>	Memastikan full height image ditampilkan walaupun image overflow secara horizontal.	
<code>BoxFit.fitWidth</code>	Memastikan full width image ditampilkan walaupun harus overflow secara vertikal	
<code>BoxFit.none</code>	Align image pada kotak target (center) dan buang bagian yang berada di luar kotak. Image tidak di-resize.	

# Dropdown

## Dropdown Button

Kita akan menambahkan dropdown pilihan selamat pagi, siang dan sore dalam code sebelumnya.

Pertama tambahkan isi dari dropdown (List berisi DropdownMenuItem):

```
Widget build(BuildContext context) {  
  //String pilihanSalam = "pagi"; //jangan letakkan var ini di sini! tapi di state  
  List<DropdownMenuItem<String>> salam = [];  
  
  var itm1 = const DropdownMenuItem<String>(  
    value: "pagi",  
    child: Text("selamat pagi"),  
  );  
  var itm2 = const DropdownMenuItem<String>(  
    value: "siang",  
    child: Text("selamat siang"),  
  );  
  
  salam.add(itm1);  
  salam.add(itm2);  
  . . .  
}
```

Tambahkan variabel di class state untuk menyimpan informasi dropdown yang dipilih user.

```
class MyAppState extends State<MyApp> {  
  . . .  
  String pilihanSalam = "pagi"; // harus diletakkan di sini  
  String pilihanSalamOut = ""; //untuk ditampilkan  
}
```

Kemudian tambahkan widget DropdownButton

```
children: [  
  DropdownButton(  
    value: pilihanSalam,  
    items: salam,  
    onChanged: (String? newValue) {  
      setState(() { //jangan lupa setState  
        if (newValue != null) {  
          pilihanSalam = newValue;  
        }  
      });  
    },  
  ),  
  . . .  
]
```

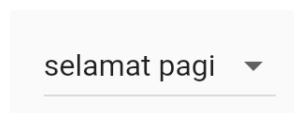
onChanged diperlukan karena setiap user mengganti pilihan, teks yang dipilih di dropdown tidak otomatis terupdate di dropdown (perhatikan `value:pilihanSalam` dan `pilhanSalam=newValue`). Jika onChanged diset null maka DropdownButton menjadi disabled.

Berikutnya kita update variabel yang isinya akan ditampilkan saat button di-tap.

```
child:
  ElevatedButton(
    onPressed: () {
      setState(() {
        _nama = textEditingController.text;
        pilihanSalamOut = pilihanSalam; //ambil dari state
      });
    },
  ),
```

Lalu pilihSalamOut ini akan di-update di UI. Ternary operator digunakan untuk mengecek apakah nama sudah terisi, sebelum ditampilkan.

```
Text(
  _nama != "" ? 'Halo $_nama selamat $pilihanSalamOut' : "",
  style: . . .
),
```



Code lengkap

<https://pastebin.com/raw/mASCSNTJ>

Isi dari dropdownlist dapat ditulis lebih singkat menggunakan `map()` jika value sama dengan labelnya:

```

DropdownButton(
    value: pilihanSalam,
    items: <String>['pagi', 'siang']
        .map<DropdownMenuItem<String>>((String value) {
            return DropdownMenuItem<String>(
                value: value,
                child: Text(value),
            );
        }).toList(),
    . . .

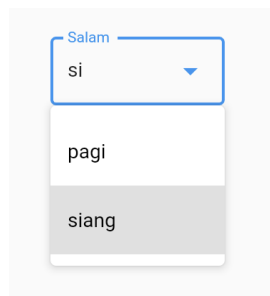
```

### Latihan 3:

Mengapa di dalam text, pilihanSalamOut digunakan bukan pilihanSalam langsung?

## Dropdown Menu

Dropdown menu mirip dengan dengan dropdown button, perbedaanya isi dropdown menu juga dapat diisi oleh user melalui keyboard (gambar bawah)



Dropdown menu menggunakan DropdownMenuEntry untuk elemennya (berbeda dengan dropdown button yang menggunakan DropDownMenuItem)

```

//isian dari drop down
List<DropdownMenuEntry<String>> items =
    <String>['pagi', 'siang'].map<DropdownMenuEntry<String>>((String value)
{
    return DropdownMenuEntry<String>(
        label: value,
        value: value,
    );
}).toList();

```

Dropdown menu juga menggunakan `textEditingController` untuk menyimpan input user melalui keyboard.

```
class _MyAppState extends State<MyApp> {  
  final TextEditingController salamController = TextEditingController();  
  String? pilihanSalam;  
  
  . . .  
  child: Column(mainAxisSize: MainAxisSize.min, children: [  
    DropdownMenu<String>(  
      initialSelection: "pagi",  
      controller: salamController,  
      label: const Text('Salam'),  
      dropdownMenuEntries: items,  
      onSelect: (String? salam) {  
        setState(() {  
          pilihanSalam = salam;  
        });  
      },  
    ),  
    Text("Pilihan : $pilihanSalam") //tampilkan hasil  
  ]),  
}
```

Code lengkap:

<https://pastebin.com/raw/wURMGYul>

## Button

Pada contoh sebelumnya kita telah menggunakan widget `elevatedButton`, Flutter menyediakan beberapa jenis button lainnya:

1. **FloatingActionButton**. Biasanya bentuknya lingkaran, menggunakan icon, berada di kanan bawah.
2. **TextButton**, text yang bisa di-tap.
3. **IconButton**, menggunakan icon.
4. **SegmentedButton**, button on/off (toggle), cocok untuk filter
5. **PopupMenuButton**, button untuk menu. Popup Menu Button akan dibahas lebih rinci di bagian appBar.
6. **CupertinoButton**, untuk iOS.

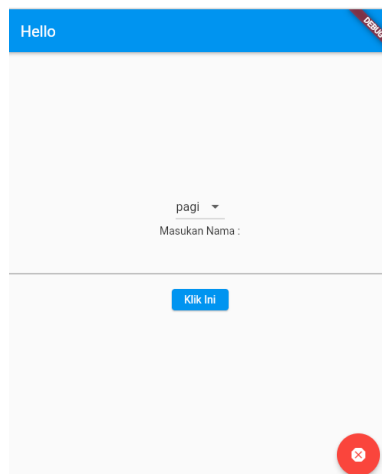
Berikut contoh singkat untuk setiap button

## FloatingActionButton

Floating button berada di atas UI (mengambang) yang digunakan untuk aksi utama. Bentuk dari Floating Action biasanya lingkaran.

```
home: Scaffold(  
  appBar: AppBar(  
    title: const Text('Hello'),  
  ),  
  //harus dibawah langsung Scaffold  
  floatingActionButton: FloatingActionButton(  
    onPressed: () {  
      // jika ditap  
    },  
    backgroundColor: Colors.red,  
    child: const Icon(Icons.dangerous),  
  ),  
  ... //dst
```

Hasilnya:



## TextButton

TextButton mirip seperti teks biasa tetapi dapat di-tap.

```
TextButton(  
  style: TextButton.styleFrom(  
    primary: Colors.green,  
    textStyle: const TextStyle(fontSize: 20),  
  ),  
  onPressed: () { // aksi jika ditap  
  },  
  child: const Text('Ini TextButton'),  
)
```



Hasilnya dapat dilihat di gambar bawah (tombol di bagian atas adalah ElevatedButton sebagai pembanding)

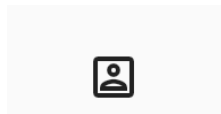


## IconButton

IconButton menampilkan button berbentuk icon.

```
IconButton(  
  icon: const Icon(Icons.account_box_outlined),  
  tooltip: 'Profil User',  
  onPressed: () {  
    // kalau ditap  
  },  
)
```

Hasilnya:



## Selection Widget

Selection widget menyediakan widget untuk memilih pilihan, terdiri atas checkbox, radio, switch, dan slider. Berikut contoh untuk setiap widget.

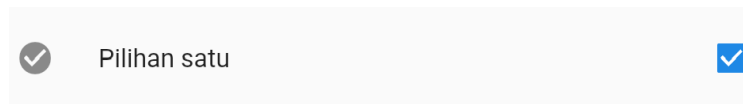
### Checkbox

CheckboxListTile adalah widget Checkbox yang memiliki label.

Checkbox yang hanya berisi kotak saja (tanpa label) dapat digunakan untuk UI yang lebih khusus dengan mengkombinasikannya dengan widget lain.

```
CheckboxListTile(  
  title: const Text('Pilihan satu'),  
  value: isChecked, //boolean  
  onChanged: (bool? value) {  
    setState(() {  
      isChecked = !isChecked;  
    });  
  },  
  secondary: const Icon(Icons.check_circle),  
)
```

Hasilnya:



### SegmentedButton

Segmented button dapat digunakan seperti checkbox, berbentuk toggle button. Berikut contohnya:

```
enum Sizes { extraSmall, small, medium, large, extraLarge }  
Set<Sizes> selection = <Sizes>{Sizes.large, Sizes.extraLarge};
```

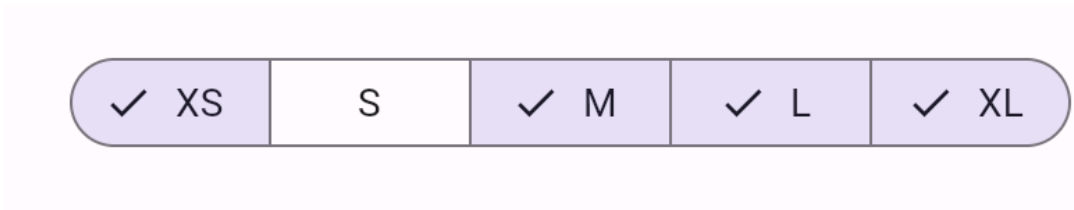
```
SegmentedButton<Sizes>(  
  segments: const <ButtonSegment<Sizes>>[  
    ButtonSegment<Sizes>(value: Sizes.extraSmall, label: Text('XS')),  
    ButtonSegment<Sizes>(value: Sizes.small, label: Text('S')),
```

```

        ButtonSegment<Sizes>(value: Sizes.medium, label: Text('M')),
        ButtonSegment<Sizes>(value: Sizes.large, label: Text('L')),
        ButtonSegment<Sizes>(value: Sizes.extraLarge, label: Text('XL')),
    ],
    selected: selection,
    onSelectionChanged: (Set<Sizes> newSelection) {
        setState(() {
            selection = newSelection;
        });
    },
    multiSelectionEnabled: true,
)

```

Hasilnya:



Code lengkap: <https://pastebin.com/raw/Pw1WspRt>

## FilterChips

Jika jumlah item untuk filter banyak dan dinamik, alternatif widget lain yang dapat digunakan adalah Filterchips

Berikut contohnya

```

enum JenisMakanan {
    sunda,yogya,solo,chinese,sumsel,padang,jepang,korea,
    western,betawi,tegal,menado,gorontalo,aceh,medan,seafood
}

class MyAppState extends State<MyApp> {
    //mana yang dipilih
    Set<JenisMakanan> filters = <JenisMakanan>{};
}

```

```

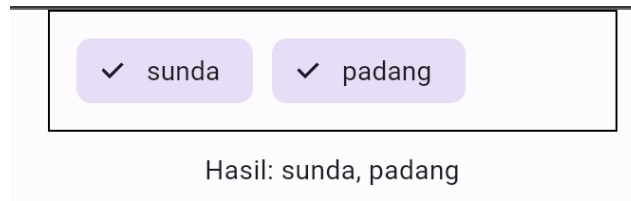
@override
Widget build(Object context) {
return MaterialApp(
  theme: ThemeData(useMaterial3: true),
  home: Scaffold(
    body: Column(children: [
      Container(
        decoration: BoxDecoration(border: Border.all()),
        padding: EdgeInsets.all(14),
        width: 300,
        child: Wrap(
          spacing: 10,
          children: [
            FilterChip(
              label: Text(JenisMakanan.sunda.name),
              selected: filters.contains(JenisMakanan.sunda),
              onSelect: (bool val) {
                setState(() {
                  if (val) {
                    filters.add(JenisMakanan.sunda);
                  } else {
                    filters.remove(JenisMakanan.sunda);
                  }
                });
              },
            ),
            FilterChip(
              label: Text(JenisMakanan.padang.name),
              selected: filters.contains(JenisMakanan.padang),
              onSelect: (bool selected) {
                setState(() {
                  if (selected) {
                    filters.add(JenisMakanan.padang);
                  } else {
                    filters.remove(JenisMakanan.padang);
                  }
                });
              },
            ),
          ],
        )),
      Container(height: 10.0),
      Text(
        'Hasil: ${filters.map((JenisMakananFilter e) => e.name).join(', ')}',
      ),
    ])),

```

```
);
}
}
```

Code lengkap: <https://pastebin.com/raw/tMSfmigu>

Hasilnya:



Tentu tidak praktis kalau setiap FilterChips widget dibuat satu persatu. Code sebelumnya dapat diubah dengan menghasilkan list filterchips berdasarkan enum jenis makanan.

```
JenisMakanan.values.map //petakan enum
(
  (JenisMakanan j) { return FilterChip( label:j.name; . . . ) }
).toList()
```

atau dengan notasi arrow

```
JenisMakanan.values.map
(
  (JenisMakanan j) => FilterChip( label:j.name; . . . )
).toList()
```

Code perubahannya:

```
child: Wrap(
  spacing: 10,
  children: JenisMakanan.values
    .map((JenisMakanan j) => FilterChip(
      label: Text(j.name),
      selected: filters.contains(j),
      onSelect: (bool val) {
        setState(() {
          if (val) {
            filters.add(j);
          } else {
            filters.remove(j);
          }
        });
      },
    )),
),
```

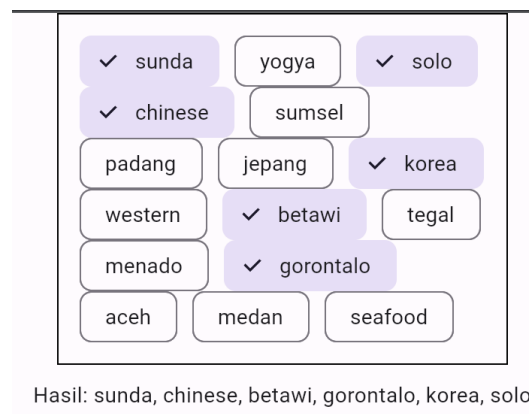
```

    ))
    .toList(),
)

```

Source code lengkap: <https://pastebin.com/raw/Pyr5Lsaj>

Hasilnya adalah sebagai berikut:



## Radio

Berbeda dengan checkbox, pada radio, hanya satu pilihan user pada suatu waktu. Berikut adalah contohnya:

Deklarasikan enum di luar class

```
enum mhsType { mhsS1, mhsS2 }
```

Tambahkan variabel pada class State

```
mhsType? jenisMhs; //awalnya bisa kosong, jadi perlu "?"
```

Tambahkan widget RadioListTile. Widget ini mengkombinasikan Radio dengan ListTile (list dengan tinggi sama)

```

RadioListTile<mhsType>(
  title: const Text('Mahasiswa S1'),
  value: mhsType.mhsS1,
  groupValue: jenisMhs,
  onChanged: (mhsType? value) {
    setState(() {
      jenisMhs = value;
    });
  },
),

```

```
RadioListTile<mhsType>(
  title: const Text('Mahasiswa S2'),
  value: mhsType.mhsS2,
  groupValue: jenisMhs,
  onChanged: (mhsType? value) {
    setState(() {
      jenisMhs = value; //jawaban user disimpan
    });
  },
),
```

Hasilnya adalah sebagai berikut:

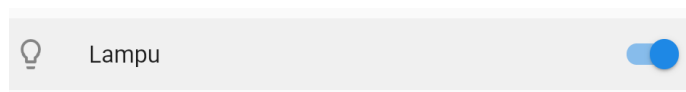


## Switch

widget switch digunakan untuk informasi yang hanya memiliki dua nilai (boolean)

Contoh code-nya:

```
SwitchListTile(
  title: const Text('Lampu'),
  value: isLampu, //boolean
  onChanged: (bool value) {
    setState(() {
      isLampu = value;
    });
  },
  secondary: const Icon(Icons.lightbulb_outline),
),
```



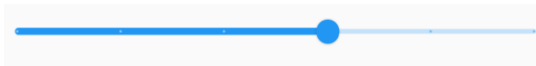
## Slider

Sesuai namanya, slider untuk memilih nilai dari suatu range yang tidak memerlukan input yang eksak (misal volume, brightness dsb).

Contoh code-nya:

```
Slider(  
    value: nilaiSlider, //tipe double  
    max: 100,  
    divisions: 5,  
    label: nilaiSlider.round().toString(),  
    onChanged: (double value) {  
        setState(() {  
            nilaiSlider = value;  
        });  
    },  
)
```

Hasilnya (ada hint saat lingkaran di-slide oleh user):



#### Latihan 4A:

Buatlah app yang menerima input dengan berbagai widget UI untuk data mahasiswa

nama: text  
gender: radio button.  
sudah bekerja: switch  
tinggi badan: slider.  
makanan favorit: checkbox  
pekerjaan ortu: dropdownmenu  
provinsi asal: dropdownbutton

## Card dan ListTile

Card menampilkan kotak yang agak membulat dan memiliki bayangan. Card cocok dikombinasikan dengan ListTile.

ListTile dapat berisi image (depan dan belakang), judul, dan subjudul. ListTile cocok digunakan untuk menampilkan informasi tentang suatu item dalam ListView.

Sebagai contoh, code berikut menghasilkan:

```
return Card(  
    child: ListTile(  
        onTap: () {},  
        leading: Image.network(  

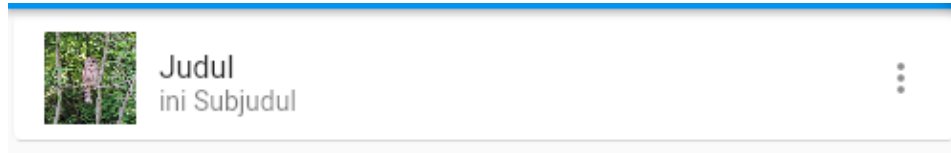
```



```

        'https://flutter.github.io/assets-for-api-docs/assets/widgets/owl-2.jpg'),
        trailing: const Icon(Icons.more_vert),
        title: const Text('Judul'),
        subtitle: const Text("ini Subjudul"),
        tileColor: Colors.white70));
    }

```



Code lengkapnya: <https://pastebin.com/raw/nwfA2edC>

## Form Validator

Flutter menyediakan form validation untuk memeriksa apakah input dari user sudah benar.

Kita akan memodifikasi contoh Hello X pada contoh sebelumnya

<https://pastebin.com/raw/wqFaw4FQ>

Tambahkan key sebagai identitas validator

```

class MyAppState extends State<MyApp> {
    final _formKey = GlobalKey<FormState>();
    . . .

```

dibagian MaterialApp home, tambahkan Form widget, tambahkan key, pindahkan scaffold menjadi child Form ini.

```

home: Form(
    key: _formKey,
    child: Scaffold(
        appBar: AppBar(
            title: const Text('Hello'),
        ),

```

Ganti TextField menjadi TextFormField, tambahkan validator bahwa nilai tidak boleh kosong.

```

TextFormField(
    validator: (value) {
        if (value == null || value.isEmpty) {
            return 'Tidak boleh kosong';
        }
        return null;
    }

```

```

    },
    controller: textEditingController, //controller
),

```

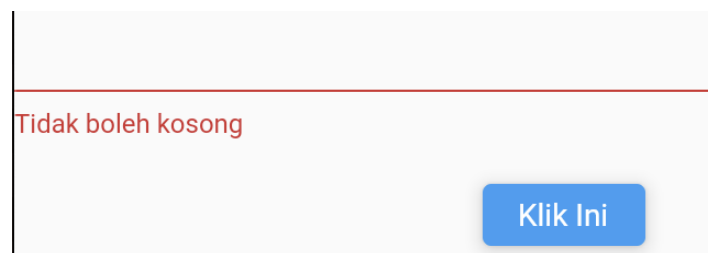
Saat button di tap, panggil validate()

```

ElevatedButton(
  onPressed: () {
    if (!_formKey.currentState!.validate()) {
      setState(() {
        _nama = textEditingController.text;
      }); //refresh
    }
  },
  child: const Text('Klik Ini'),
),

```

Hasilnya jika user tidak memasukkan data:



Selain TextFormField, tersedia DropdownButtonFormField.

#### Latihan 4B

Tambahkan validasi untuk latihan 4A, provinsi dan nama tidak boleh kosong

## SnackBar

SnackBar adalah pesan yang muncul di bagian bawah, dan menghilang otomatis setelah durasi tertentu. SnackBar cocok untuk memberikan feedback singkat pada user bahwa task sudah berhasil.

Sebagai contoh:



Snackbar memerlukan context bertipe `MaterialApp`, jadi salah satu solusinya kita buat home di class terpisah:

Codenya:

```
void main() {  
  runApp(const MyApp());  
}  
  
class MyApp extends StatelessWidget {  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      title: 'Hello App',  
      home: Scaffold(  
        appBar: AppBar(  
          title: const Text('Hello'),  
        ),  
        body: const BodyApp() , //contoh pemisahan nilai atribut di class terpisah  
      );  
    }  
  }  
}
```

```

class BodyApp extends StatelessWidget {
  const BodyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Center(
      child: ElevatedButton(
        onPressed: () {
          const snackBar = SnackBar(
            duration: Duration(seconds: 20),
            content: Text('Halo ini snack bar'),
          );
          ScaffoldMessenger.of(context).showSnackBar(snackBar);
        },
        child: const Text("Tap ini"),
      ),
    );
  }
}

```

## Dialog

### AlertDialog

AlertDialog dapat digunakan untuk menampilkan pesan yang membutuhkan aksi dari user.

Pada contoh berikut, di dalam class `MyAppState` ditambahkan method `tampilKandialog()` Di dalamnya panggil `showDialog()` dengan builder `AlertDialog`

```

void tampilKandialog(BuildContext context) {
  showDialog<String>(
    context: context,
    builder: (BuildContext context) => AlertDialog(
      title: const Text('Konfirmasi'),
      content: const Text('Anda yakin membatalkan?'),
      actions: <Widget>[
        TextButton(
          onPressed: () => Navigator.pop(context, 'Cancel'),
          child: const Text('Batal'),
        ),
        TextButton(
          onPressed: () => Navigator.pop(context, 'OK'),
          child: const Text('OK'),
        ),
      ],
    ),
  );
}

```

```

    ],
  ),
);
}

```

Panggil `tampilkanDialog()` ini saat button di-tap:

```

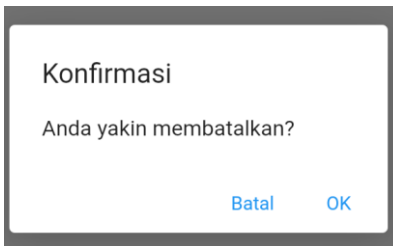
ElevatedButton(
  onPressed: () => tampilkanDialog(context),
  child: const Text('Klik Ini'),
),

```

Bagian `main()` perlu diubah karena diperlukan context `MaterialApp` di level yang lebih tinggi.

```
void main() => runApp(MaterialApp(home: MyApp()));
```

Jika dijalankan hasilnya:



Code lengkap: <https://pastebin.com/PWsU9pP6>

### Latihan 5

Tambahkan hasil dari latihan 4B, saat user men-tap button untuk submit, tambahkan konfirmasi apakah data sudah benar. Jika user menjawab "OK", tambahkan snackbar "Data berhasil disimpan"

# Referensi

Alessandria, Simone. *Flutter Projects: A Practical, Project-based Guide to Building Real-world Cross-platform Mobile Applications and Games*. Packt Publishing, 2020.

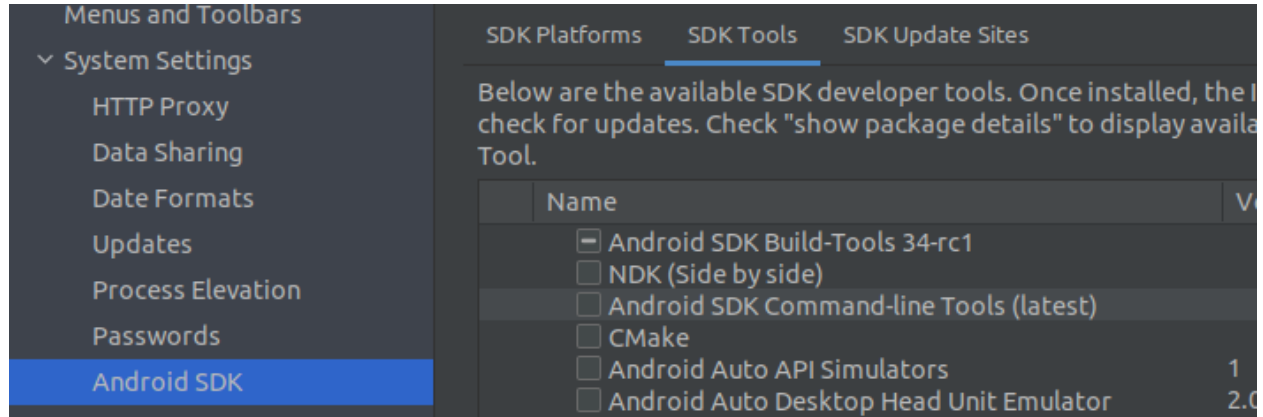
Bailey, Thomas. *Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter 2.5 and Dart*, 2nd Edition

<https://api.flutter.dev>

Tambahan

Flutter dan Android Studio

Setting → SDK Manager, command line tools



## Todo lain-lain:

SafeArea

TextSpan

Positioned

Flexible

isExpanded (DropDown)

SingleChildScrollView

Carousel

with (keyword seperti interface)

passing data antar widget

Bottomsheet

Expansion Panel

CircleAvatar, Divider

[https://api.flutter.dev/flutter/material/ListTile-](https://api.flutter.dev/flutter/material/ListTile-class.html?gclid=CjwKCAjwscGjBhAXEiwAswQqNEK78WoWKLTYXAQVpG1vi7a7CFOFFCuHP33Kk8QCFpv4w0SI1OrMABoCzUQQAvD_BwE&gclsrc=aw.ds)

[class.html?gclid=CjwKCAjwscGjBhAXEiwAswQqNEK78WoWKLTYXAQVpG1vi7a7CFOFFCuHP33Kk8QCFpv4w0SI1OrMABoCzUQQAvD\\_BwE&gclsrc=aw.ds](https://api.flutter.dev/flutter/material/ListTile-class.html?gclid=CjwKCAjwscGjBhAXEiwAswQqNEK78WoWKLTYXAQVpG1vi7a7CFOFFCuHP33Kk8QCFpv4w0SI1OrMABoCzUQQAvD_BwE&gclsrc=aw.ds)

Icon:

<https://fonts.google.com/icons>

## App hello world paling sederhana

```
import 'package:flutter/material.dart';
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override

  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```



