



# Flutter

การพัฒนาแอปพลิเคชันบนระบบปฏิบัติการ

Android / iOS ด้วย Flutter



# ร.ท. สมโภชน์ กุลธารารมณ์ อาจารย์แผนกวิศวกรรมศาสตร์ กองวิชาการศูนย์ใช้เบอร์ทหาร กองบัญชาการกองทัพไทย

## ประวัติการศึกษา

### ปริญญาโท

- วท.ม. สาขาวิชาเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

### ปริญญาตรี

- วท.บ. สาขาวิชาเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร

## ประวัติการทำงาน

พ.ศ. 2549- 2551

- ตำแหน่ง Developer บริษัท AISOF (บริษัทรับพัฒนาแอปพลิเคชันด้านการท่องเที่ยว)

พ.ศ. 2551- 2561

- หัวหน้างานพัฒนาซอฟต์แวร์ สำนักวิทยบริการและเทคโนโลยีสารสนเทศมหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร

พ.ศ. 2561 – ปัจจุบัน

- อาจารย์แผนกวิศวกรรมศาสตร์ กองวิชาการศูนย์ใช้เบอร์ทหาร กองบัญชาการกองทัพไทย

## Certificate

CompTIA: Classroom Trainer (CTT+), Project+, A+, Network+, Security+

ORACLE: Oracle Database 11g Administrator Certified Associate

ADOBE: Visual Communication using Adobe Photoshop® CS6

NSTDA (สวทช.): Information Technology Professionals Examination Council: ITPEC

Microsoft: Microsoft Certified Trainer, Microsoft Certified Solution Expert

## Flutter Intro

- Flutter คือ Framework ที่ใช้สร้าง UI สำหรับ Mobile Application ที่สามารถทำงานได้ทั้ง iOS และ Android โดยภาษาที่ใช้ใน Flutter นี้จะเป็นภาษา dart ซึ่งถูกพัฒนาโดย Google และที่สำคัญคือเป็น open source ที่สามารถใช้งานได้แบบฟรี ๆ อีกด้วย



Flutter

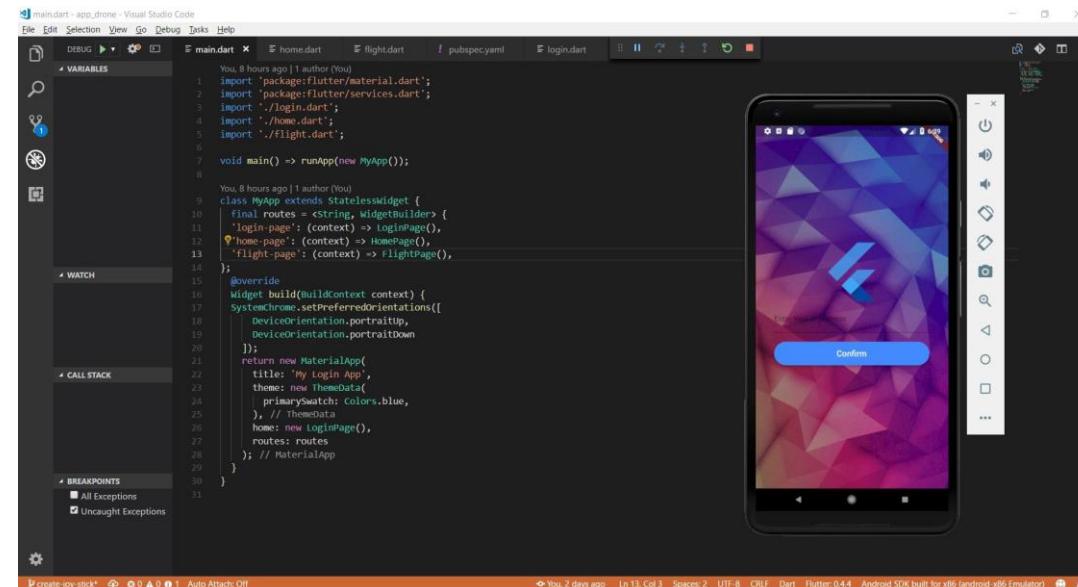
## Flutter Intro (ต่อ)

- ตัวอย่าง syntax ของภาษา dart ที่ใช้ใน Flutter ซึ่งจะมีความคล้ายกับภาษา Java เนื่องจาก dart เป็นภาษาที่รองรับ OOP และมีแนวคิดของ class และ inheritance เช่นเดียวกับภาษา Java นั้นเอง

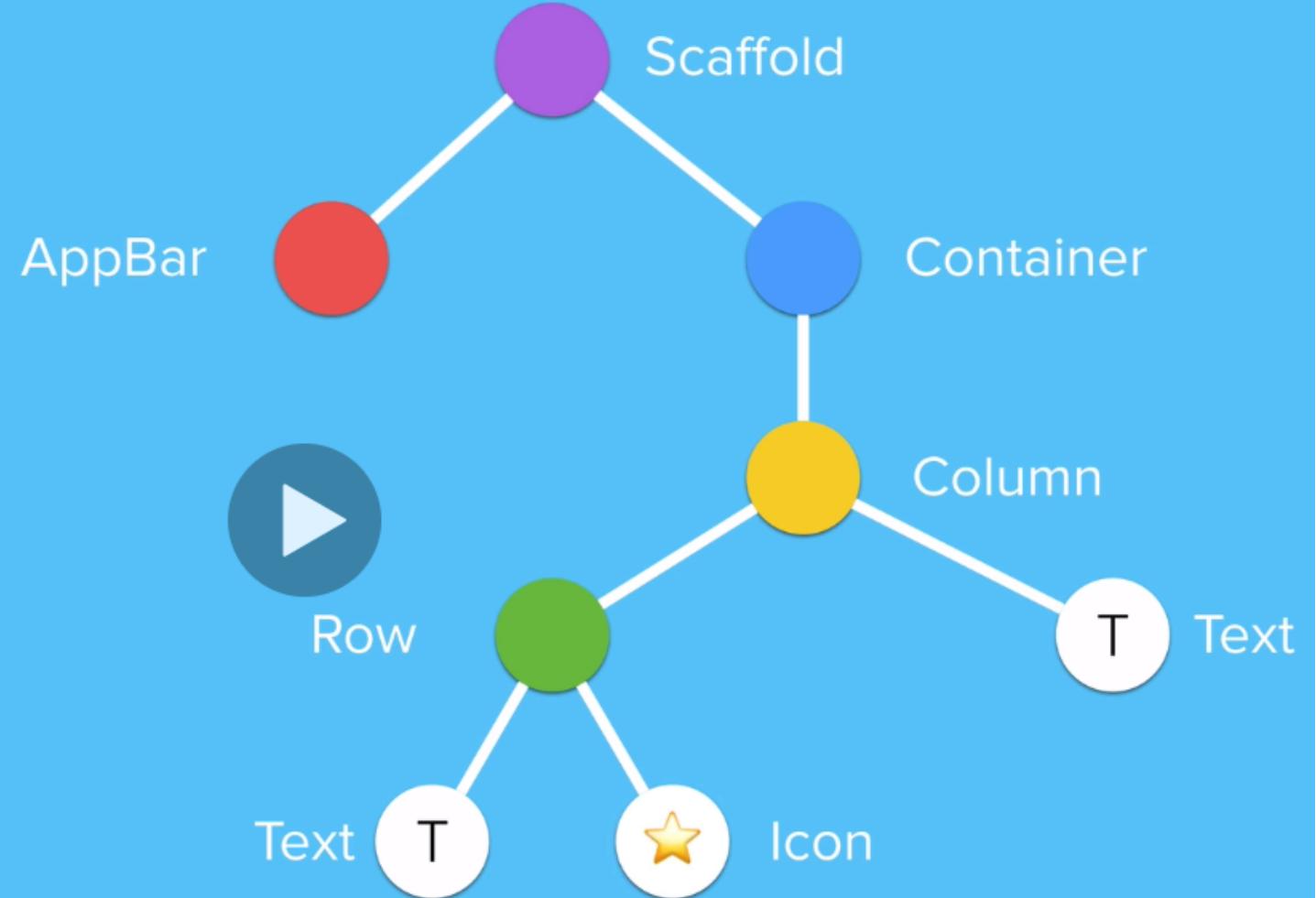
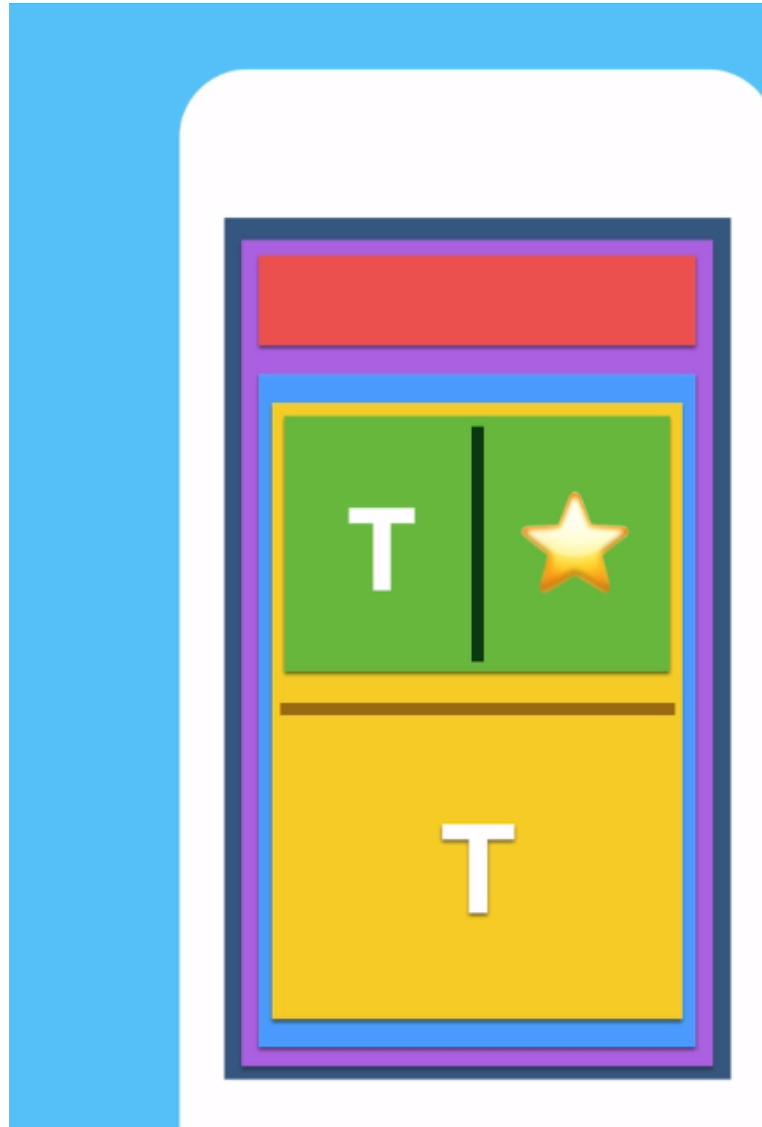
```
1 import 'package:flutter/material.dart';
2
3 void main() {
4     runApp(new MaterialApp(
5         home: new MyApp(),
6     ));
7 }
8
9 class MyApp extends StatelessWidget {
10    @override
11    Widget build(BuildContext context) {
12        return new Scaffold(
13            appBar: new AppBar(
14                title: new Text("Example App"),
15                backgroundColor: Colors.blue,
16            ),
17            backgroundColor: Colors.blue,
18            body: new Center(
19                child: new Column(
20                    mainAxisAlignment: MainAxisAlignment.center,
21                    children: <Widget>[
22                        new Icon(Icons.favorite, color: Colors.redAccent, size: 200.0,
23                        ),
24                    ],
25                ),
26            ),
27        );
28    }
29 }
```

# จุดเด่นของ Flutter คืออะไร?

- จุดเด่นหลัก ๆ ของ Flutter คือ ระบบ Hot Reload โดยเมื่อมีการทดสอบ, การสร้าง, การ add features หรือการกระทำต่าง ๆ กับ UI จะต้องมีการ reload เพื่อให้หน้า UI update ซึ่งระบบ Hot Reload จะเข้ามาช่วยในส่วนของการ reload โดยจุดเด่นของระบบนี้คือการย่นระยะเวลาที่ใช้ในการ reload ให้เหลือเพียงเสี้ยววินาทีเท่านั้น ทำให้การพัฒนา UI ของ application มีความรวดเร็วขึ้นอย่างมาก และยังมีจุดเด่นอื่น ๆ ที่ช่วยให้การพัฒนาเป็นไปได้ง่าย ซึ่งไม่ว่าจะเป็น Build-In ที่ช่วยในการออกแบบ UI ให้มีความสวยงามยิ่งขึ้นอย่าง Material Design และ Cupertino (iOS-flavor), มี Framework ที่ช่วยให้การทำ animation ต่าง ๆ หรือ gesture ของ UI เป็นเรื่องง่ายยิ่งขึ้น และยังสามารถใช้งานร่วมกับ IDE ที่กำลังเป็นที่นิยมอยู่ในปัจจุบันอย่าง VS Code และ Android Studio ได้อีกด้วย

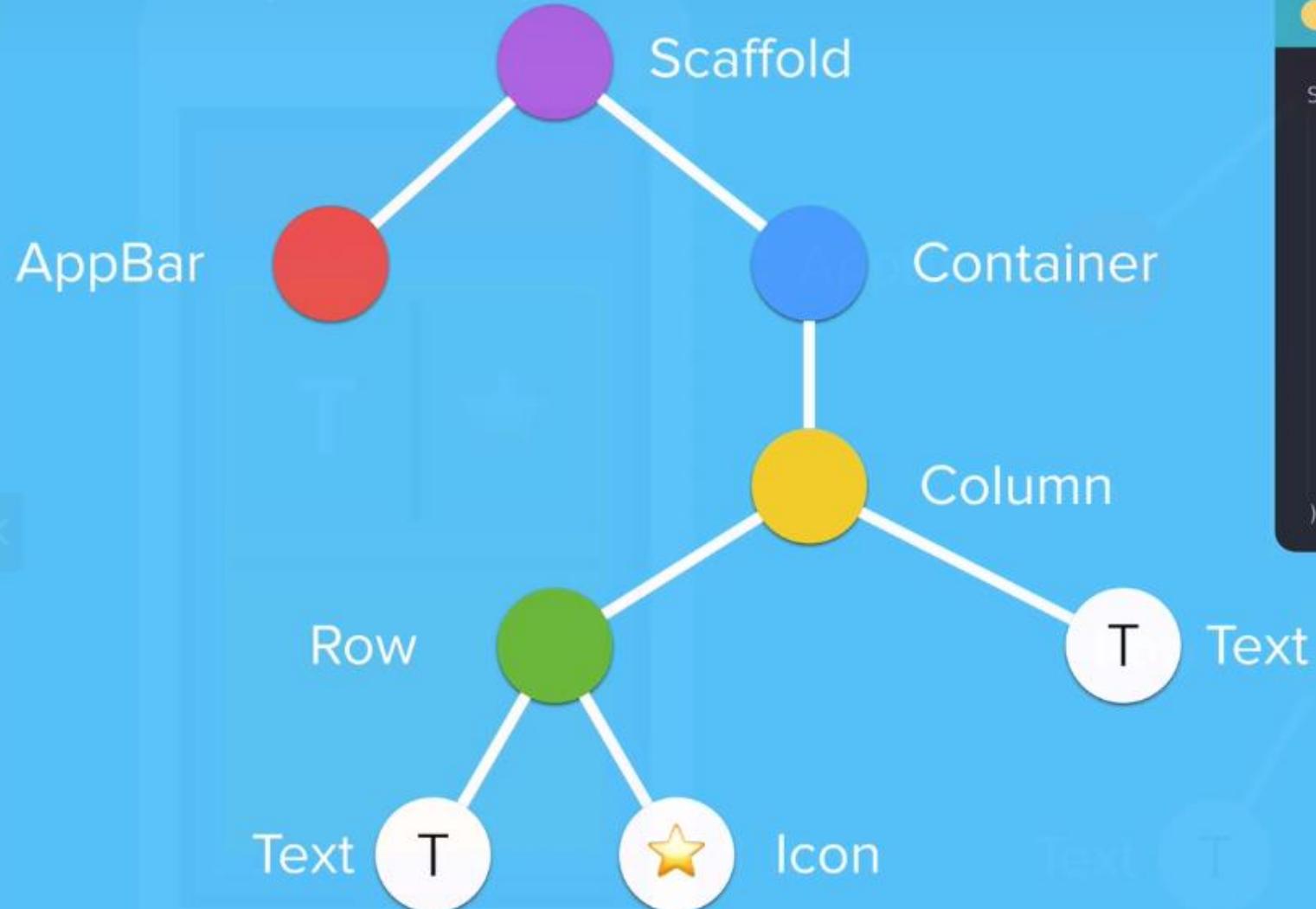


# Anatomy of Flutter



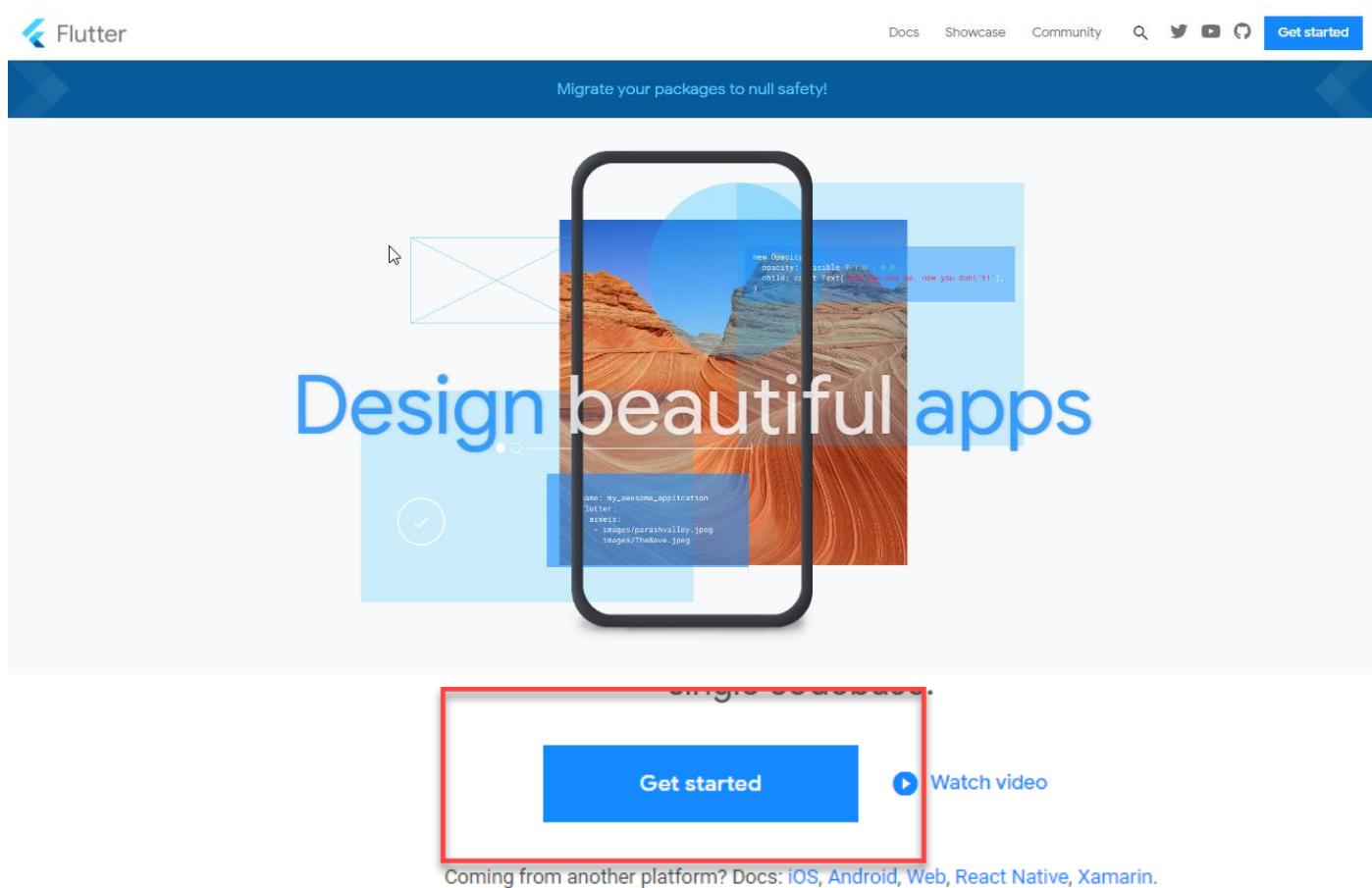
# Anatomy of Flutter

## 5. The Anatomy of a Flutter App



# 1. Install Flutter

1.1 เข้า Link Download <https://flutter.dev/>



แล้วกด Get started

Migrate your packages to null safety!

## Install

Docs > Get started > Install

Select the operating system on which you are installing Flutter:

Windows   macOS   Linux   Chrome OS

**Important:** If you're in China, see [Get Flutter in China.](#)

เลือก OS ที่ใช้พัฒนา

Set up an editor

flutter-dev@ ~ terms · brand usage · security · privacy · español · 社区中文资源 · 한국어 · We stand in solidarity with the Black community. Black Lives Matter

Flutter

Get started

1. Install
2. Set up an editor
3. Test drive
4. Write your first app
5. Learn more

From another platform?

- Flutter for Android devs
- Flutter for iOS devs
- Flutter for React Native devs
- Flutter for web devs
- Flutter for Xamarin.Forms devs
- Introduction to declarative UI
- Dart language overview
- Building a web app

Samples & tutorials

Development

Contents

- 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](#)
- [Web setup](#)
- [Next step](#)

## Get the Flutter SDK

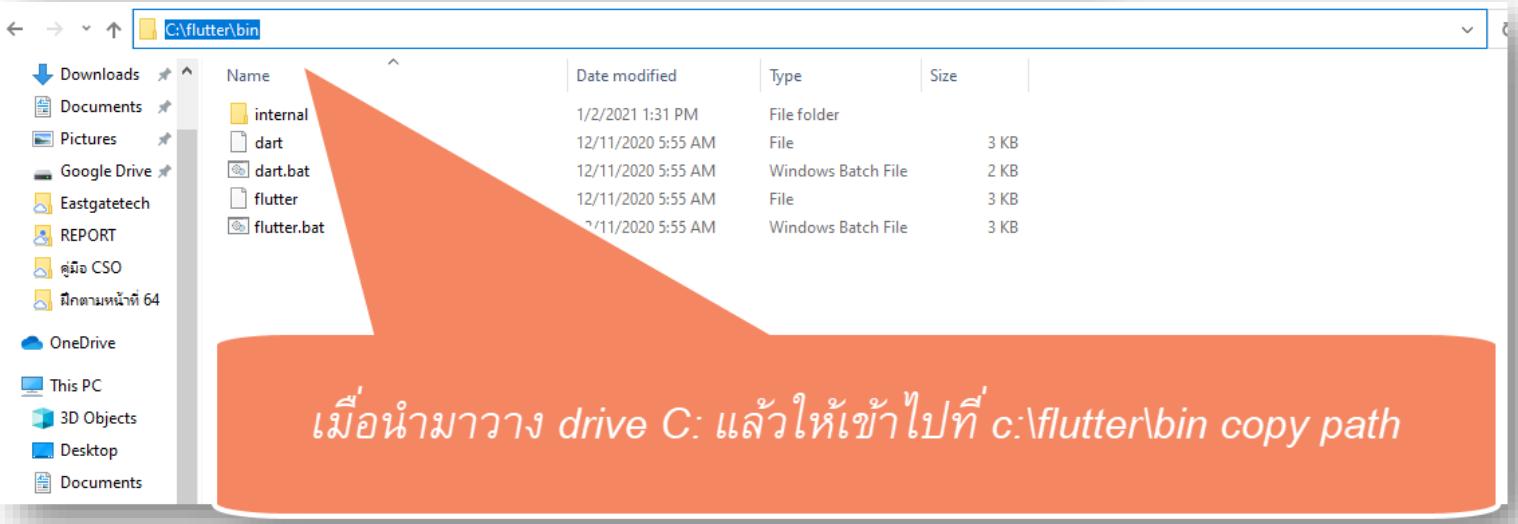
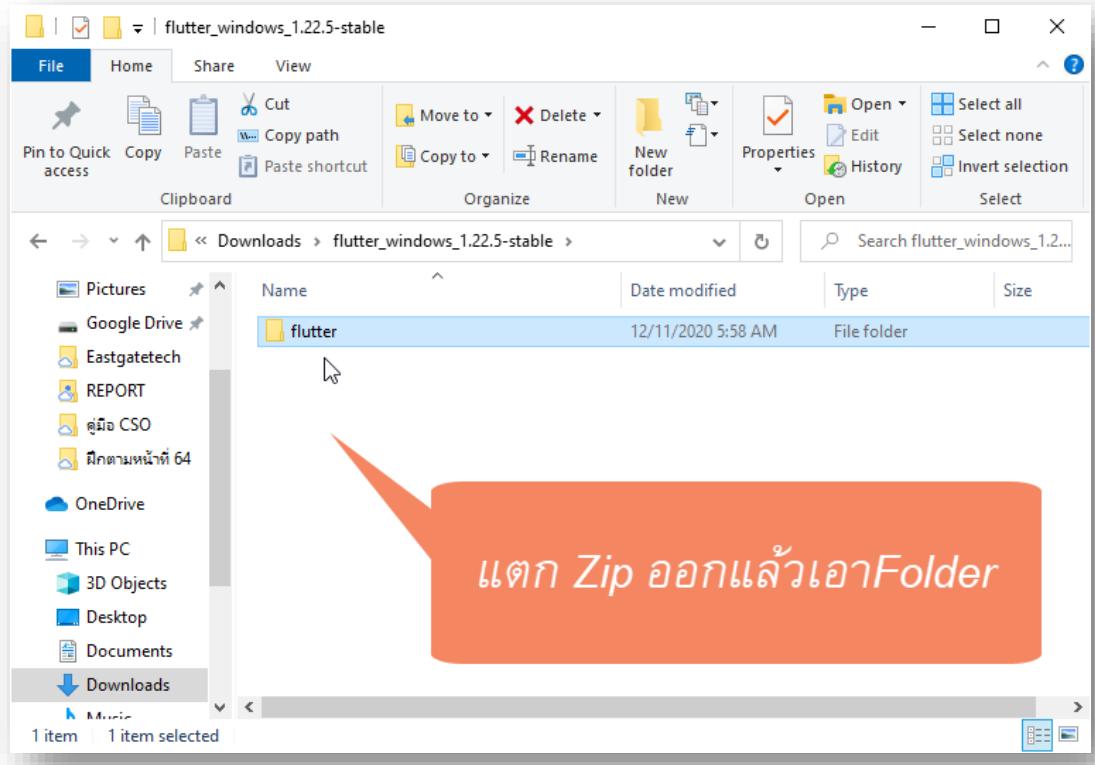
- Download the following installation bundle to get the latest stable release of the Flutter SDK:  
[flutter\\_windows\\_1.22.5-stable.zip](#)
- Extract the zip file and place the contained `flutter` directory in `C:\src\flutter`.

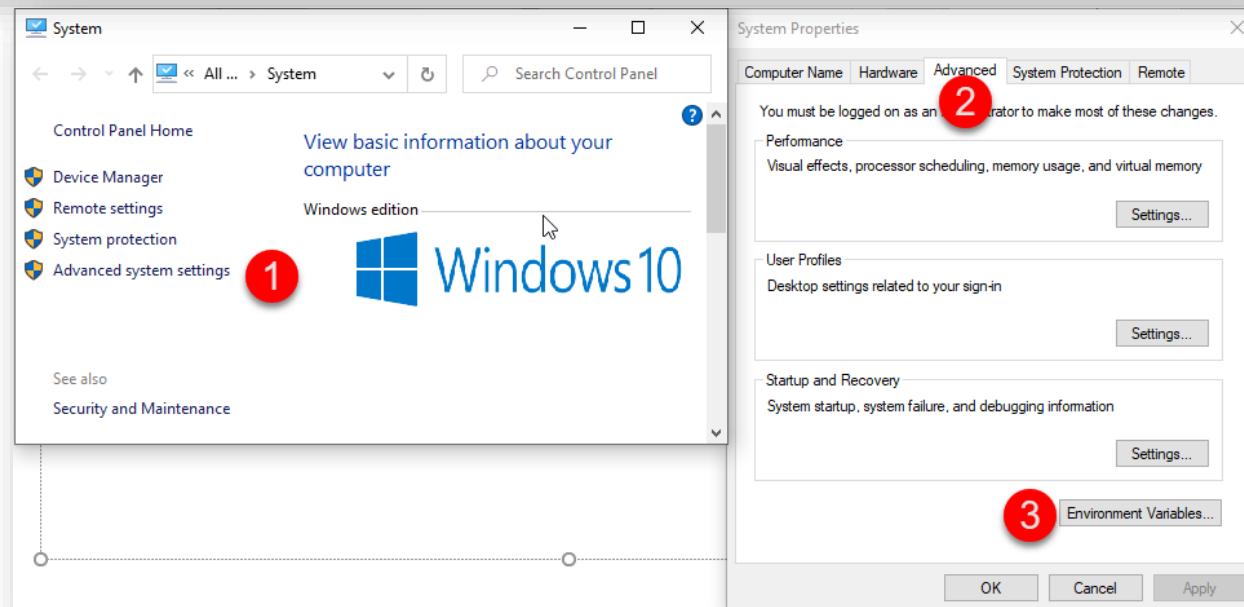
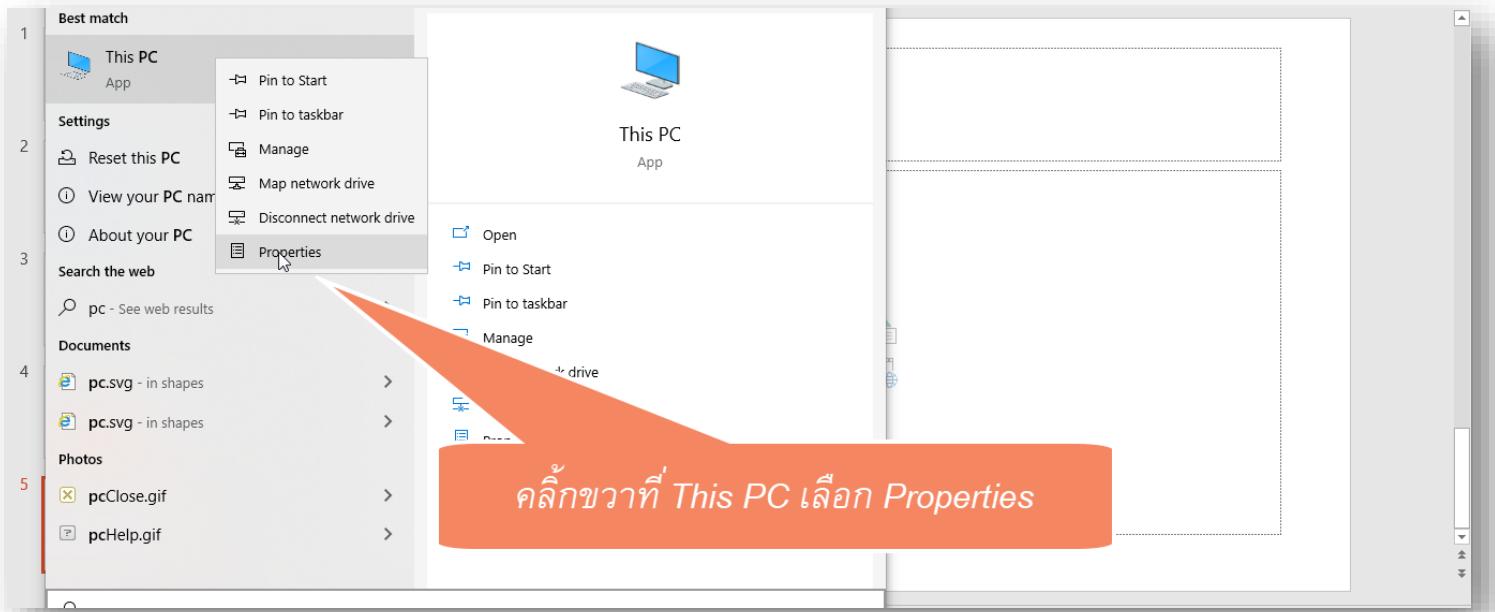
**Warning:** Do not install Flutter in a directory like `C:\Program Files\` that requires elevated privileges.

If you don't want to install a fixed version of the installation bundle, you can skip steps 1 and 2. Instead, get the source code from the [Flutter repo](#) on GitHub, and change branches or tags as needed. For example:

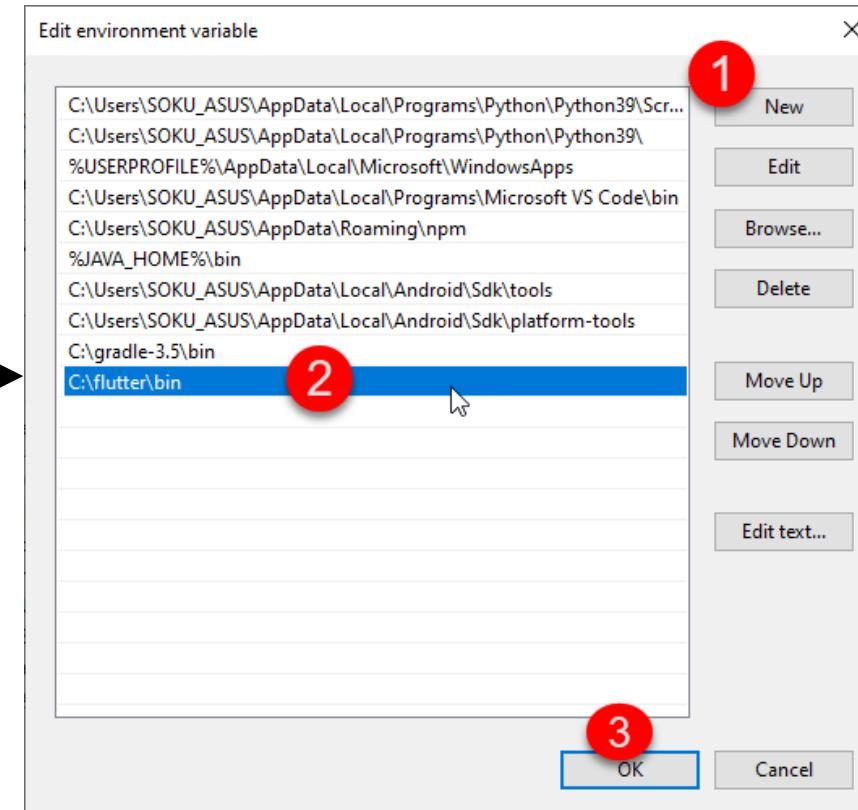
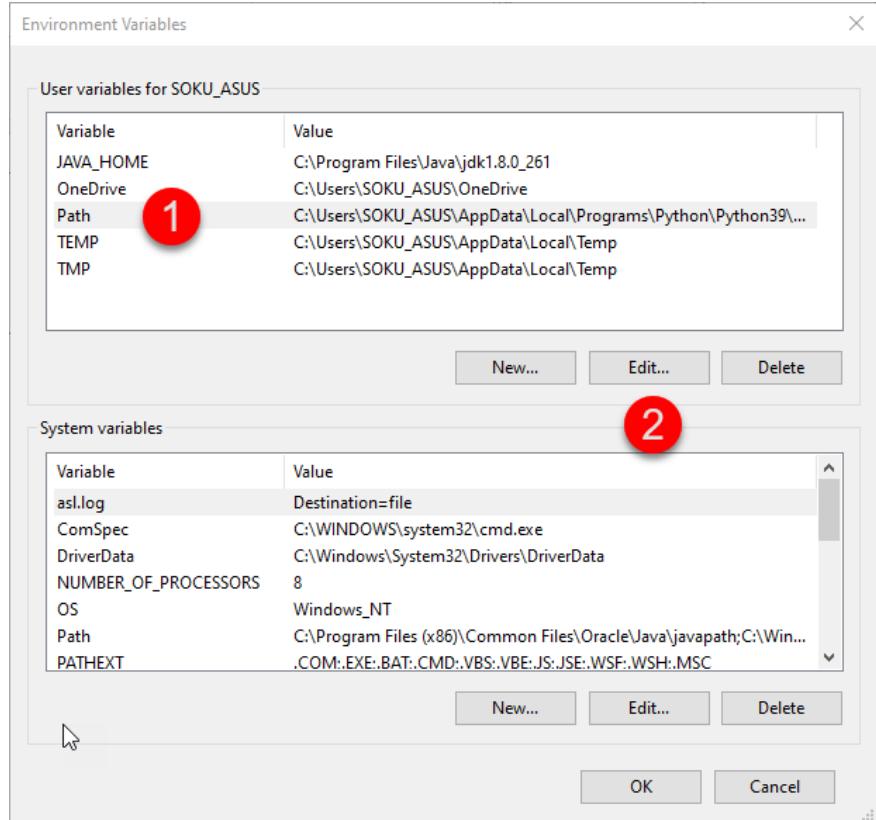
```
C:\src>git clone https://github.com/flutter/flutter.git -b stable
```

กด Download นี่!



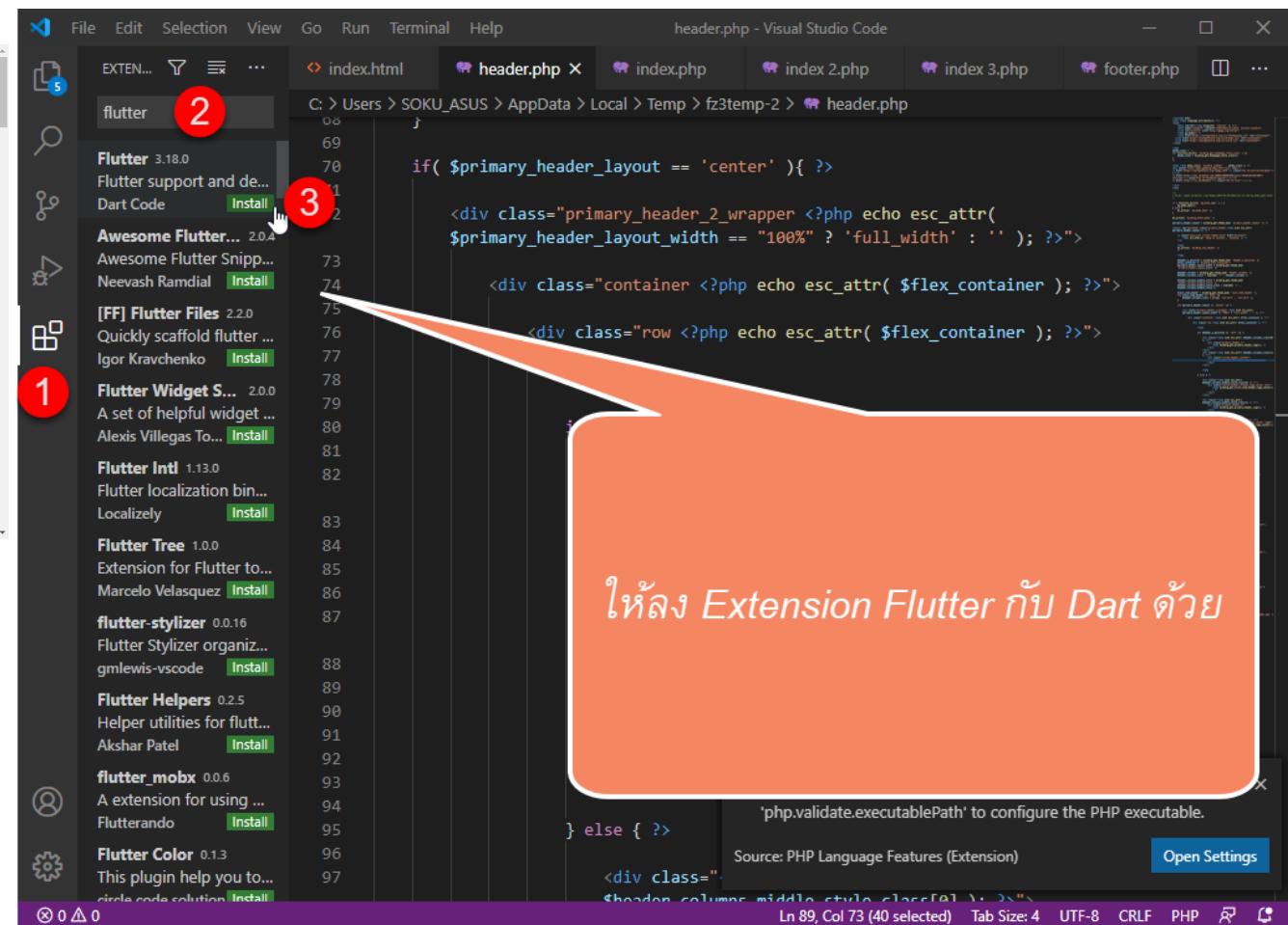
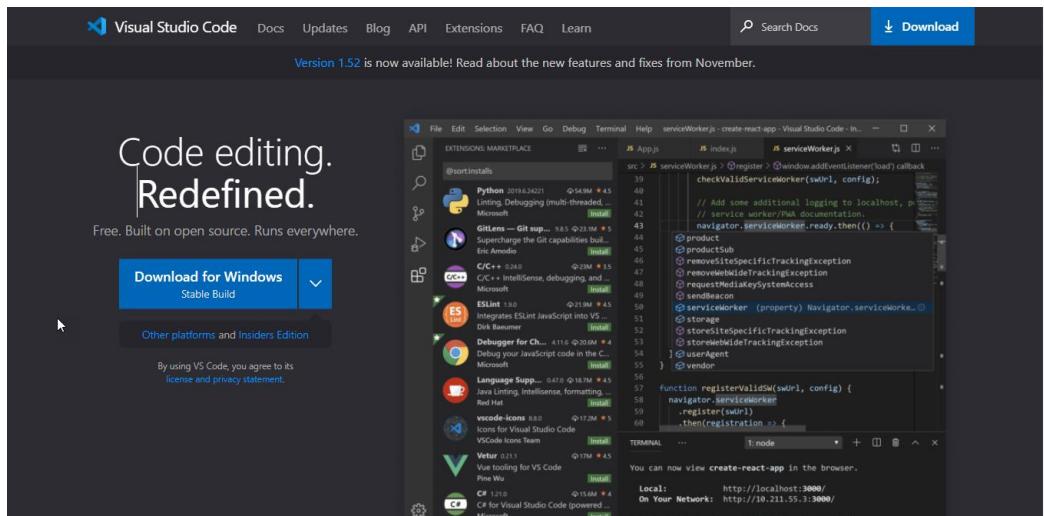


Advance System setting->Advance->Environment Variable



## 2. Install Visual Studio Code

- <https://code.visualstudio.com/> โหลดเสร็จให้ Install กด next จนเสร็จ



### 3. Run Flutter Doctor

- เปิด cmd มา พิมพ์ flutter doctor

```
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\SOKU_ASUS>flutter doctor
```

```
flutter doctor จะเช็คความพร้อมก่อนสร้างแอป

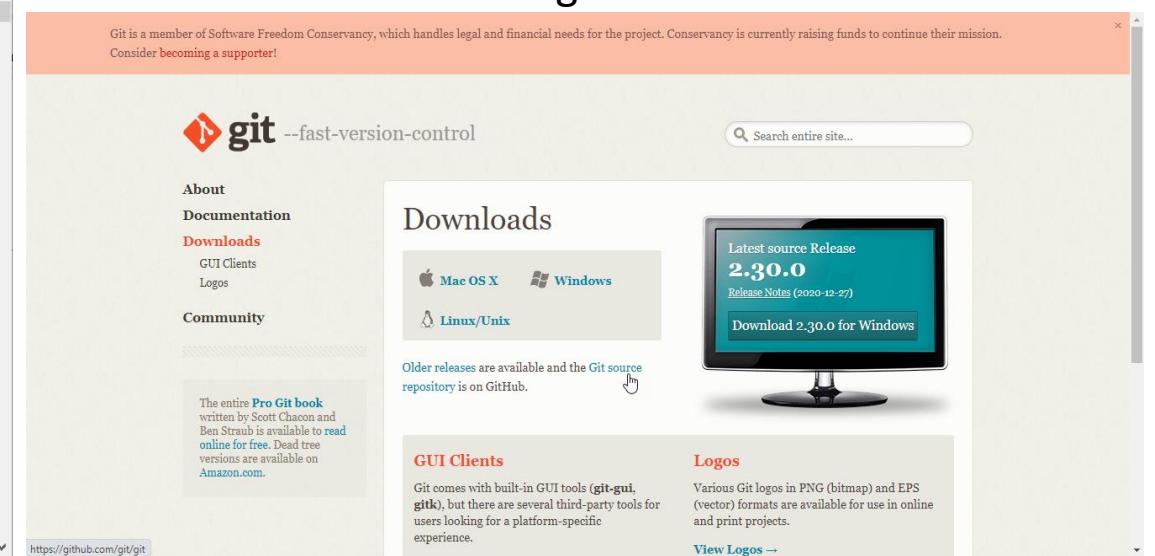
Downloading Material fonts...
Downloading Gradle Wrapper...
Downloading package sky engine...
Downloading flutter patched sdk tools...
Downloading flutter patched sdk_product tools...
Downloading windows-x64 tools...
Downloading windows-x64/font-subset tools...
Downloading android-arm-profile/windows-x64...
0.1s
0.3s
0.4s
0.6s
1.1s
Doctor summary (to see all details, run flutter doctor --v):
[!] Flutter (Channel stable, 1.22.5, on Microsoft Windows [Version 10.0.18363.1256], locale en-US)

[!] Android toolchain - develop for Android devices (Android SDK version 30.0.2)
    ! Some Android licenses not accepted. To resolve this, run: flutter doctor --android-licenses
[!] Android Studio (version 4.0)
    X Flutter plugin not installed; this adds Flutter specific functionality.
    X Dart plugin not installed; this adds Dart specific functionality.
[!] VS Code (version 1.52.1)
[!] Connected device
    ! No devices available

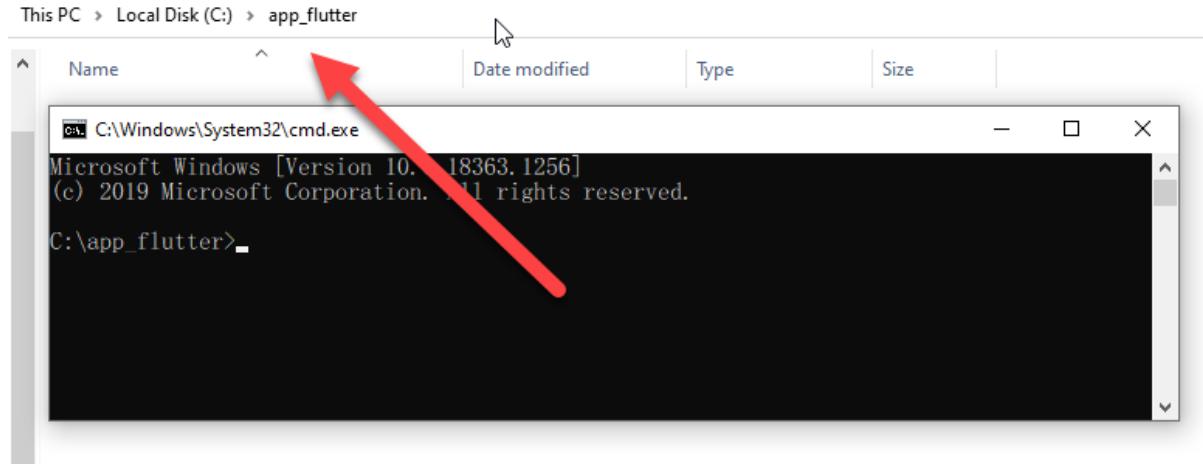
! Doctor found issues in 3 categories.

C:\Users\SOKU ASUS>
```

\* กรณี cmd แจ้ง error เกี่ยวกับ git จะต้องไปดาวน์โหลดและลงก่อน



## 4 Create App



- สร้าง Folder app\_flutter ขึ้นมาเพื่อเก็บแอพ แล้วเปิด cmd ขึ้นมาแล้ว cd c:\app\_flutter
- จากนั้นใช้คำสั่ง flutter create myapp จะได้ดังภาพด้านล่าง

```
C:\Windows\System32\cmd.exe
In order to run your application, type:
$ cd myapp
$ flutter run

Your application code is in myapp\lib\main.dart.

C:\app_flutter>
```

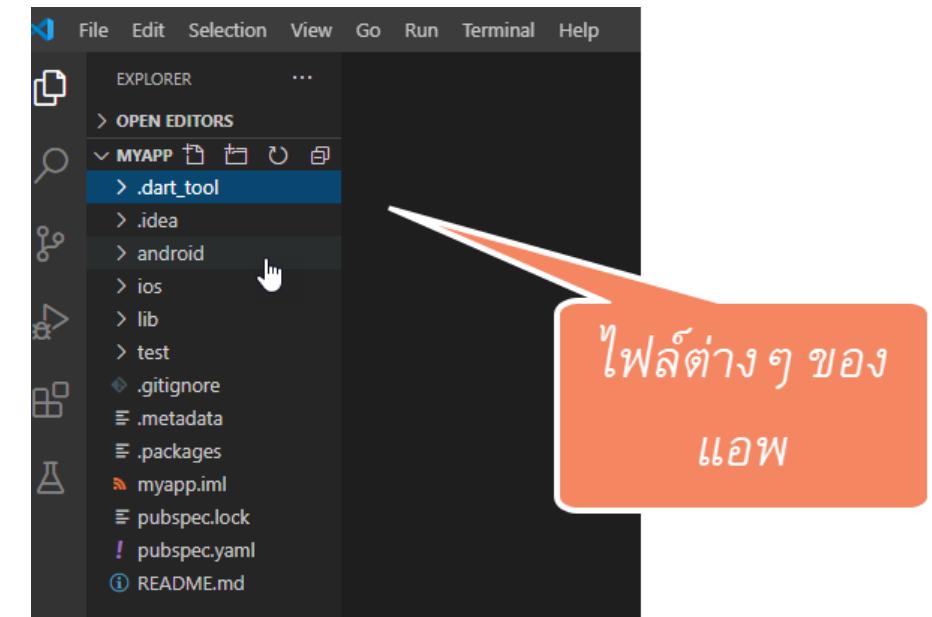
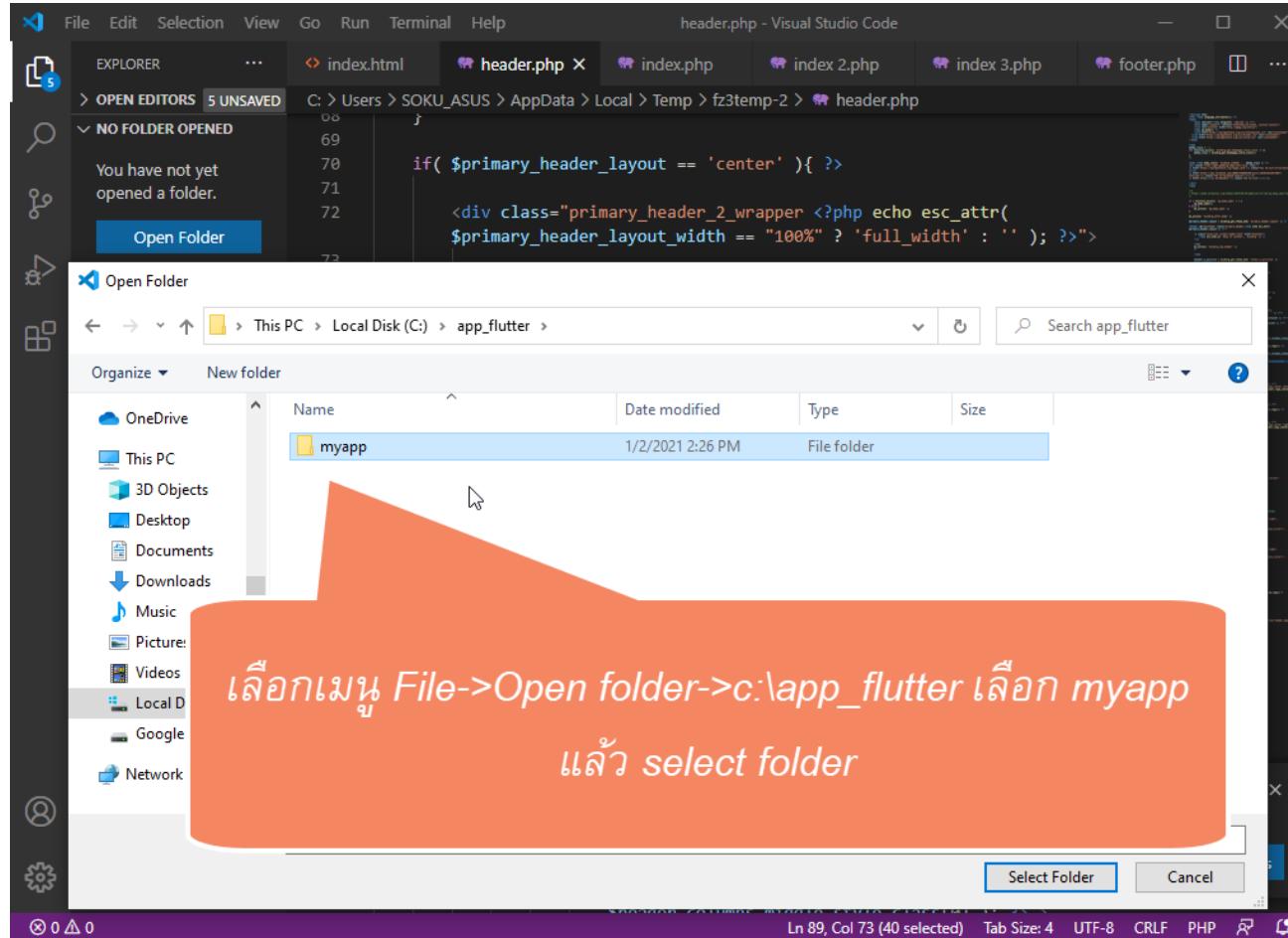
A screenshot of a Windows cmd window titled 'C:\Windows\System32\cmd.exe'. The window contains the following text:

In order to run your application, type:  
\$ cd myapp  
\$ flutter run

Your application code is in myapp\lib\main.dart.

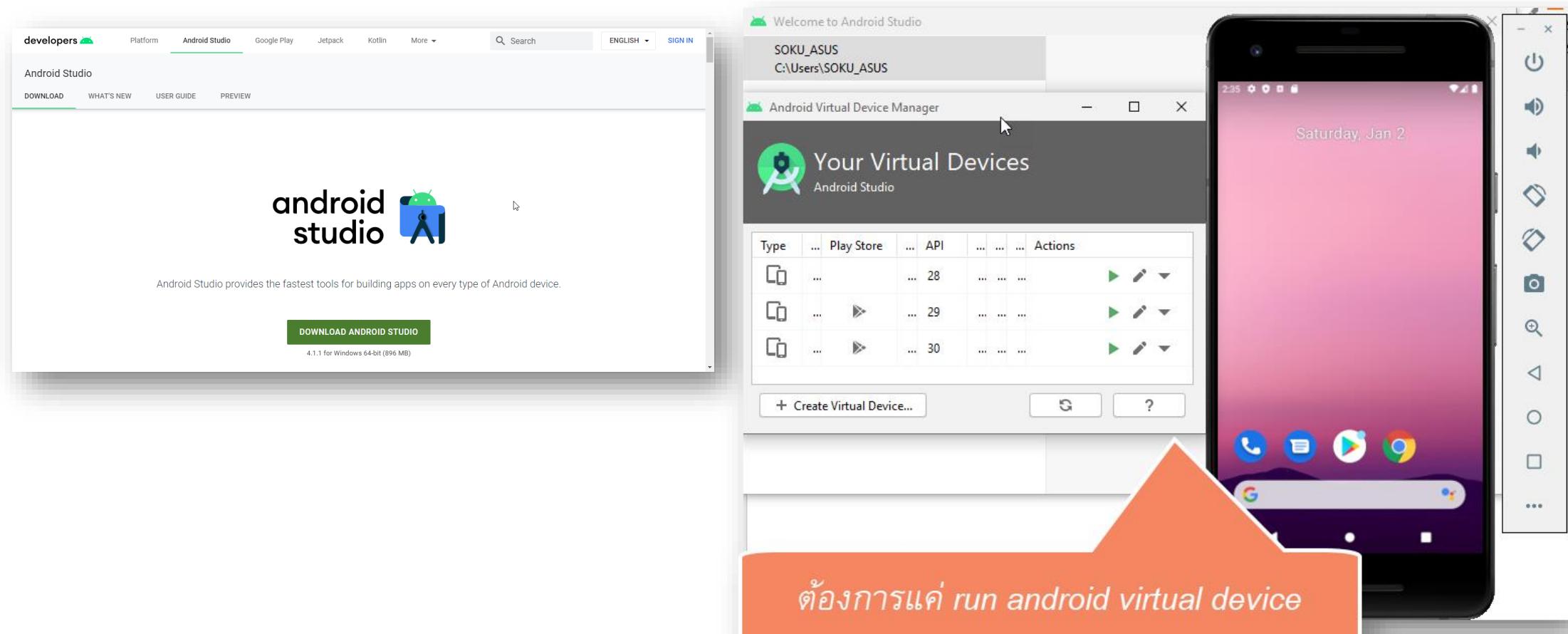
C:\app\_flutter>

# 5. เปิด Visual Studio Code ขึ้นมา เพื่อแก้ไข Code



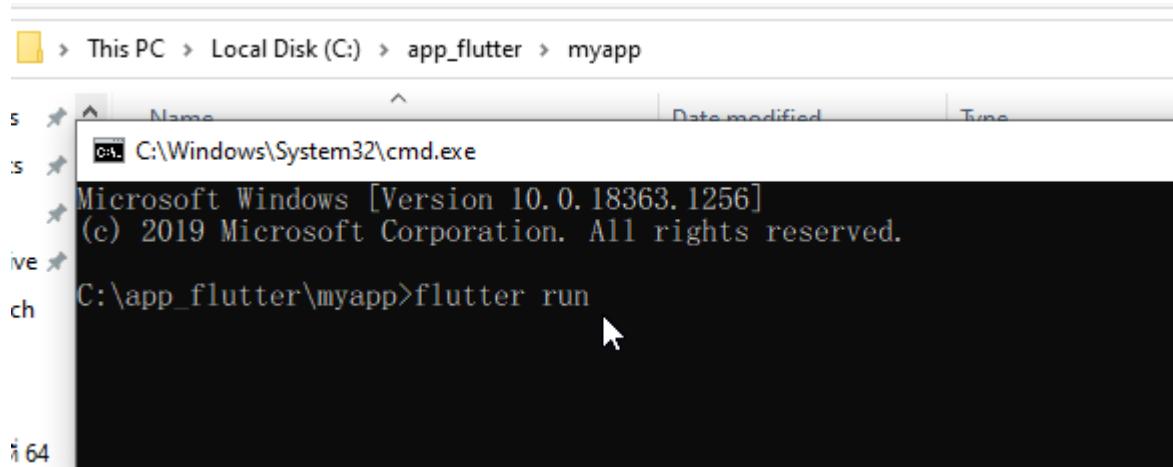
# 6 Install Android Studio

- <https://developer.android.com/studio> ดาวน์โหลดแล้ว install

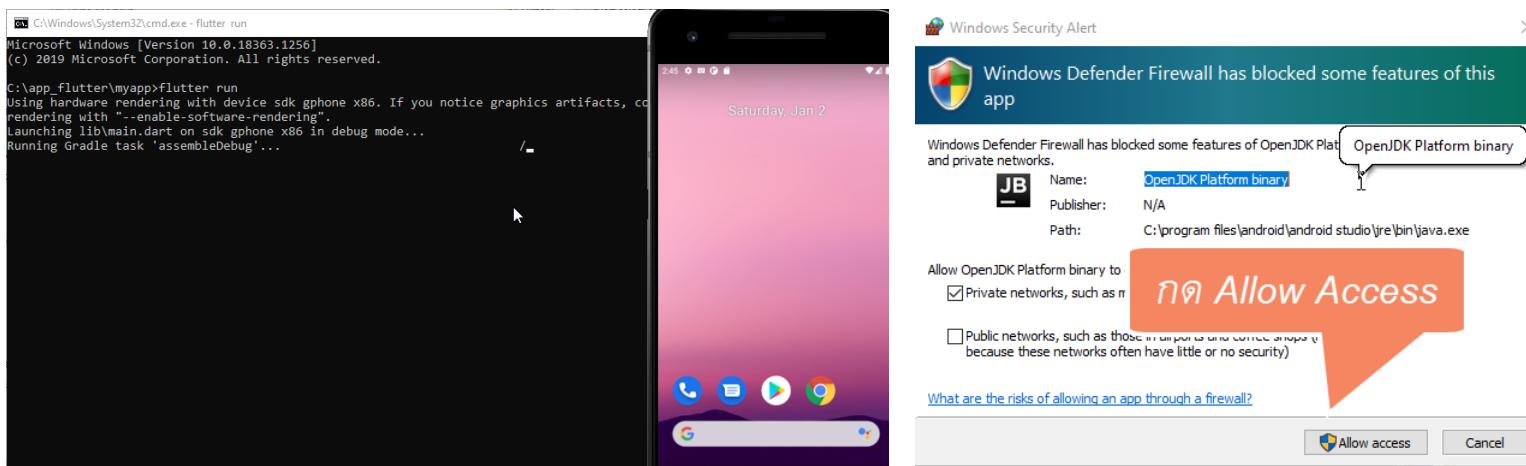


# 7. ให้ run flutter บน emulator ที่เปิดขึ้น

1. เปิด cmd ชี้ไป c:\app\_flutter\myapp



2. พิมพ์ flutter run



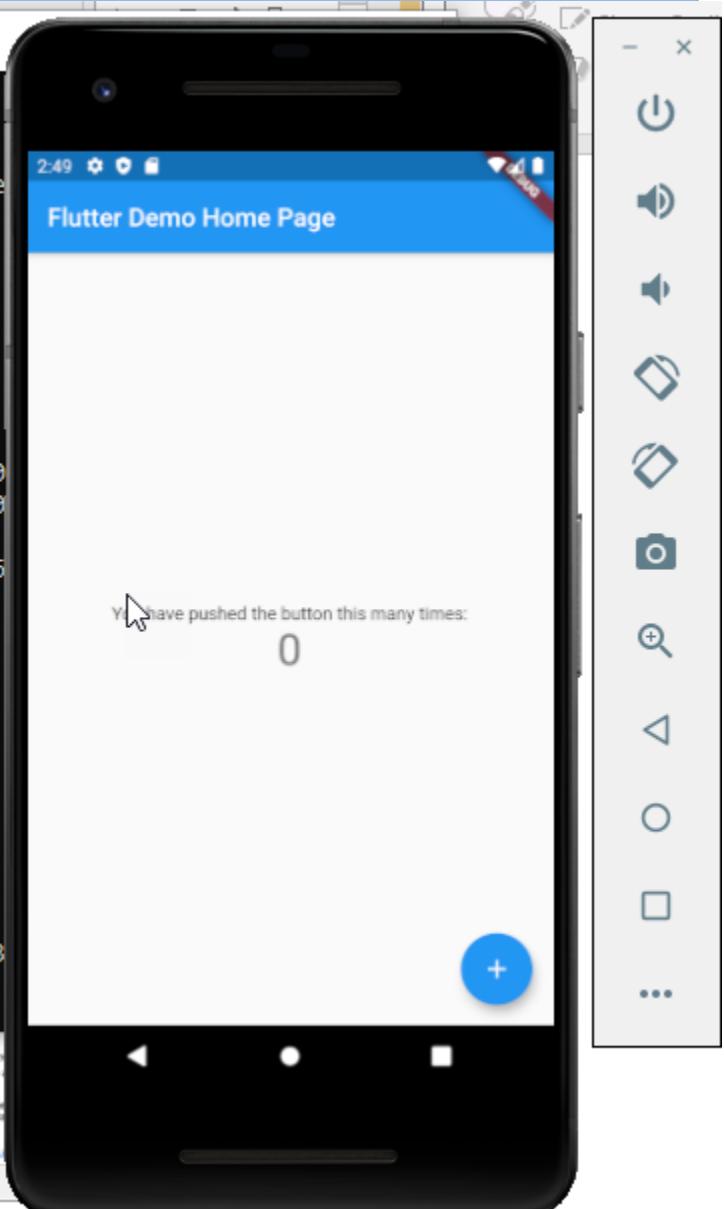
# พร้อมสำหรับการสร้างแอปแล้ว

```
ch: C:\Windows\System32\cmd.exe - flutter run
(c) 2019 Microsoft Corporation. All rights reserved.

C:\app_flutter\myapp>flutter run
Using hardware rendering with device sdk gphone x86. If you notice graphics artifacts, consider
rendering with "--enable-software-rendering".
Launching lib\main.dart on sdk gphone x86 in debug mode...
Running Gradle task 'assembleDebug'...

Running Gradle task 'assembleDebug'... Done                                215.7s (!)
✓ Built build\app\outputs\flutter-apk\app-debug.apk.
Installing build\app\outputs\flutter-apk\app.apk...                      1.5s
Waiting for sdk gphone x86 to report its views...                         10ms
I/OpenGLRenderer( 6301): Davey! duration=1020ms; Flags=1, IntendedVsync=307027078737, Vsync=30
nt=9223372036854775807, NewestInputEvent=0, HandleInputStart=307035043100, AnimationStart=3070
sStart=307035143800, DrawStart=307955220800, SyncQueued=307961463500, SyncStart=308120087800,
122044600, SwapBuffers=308165325600, FrameCompleted=308205726000, DequeueBufferDuration=145686
10900, GpuCompleted=0,
I/Choreographer( 6301): Skipped 70 frames!  The application may be doing too much work on its
Syncing files to device sdk gphone x86...                                     479ms

Flutter run key commands.
r Hot reload.
R Hot restart.
h Repeat this help message.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).
An Observatory debugger and profiler on sdk gphone x86 is available at: http://127.0.0.1:50893
I/m.example.myap( 6301): Waiting for a blocking GC ProfileSaver
```



File Edit Selection View Go Run Terminal Help main.dart - myapp - Visual Studio Code

EXPLORER main.dart

OPEN EDITORS lib > main.dart > MyHomePage

MYAPP .dart\_tool .idea android build ios lib main.dart test .gitignore .metadata .packages myapp.iml pubspec.lock pubspec.yaml README.md

21 // Notice that the counter didn't reset back to zero; the application  
22 // is not restarted.  
23 primarySwatch: Colors.blue,  
24 // This makes the visual density adapt to the platform that you run  
25 // the app on. For desktop platforms, the controls will be smaller and  
26 // closer together (more dense) than on mobile platforms.  
27 visualDensity: VisualDensity.adaptivePlatformDensity,  
28 ), // ThemeData  
29 home: MyHomePage(title: 'My SoKu App 2'),  
30 ); // MaterialApp  
31 }  
32 }  
33  
34 class MyHomePage extends StatelessWidget {  
35 MyHomePage({Key key}) : super(key: key);  
36 // This widget is the root of the application.  
37 // It has a child widget, the Counter widget.  
38 // That child widget is what the user sees.  
39 // It has a title bar.  
40 // This class is the home page of the application.  
41 // It has a title bar.  
42 // case the user wants to go back.  
43 // used to be called MainWidget.  
44  
45 PROBLEMS OUT

r Hot reload.  
R Hot restart.  
h Repeat this help message.  
d Detach (terminate "flutter run" but leave application running).  
c Clear the screen.  
q Quit (terminate the application on the device).  
An Observatory debugger and profiler on sdk gphone x86 is available at: http://127.0.0.1:51245/hEIsqIEw5oo=/

Performing hot reload...

My SoKu App 2

ให้ใช้ Terminal และ พิมพ์ flutter run ทำให้ง่ายต่อการใช้รีสอร์ฟ หรือ hot reload และอีกครั้งทำให้แก้ไขง่ายขึ้น

แอปจะเปลี่ยนตามที่แก้ไขแล้ว

ลบ Code ทั้งหมดให้เหลือแค่ void main และพิมพ์ด้านหลัง =>

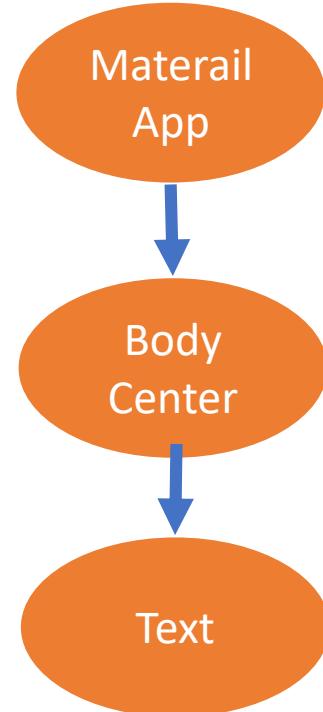
The screenshot shows a Flutter project in a code editor. The code in `main.dart` is:

```
1 import 'package:flutter/material.dart';
2
3 void main() => runApp(MaterialApp(home: Center(child: Text('Hello World'))));
4
```

To the right, a mobile device screen displays the application with the text "Hello World" centered in red font with a yellow underline.

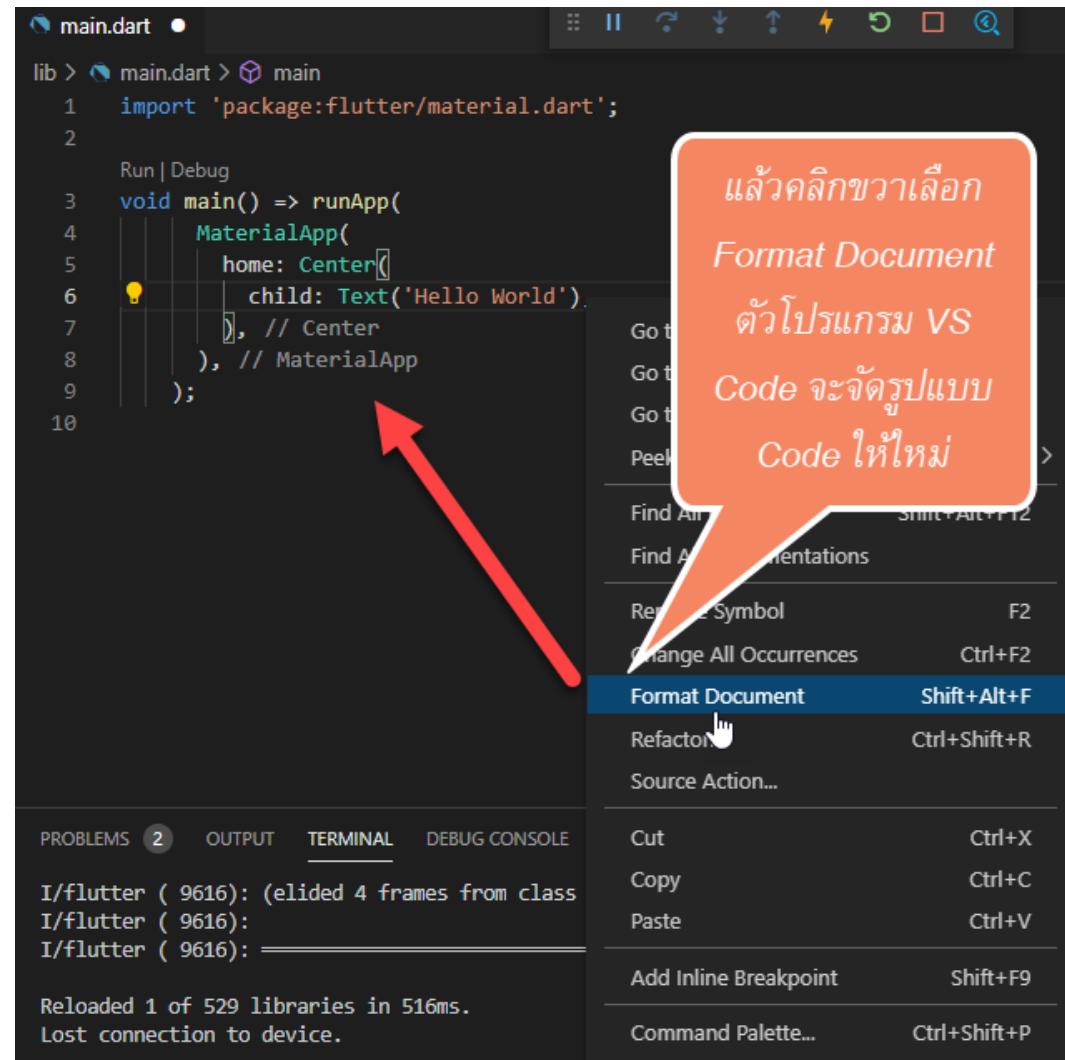
At the bottom, the terminal shows:

```
I/flutter ( 9616): (elided 4 frames from class _RawReceivePortImpl, class _Timer, and dart:async-patch)
I/flutter ( 9616):
```



```
b > main.dart > ...
1 import 'package:flutter/material.dart';
2
3 Run | Debug
3 void main() => runApp(MaterialApp(home: Center(child: Text('Hello World'),),));
4
```

ໃຫ້ສື່ສຸກນໍາຄົນໃນວັງເລີບ



The screenshot shows the VS Code interface with the following details:

- Code Editor:** Displays the file `main.dart` containing Dart code. A yellow lightbulb icon is shown at line 6, column 10, indicating a potential issue.
- Context Menu:** A context menu is open over the code editor, listing various code-related commands. The command `Format Document` is highlighted with a blue background and a mouse cursor icon.
- Bottom Status Bar:** Shows the following information:
  - PROBLEMS (2)
  - OUTPUT
  - TERMINAL
  - DEBUG CONSOLE
  - I/flutter ( 9616): (elided 4 frames from class
  - I/flutter ( 9616):
  - I/flutter ( 9616): \_\_\_\_\_
  - Reloaded 1 of 529 libraries in 516ms.
  - Lost connection to device.

ແລ້ວຄູ່ລົກຂວາເລື່ອກ

*Format Document*

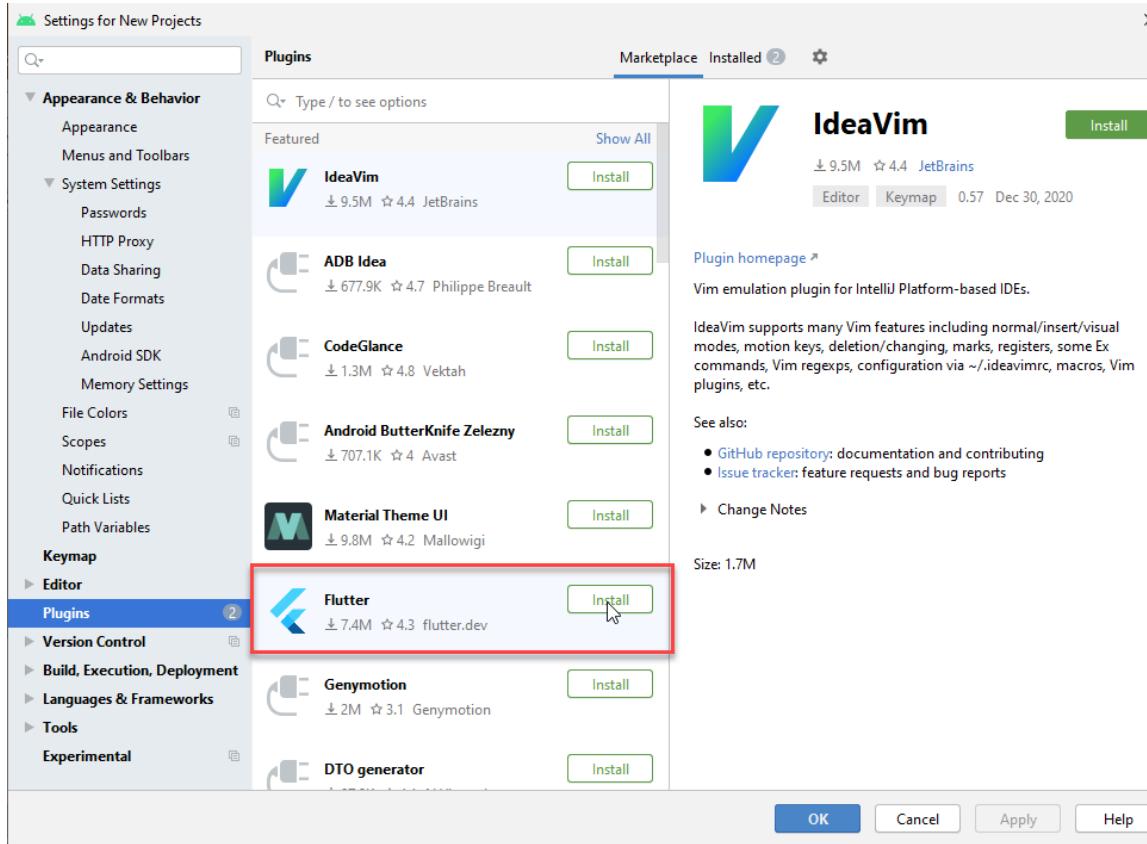
ຕັ້ງໂປຣແກຣມ VS

Code ຈະຈັດຮັບແບບ

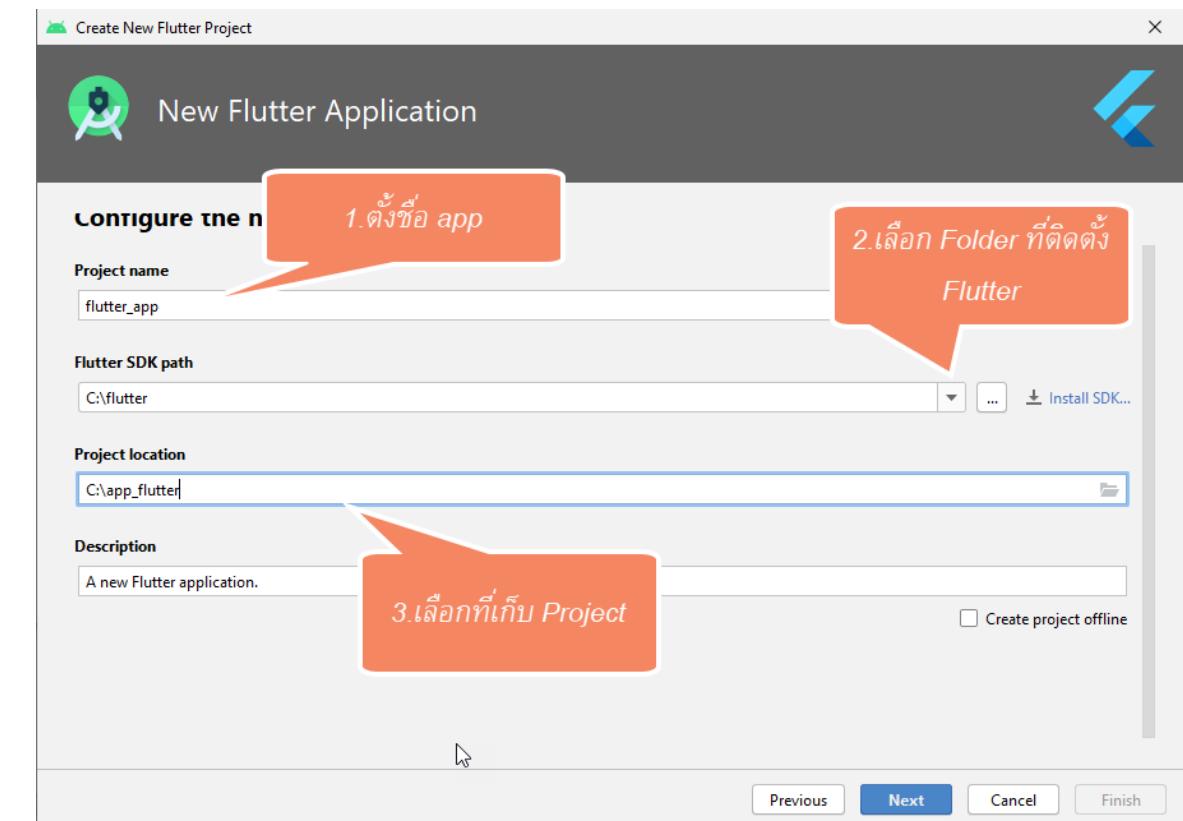
Code ໃຫ້ທຶນ

# การสร้าง Project บน Android Studio

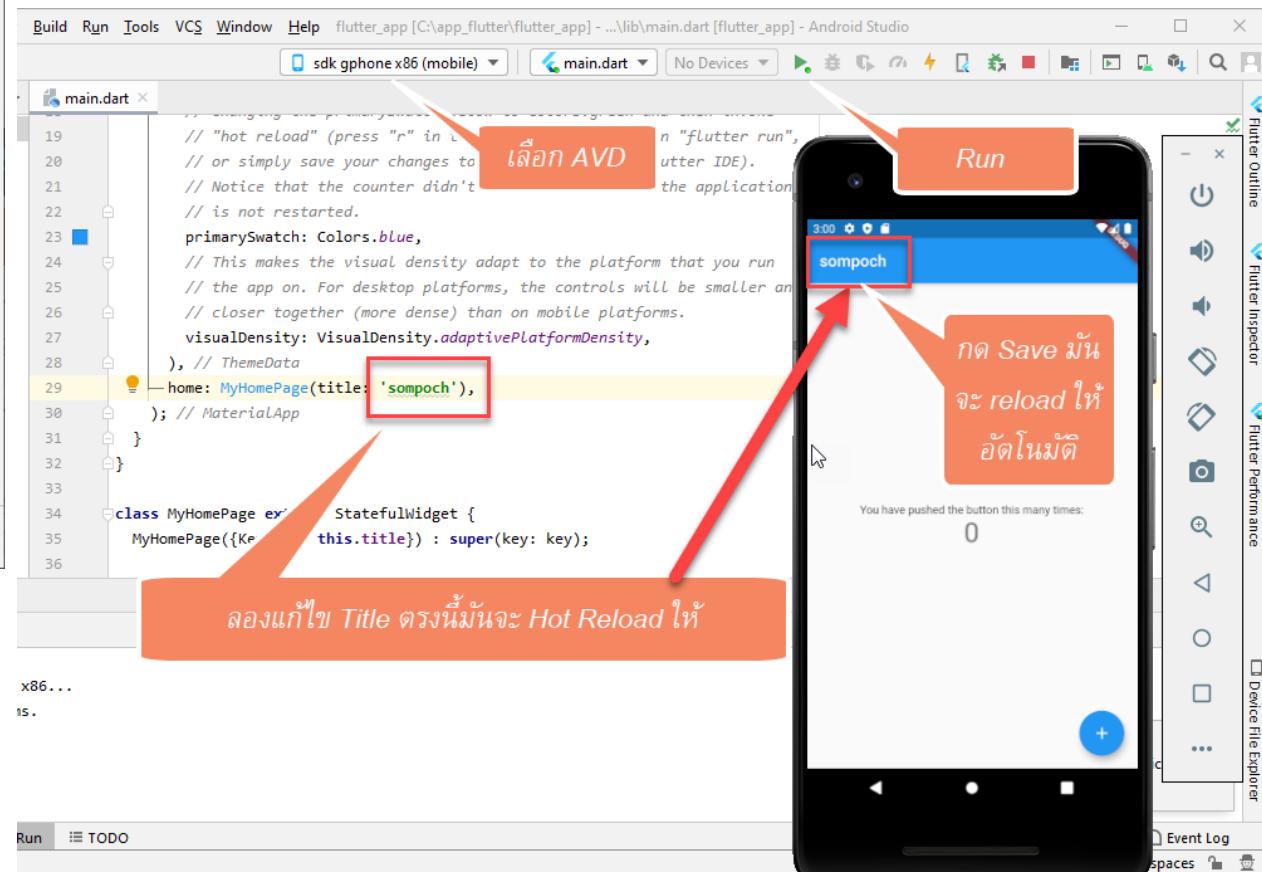
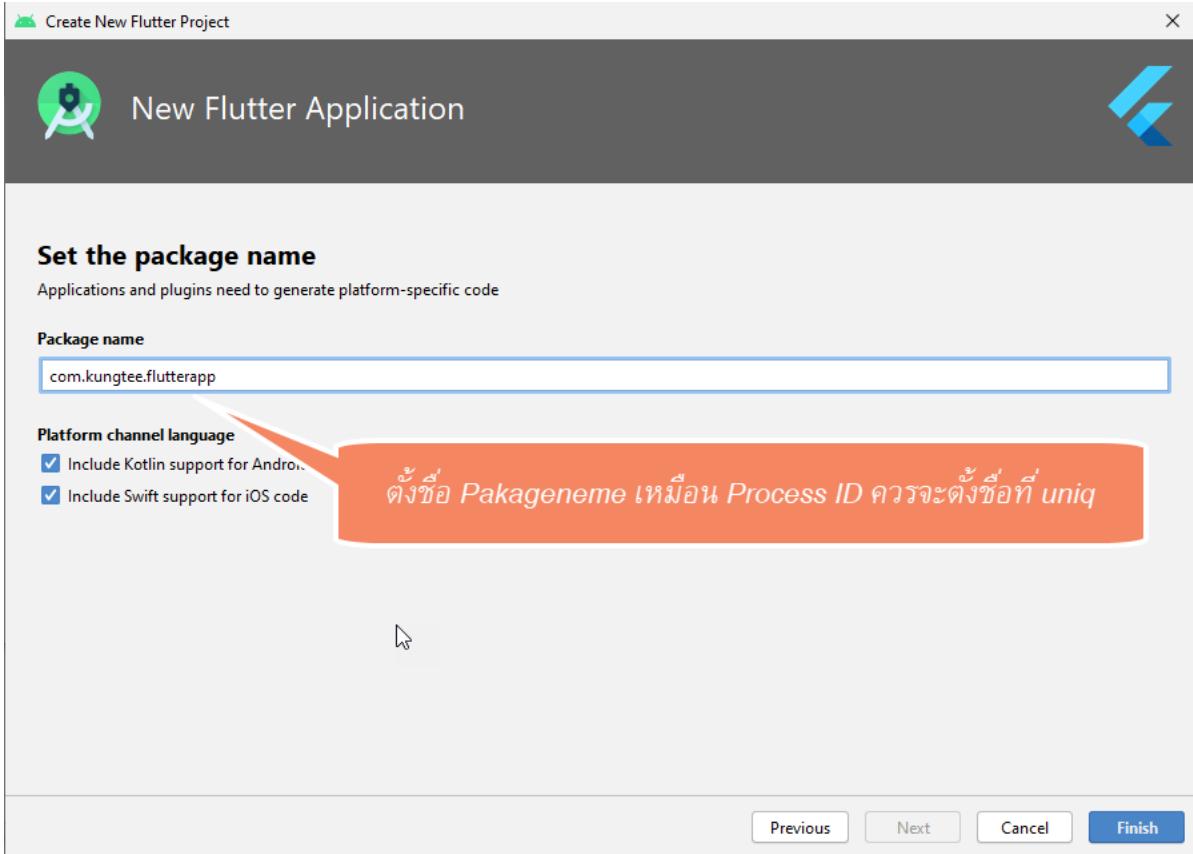
- เปิด Android Studio ขึ้นมาแล้วไป Config->Setting->plugin ทำการ Install Flutter



หลังจากนั้นให้ New Project เลือก New Flutter Application  
แล้วตั้งค่าตามภาพด้านล่าง

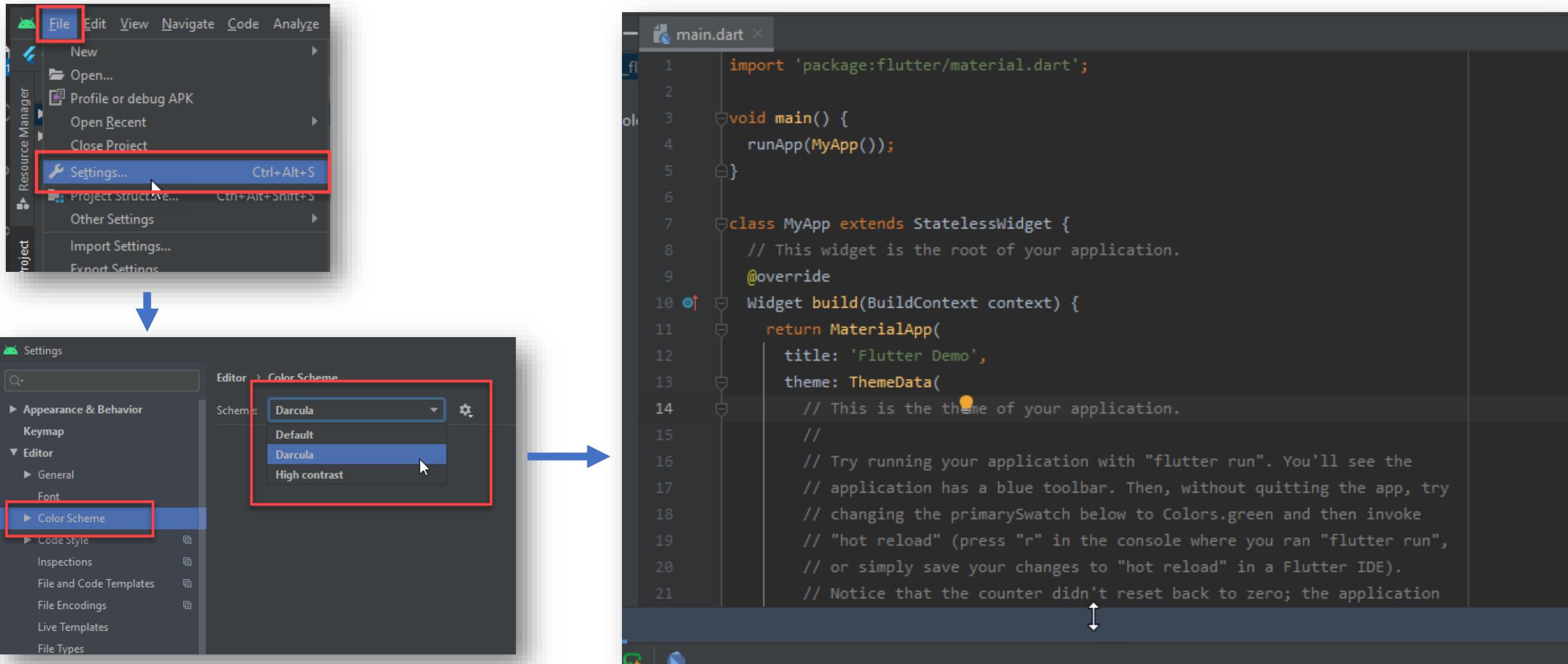


# การสร้าง Project บน Android Studio

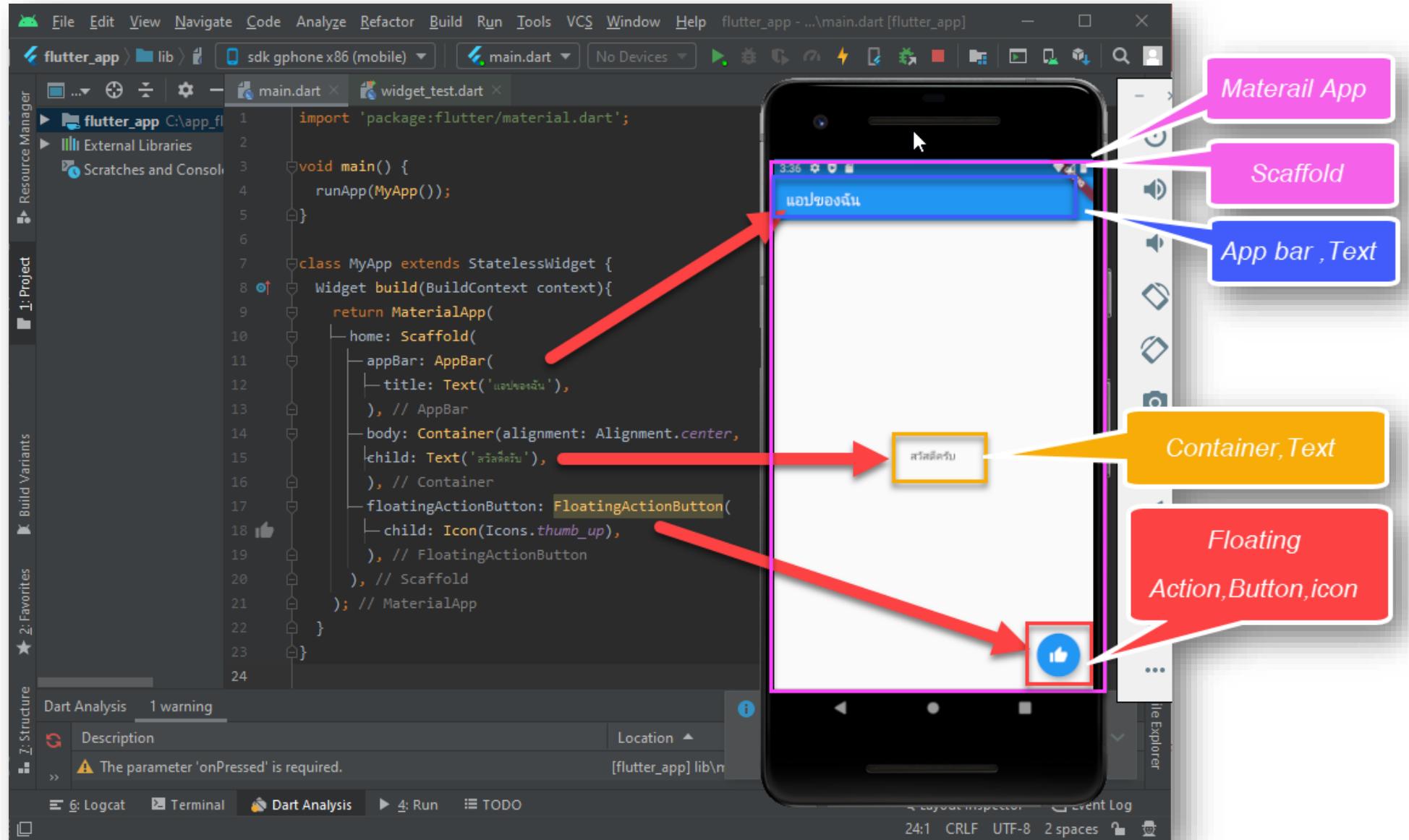


# Android Studio เปลี่ยนสี theme ให้เป็นสีดำเพื่อจะเขียนโค้ดสบายๆ

- ไปที่เมนู File->Settings->Color Schema->Darcula



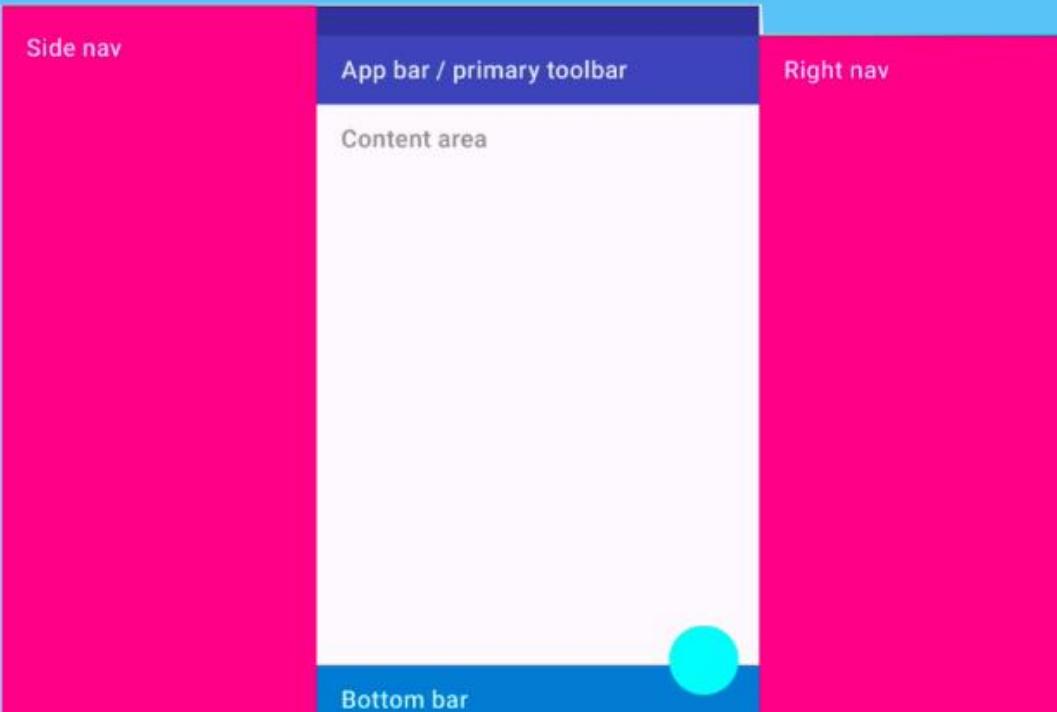
# ตัวอย่างแรกสร้าง AppBar สร้าง Content สร้างปุ่มกด Like



# การใช้ Scaffold() Widget

- Scaffold() คือ widget ที่ช่วยให้เราออกแบบแอพได้สวยงาม มี element ให้เลือกใช้มากมาย เช่น ปุ่มที่ลอยอยู่ตรงหน้าจอ

## Scaffold() Widget

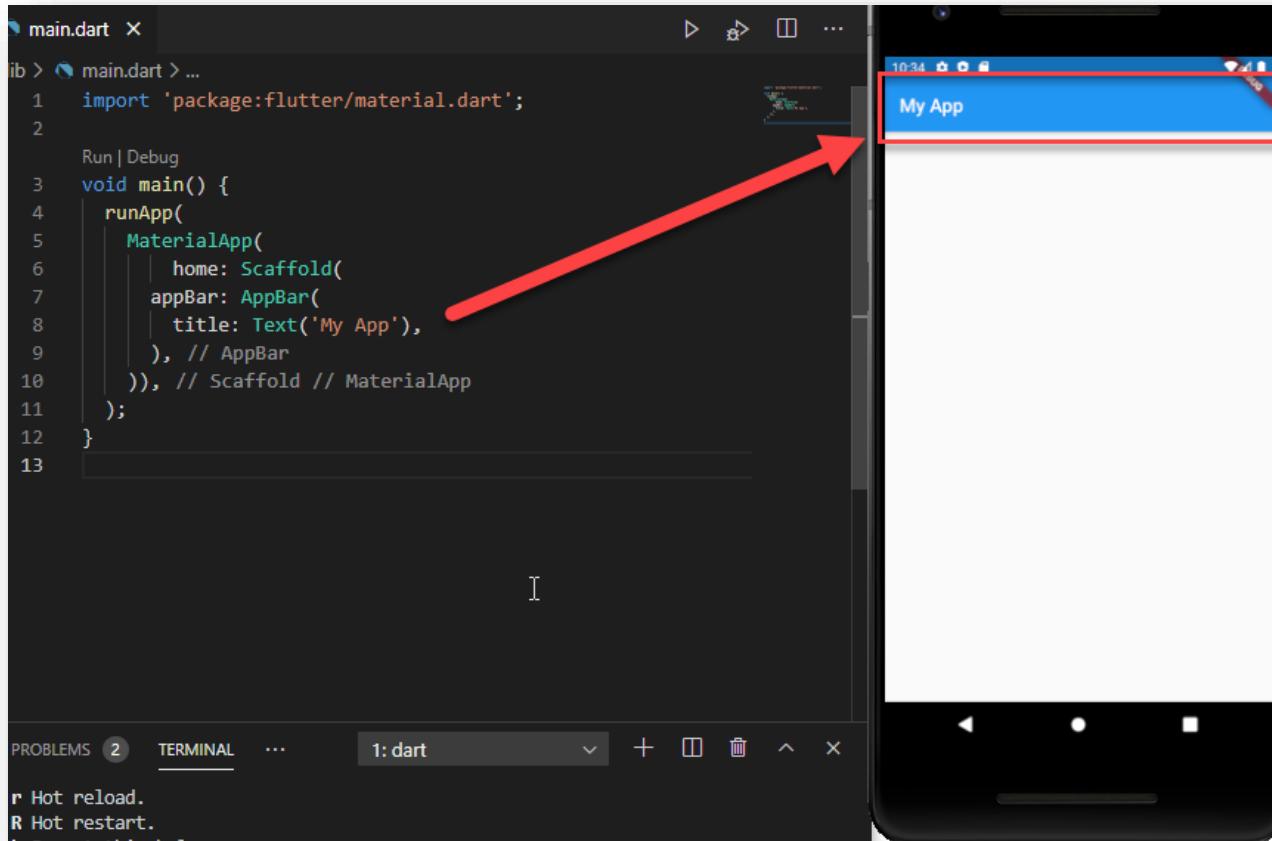


```
import 'package:flutter/material.dart';

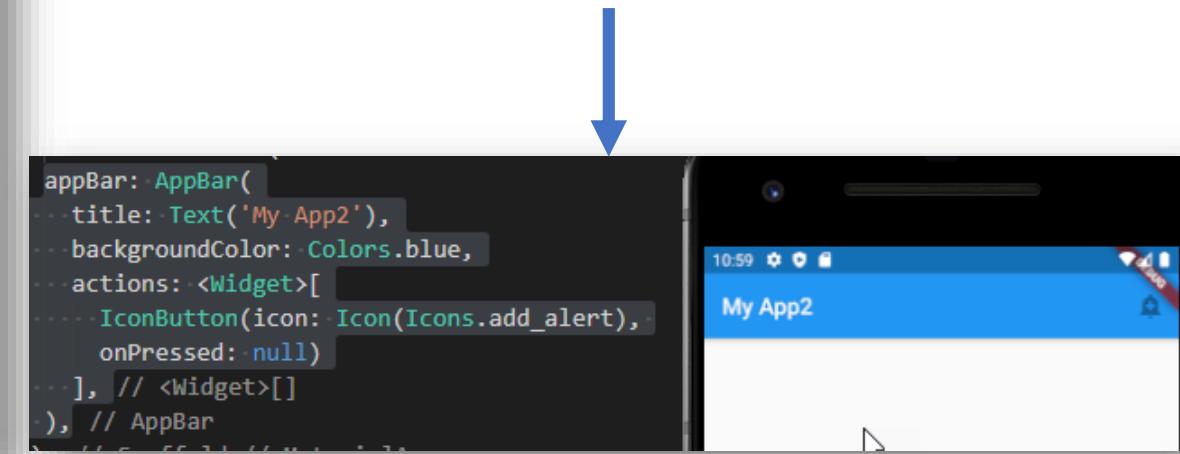
Run | Debug
void main() {
  runApp(
    MaterialApp(
      home: Scaffold(),
    ), // MaterialApp appBar: PreferredSizeWidget
  );
}
```

[@] backgroundColor:  
[@] body:  
[@] bottomNavigationBar:  
[@] bottomSheet:  
[@] drawer:  
[@] drawerDragStartBehavior:  
[@] drawerEdgeDragWidth:  
[@] drawerEnableOpenDragGesture:  
[@] drawerScrimColor:  
[@] endDrawer:  
[@] endDrawerEnableOpenDragGesture:

# Scaffold() สร้าง AppBar

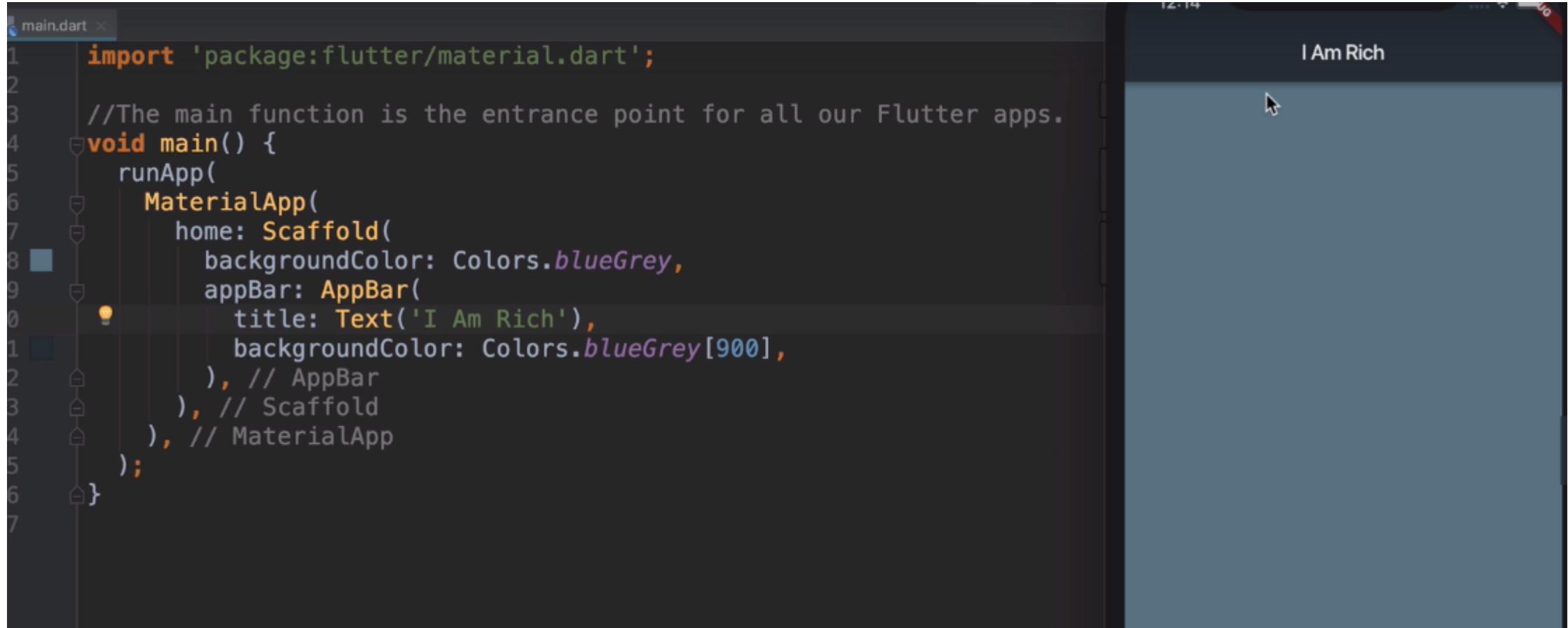


ใน AppBar จะมี Properties ให้แก้ไขได้เยอะมาก เช่น  
ใส่ข้อความ เปลี่ยนสี BackGround ใส่ icon



- <https://api.flutter.dev/flutter/material/AppBar-class.html>
- สามารถดูสีได้ที่ <https://material.io/design/color/the-color-system.html#tools-for-picking-colors>

# Body Color



The screenshot shows a Flutter application running on an iPhone X simulator. The app has a dark blue-grey background. At the top, there is a white navigation bar with the title "I Am Rich". The code editor on the left shows the main.dart file with the following code:

```
1 import 'package:flutter/material.dart';
2
3 //The main function is the entrance point for all our Flutter apps.
4 void main() {
5   runApp(
6     MaterialApp(
7       home: Scaffold(
8         backgroundColor: Colors.blueGrey,
9         appBar: AppBar(
10           title: Text('I Am Rich'),
11           backgroundColor: Colors.blueGrey[900],
12         ), // AppBar
13         ), // Scaffold
14       ), // MaterialApp
15     );
16 }
```

The code defines a simple Flutter application with a single screen. The screen uses a `Scaffold` widget as its base. The `backgroundColor` of the `Scaffold` is set to `Colors.blueGrey`. The `AppBar` has a title "I Am Rich" and a background color of `Colors.blueGrey[900]`, which is a darker shade of blue-grey.

# Body การแทรก Image

The screenshot shows the Android Studio interface. On the left, the code editor displays the `main.dart` file. A yellow box highlights the following code block:

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('แอปของฉัน'),
        ), // AppBar

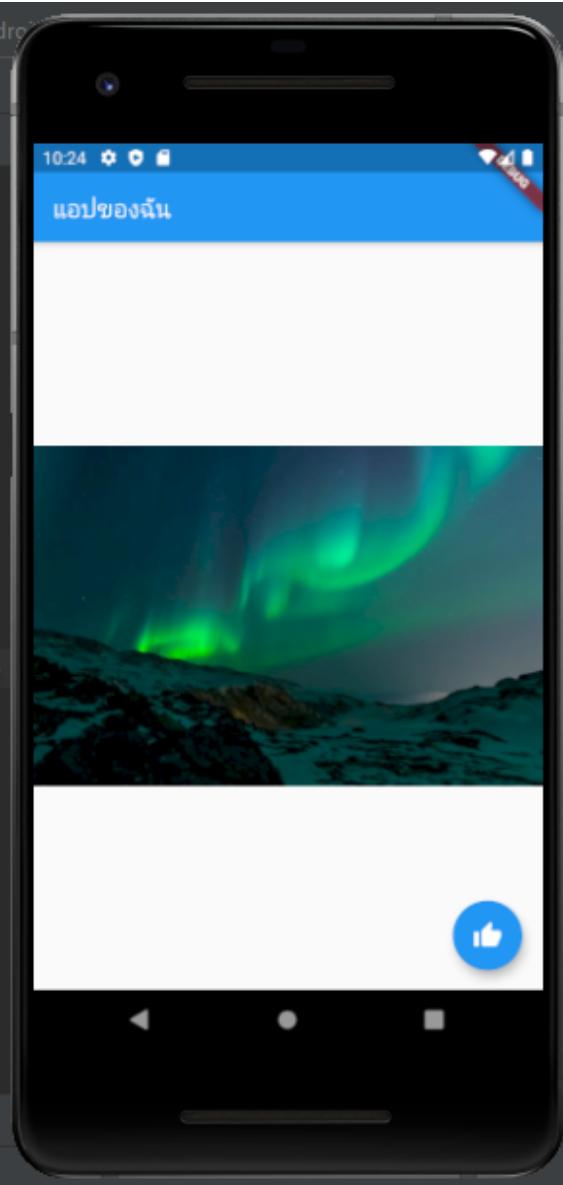
        body: Image(
          image: NetworkImage('https://www.w3schools.com/w3css/img_lights.jpg'),
        ), // Image

        floatingActionButton: FloatingActionButton(
          child: Icon(Icons.thumb_up),
        ), // FloatingActionButton
      ), // Scaffold
    ); // MaterialApp
  }
}
```

The middle part of the interface shows the emulator window displaying a mobile application. The app has a blue header bar with the text "แอปของฉัน". The main content area shows a photograph of a camera and some flowers. A blue floating action button with a thumbs-up icon is at the bottom right. The bottom of the screen shows the Android navigation bar. On the right, there is a vertical toolbar with various icons for device control.

```
flutter: I/OpenGLRenderer(14447): Skipped 50 frames! The application may be doing too much work on its main thread.
```

# Body Image ถ้าต้องการให้อยู่กลางจะต้องใช้ Center Widget และสร้าง Child



The screenshot shows the Android Studio IDE with the main.dart file open. A red box highlights the code block from line 15 to 18, which defines a Center widget containing an Image widget. An arrow points from this highlighted code to the corresponding visual representation on the right.

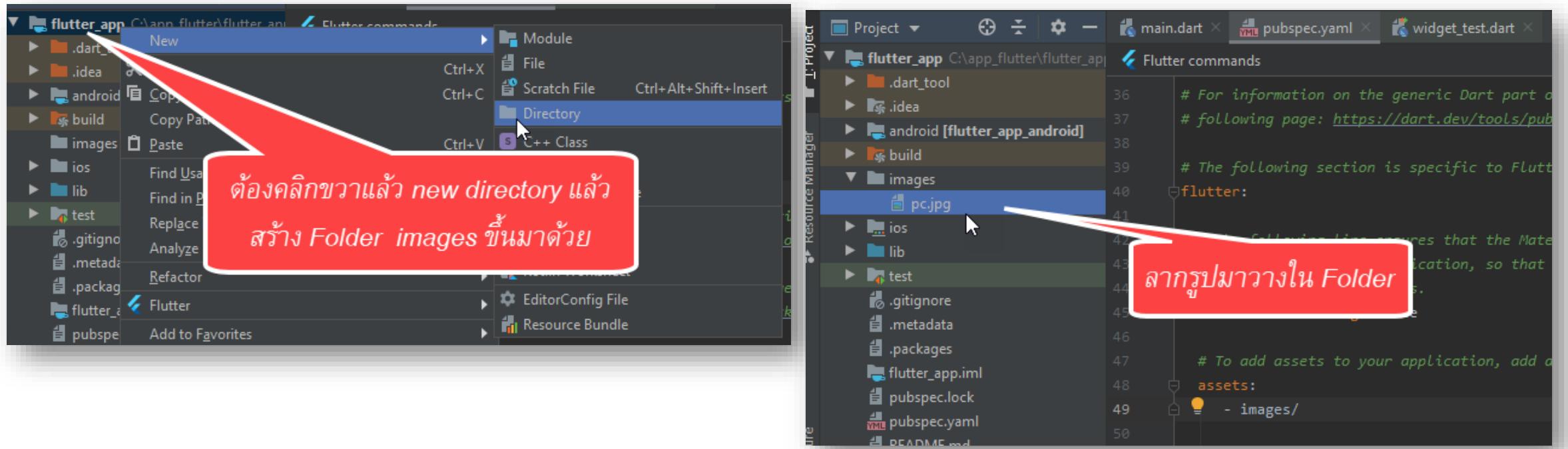
```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('ແອປຂອງຈັນ'),
        ), // AppBar

        body: Center(
          child: Image(
            image: NetworkImage('https://www.w3schools.com/w3css/img_lights.jpg'),
          ), // Image
        ), // Center

        floatingActionButton: FloatingActionButton(
          child: Icon(Icons.thumb_up),
        ), // FloatingActionButton
      ), // Scaffold
    ); // MaterialApp
  }
}
```

# กรณีที่ต้องการเอารูปภาพมาใส่ไว้ใน app เลย

- ในกรณีนี้จะต้องไปเปิดใช้งานในไฟล์ pubspec.yaml (YAML เป็น markup language คล้ายๆ ภาษา html ที่สำหรับใช้ config เป็นภาษาที่มนุษย์เข้าใจง่าย แต่เครื่องคอมพิวเตอร์จะต้องแปลงก่อน) ไปเปิด comment ในส่วนของ asset เพื่อให้โปรแกรมรู้จัก และกด pub get เพื่อให้ update ค่า config



# กรณีที่ต้องการเอารูปภาพมาใส่ไว้ใน app เลย (ต่อ)

- เอา # ออก เพื่อเปิดใช้งาน

The screenshot shows the Android Studio interface with the pubspec.yaml file open. A red callout box points to the code at line 49: "assets: - images/". The text inside the box reads: "ตอนเอา # ออกต้องระวังต้องเว้นช่องว่าง 2 เคาะ ด้านหน้าด้วย". Another red callout box points to the "Pub get" button in the top right corner of the editor. The text inside the box reads: "กด Pub get เพื่อ update config ด้วย".

```
# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter.
flutter:
  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons
  # from the Material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like this:
  assets:
    - images/
```

# กรณีที่ต้องการเอารูปภาพมาใส่ไว้ใน app เลย (ต่อ)

- กลับมาแก้ไขโค้ดในไฟล์ lib/main.dart

The screenshot shows the Android Studio interface with the following details:

- File Structure:** The project tree on the left shows a folder named "images" containing a file "pc.jpg".
- Code Editor:** The main.dart file contains the following code:

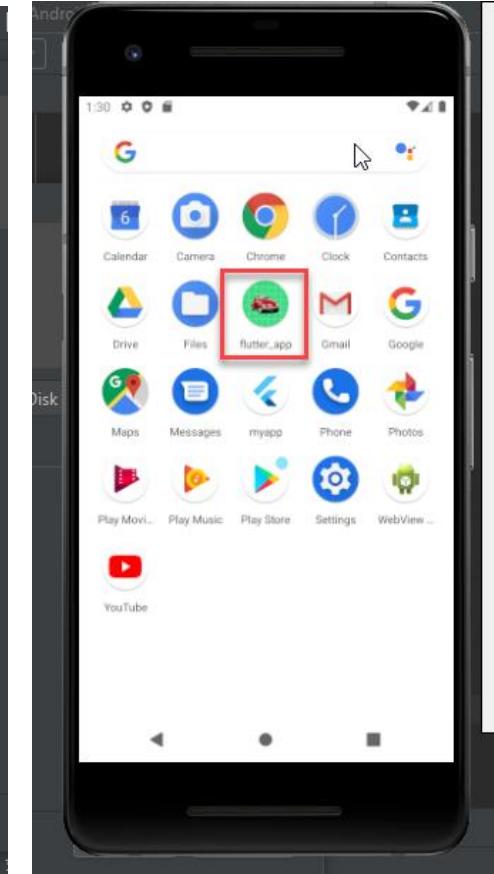
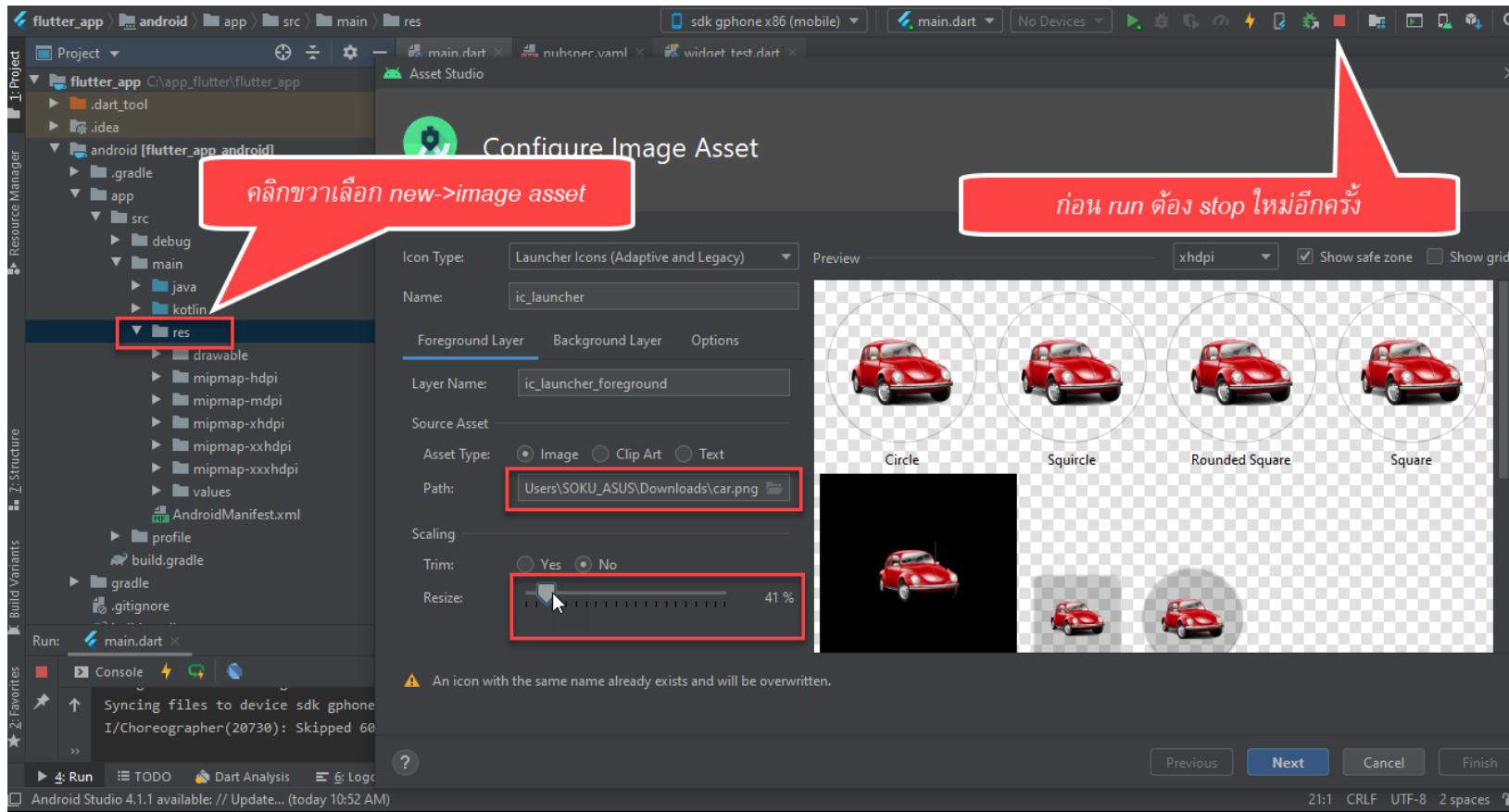
```
class MyApp extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            home: Scaffold(
                appBar: AppBar(
                    title: Text('แอปของฉัน'),
                ), // AppBar

                body: Center(
                    child: Image(
                        image: AssetImage('images/pc.jpg'),
                    ), // Image
                ), // Center

                floatingActionButton: FloatingActionButton(
                    child: Icon(Icons.thumb_up),
                ), // FloatingActionButton
            ), // Scaffold
        ); // MaterialApp
    }
}
```
- Virtual Device:** On the right, a virtual device (sdk gphone x86 (mobile)) displays the app's UI. It features a blue header bar with the text "แอปของฉัน". The main content area shows a laptop screen displaying a credit card with the number "1234 5678 9012 3456" and the text "USER NAME" and "PASSWORD". A large red arrow points from the highlighted code in the editor to the image displayed on the device screen.
- Bottom Bar:** The bottom navigation bar includes tabs for "main.dart", "Console", and other development tools.

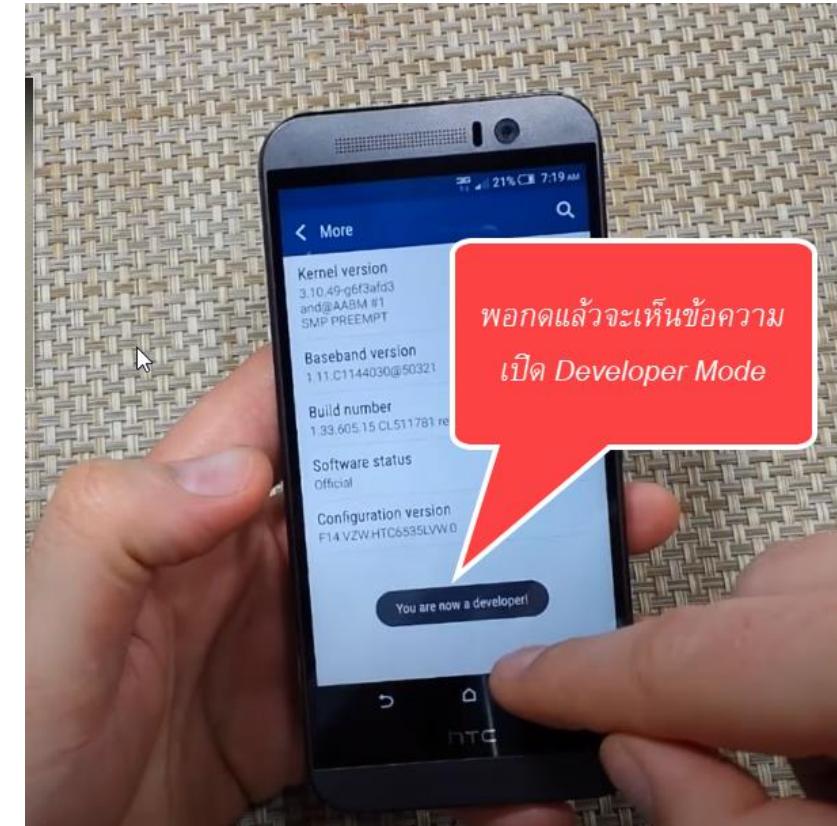
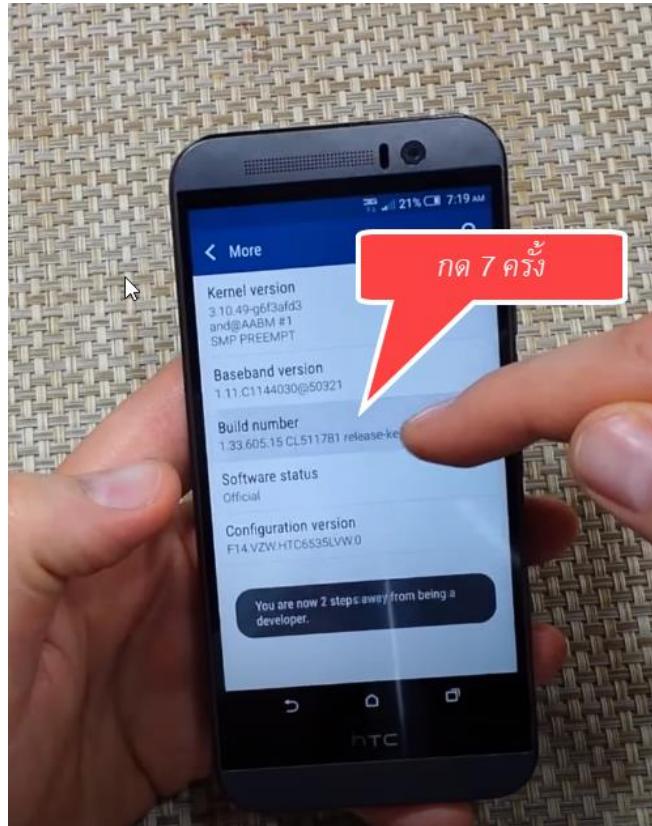
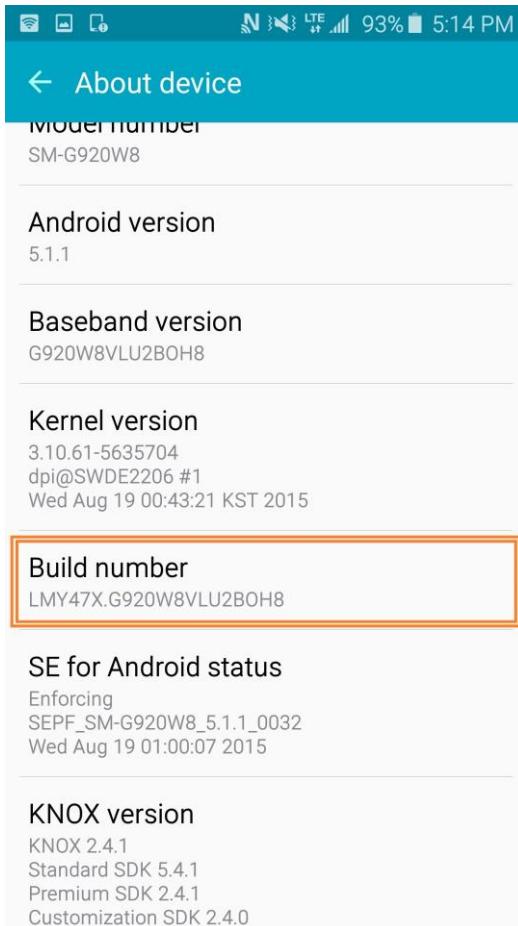
# การสร้าง icon บน ios และ android

- หารูป icon ที่ต้องการหรือไปดาวน์โหลดมาจากเว็บ iconarchive นามสกุล png
- ใน android studio เลือก android->app->src->main->res และคลิกขวาที่ res เลือก new เลือก image assets
- ให้เลือกไฟล์ icon ที่จะสร้าง แล้วปรับขนาดดูภาพจากซองด้านขวา แล้วกด next -> finish แล้วรอง run ดู



# การ Deploy app สู่ physical android device

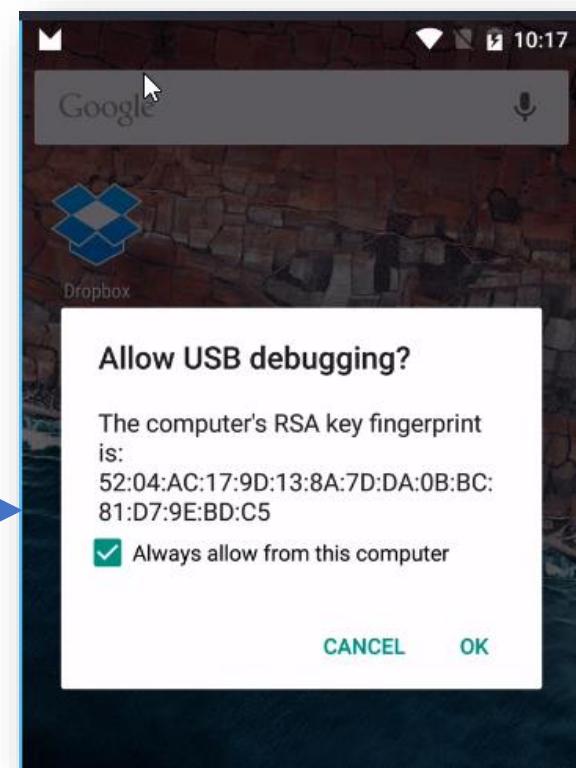
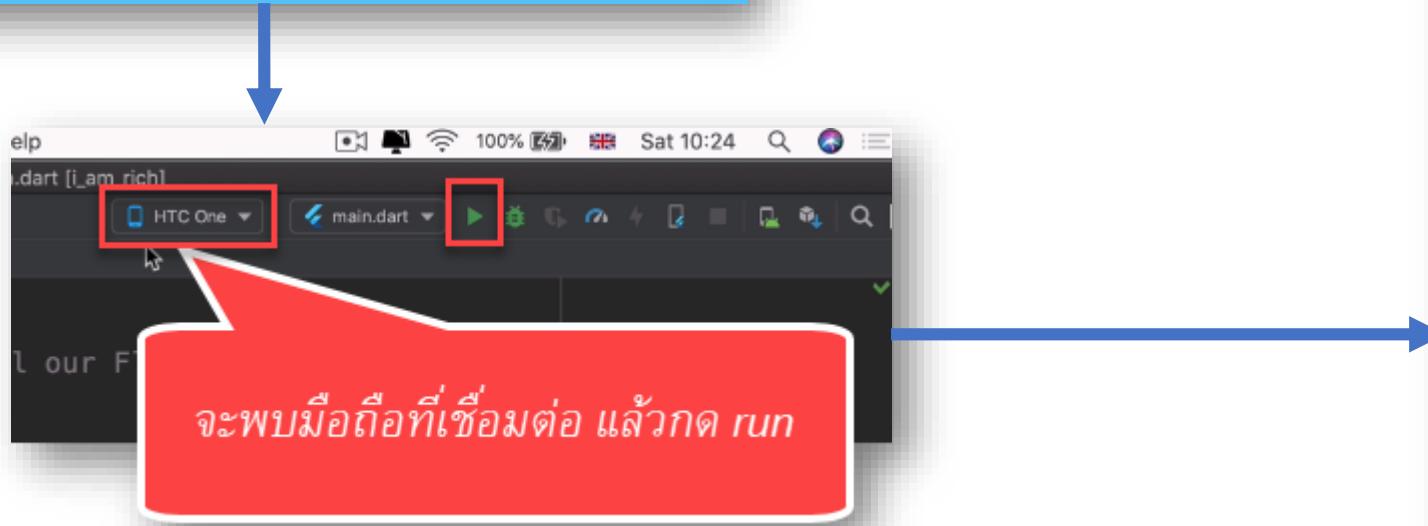
- ต้องนำเครื่องมือถือ android เข้าไปใน setting ไปหา build number กด 7 ครั้งเพื่อเปิด Developer Mode เพื่อให้อุปกรณ์ android สามารถเสียบสาย usb แล้ว debug app ได้ เพื่อจะได้ลงแอพที่ทำจากโปรแกรมเราได้



# การ Deploy app สู่ physical android device (ต่อ)

## Steps

1. Enable Developer Mode (Phone)
2. Enable USB Debugging (Phone)
3. Connect Your Phone with USB
4. Trust Your Computer if Prompted (Phone)



หลังจากนั้น กด run ใน android studio ที่เชื่อมกับมือถือแล้ว ในหน้าจอ มือถือจะแสดงข้อความให้กด allow usb debugging

# การเอา Icon Debug Banner ออกจากโค้ด Flutter



ให้ใส่ `debugShowCheckedModeBanner: false,` ก่อน `home`

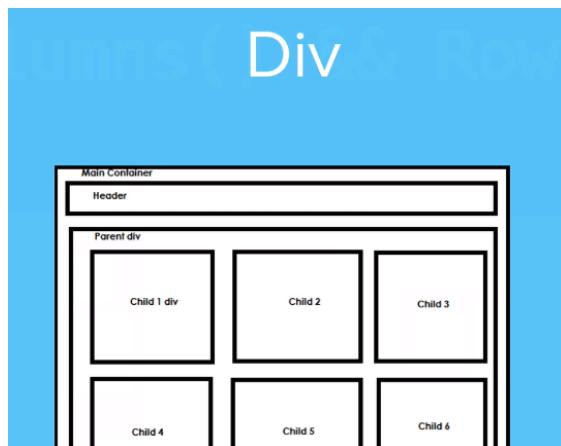
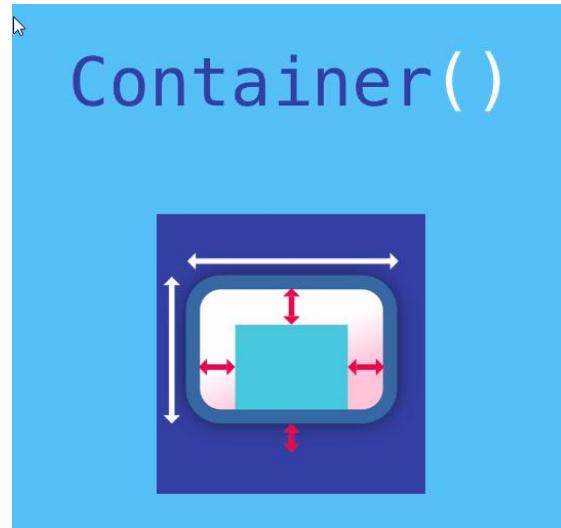
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget{
  Widget build(BuildContext context){
    var hello='สวัสดี';
    return MaterialApp(
      debugShowCheckedModeBanner: false, // This line is highlighted with a red box
      home:Scaffold(
        appBar: AppBar(
          backgroundColor: Colors.amberAccent,
          title:Text('My App Flutter day 1'),
          actions: <Widget>[
            IconButton(icon: Icon(Icons.ac_unit), onPressed: (){
              print('click appBar');
            })
          ] // IconButton
      )
    );
  }
}
```



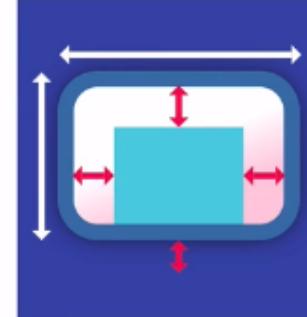
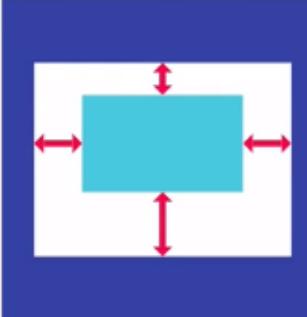
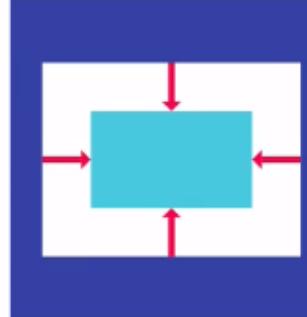
# การวาง Container Layout



Layout widgets - Flutter <https://flutter.dev/docs/development/ui/widgets/layout>

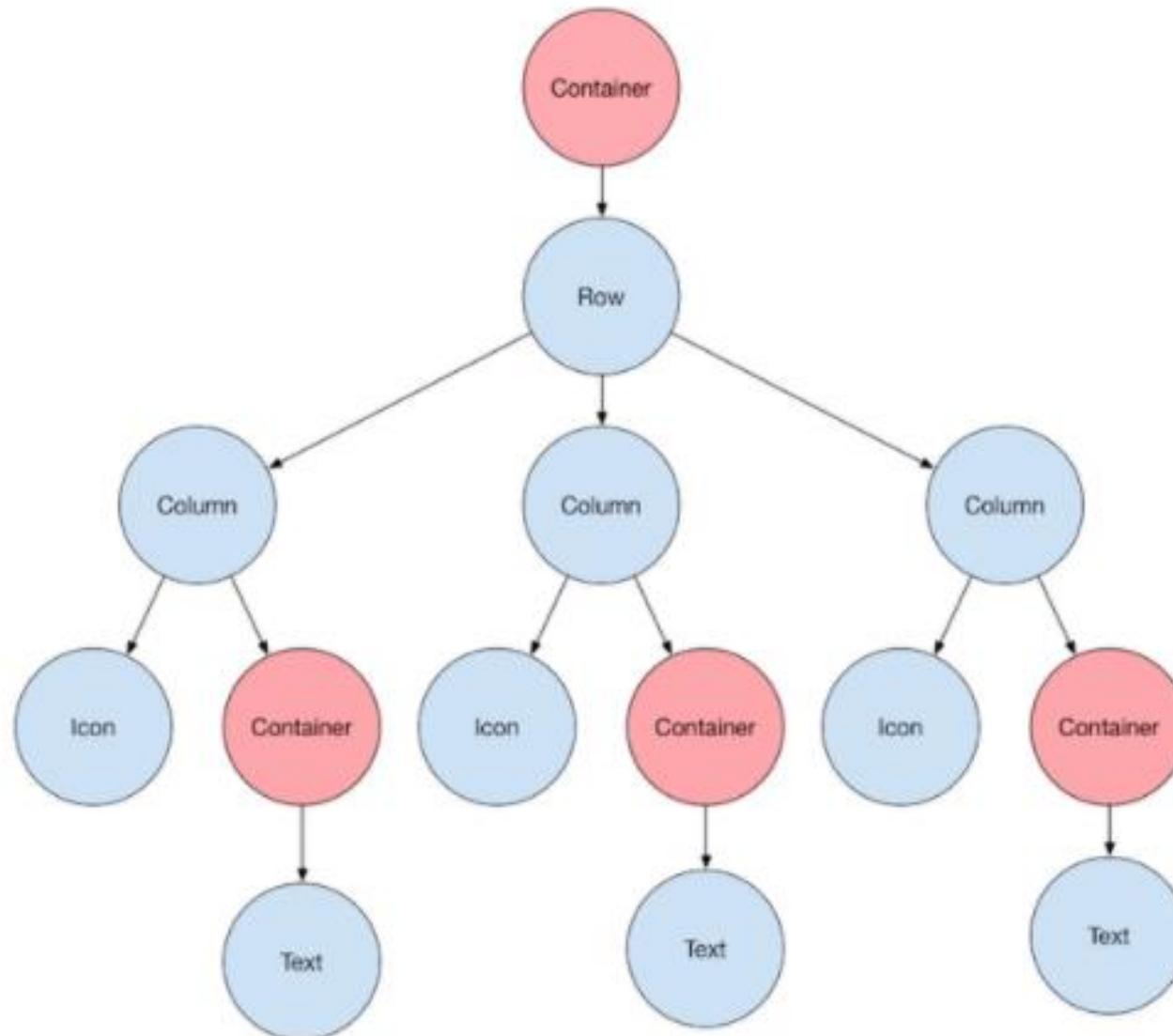
Flutter Docs Showcase Community

Single-child layout widgets

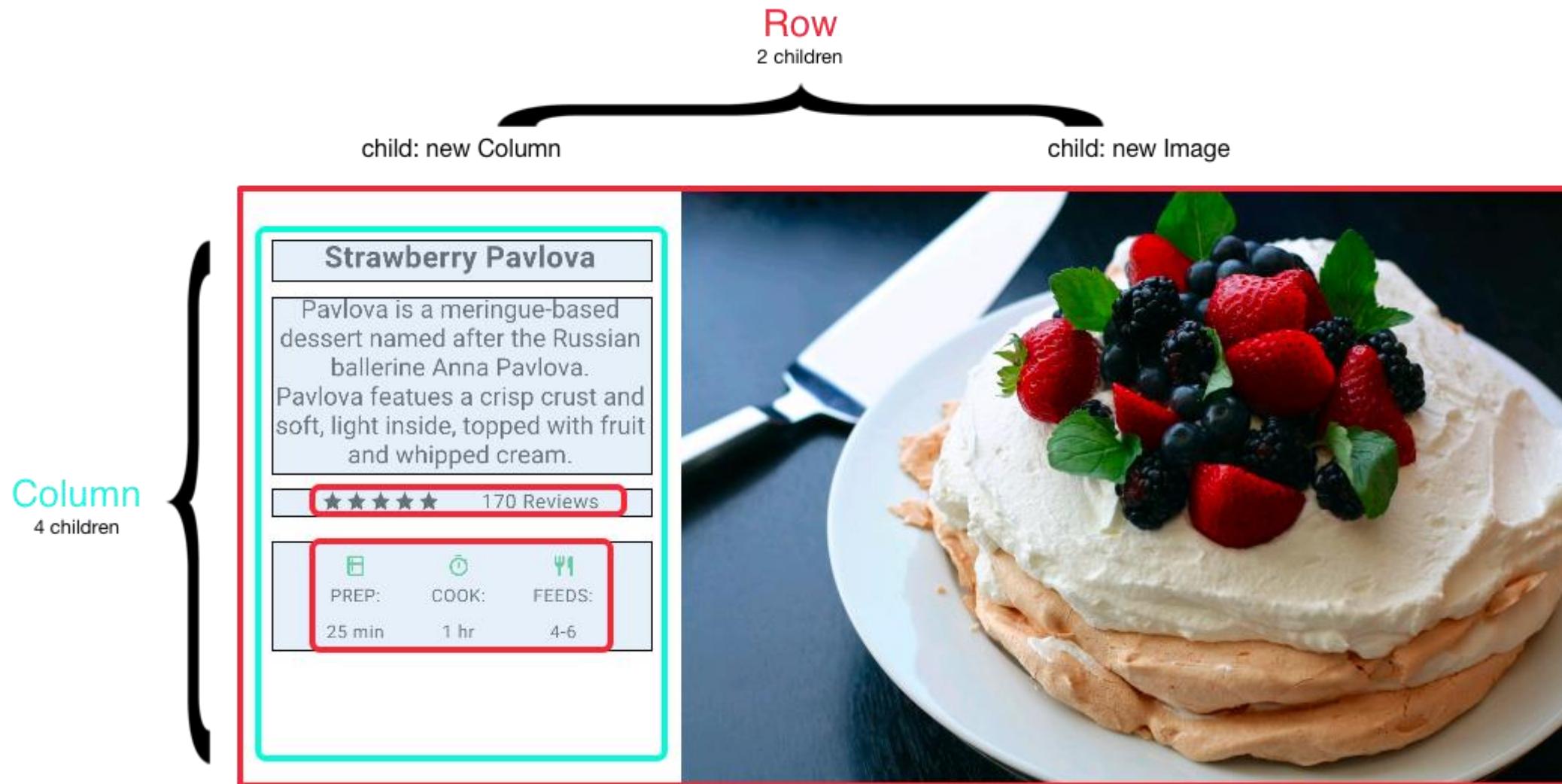
 <b>Container</b> A convenience widget that combines common painting,	 <b>Padding</b> A widget that insets its child by the given padding.	 <b>Center</b> A widget that centers its child within itself.
---	--	---

<https://flutter.dev/docs/development/ui/layout>

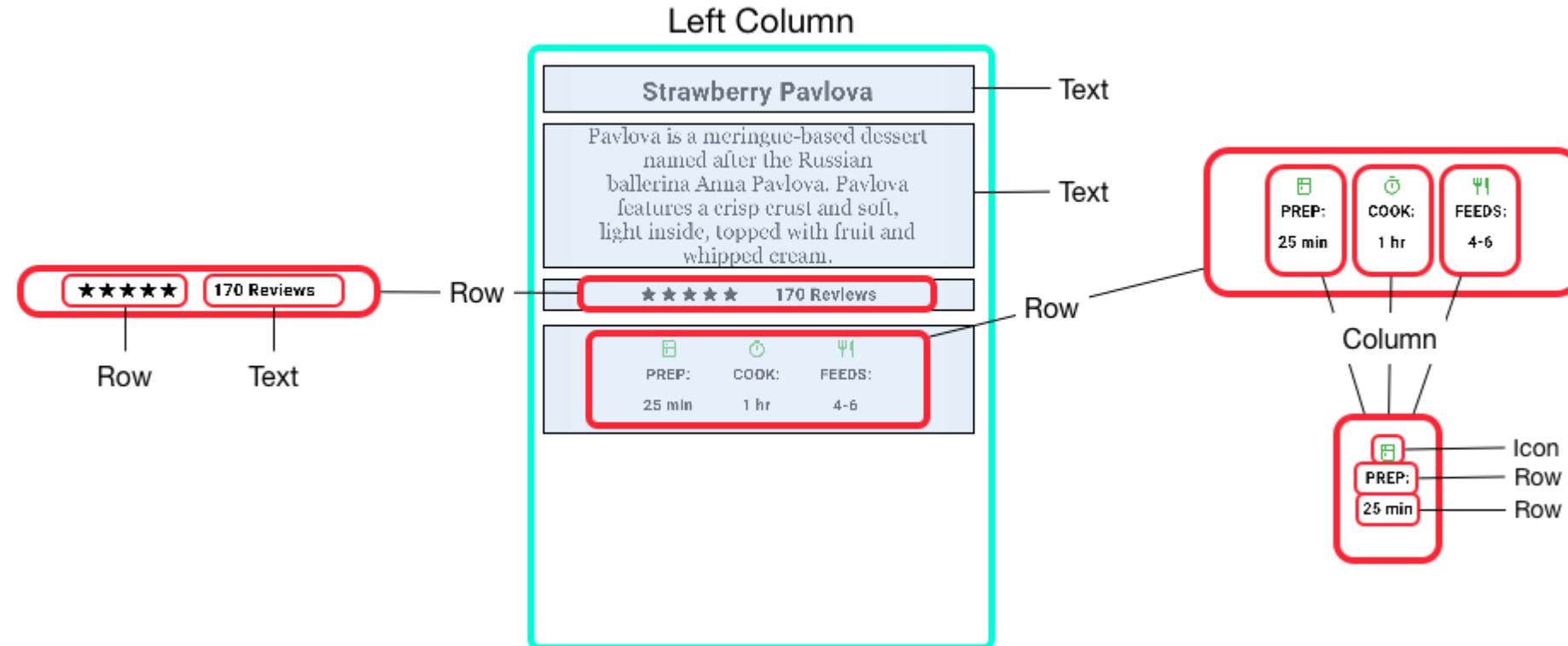
# Layouts in Flutter



# Layouts in Flutter Row & Column

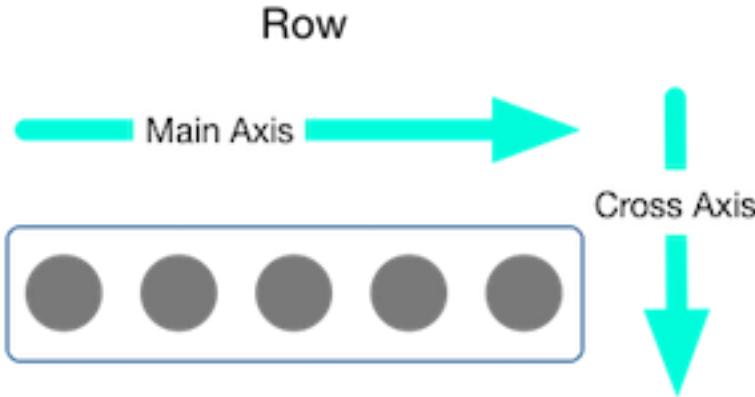


# Layouts in Flutter Row & Column



- จากภาพจะสังเกตว่า column และ row อาจจะซ่อนอยู่ข้างในของกันและกันได้ ให้จำง่ายๆ ว่าคือล้มน์เรียงบนลงล่าง ส่วน row เรียงจากซ้ายไปขวา

# การจัด Aligning Widget



```
Row(  
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
  children: [  
    Image.asset('images/pic1.jpg'),  
    Image.asset('images/pic2.jpg'),  
    Image.asset('images/pic3.jpg'),  
  ],  
);
```



App source: [row\\_column](#)

```
Column(  
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
  children: [  
    Image.asset('images/pic1.jpg'),  
    Image.asset('images/pic2.jpg'),  
    Image.asset('images/pic3.jpg'),  
  ],  
);
```

App source: [row\\_column](#)



# Layout



## MainAxisAlignment

การจัดในแกนหลัก เช่น ถ้าใช้ Row คือ การจัดในแนวอน, ถ้าใช้ Column คือ การจัดในแนวตั้ง

นอกจาก Main แล้วจะมี **CrossAxisAlignment** คือแนวอนตั้ง แกน X

# Example Row & MainAxisAlignment.start



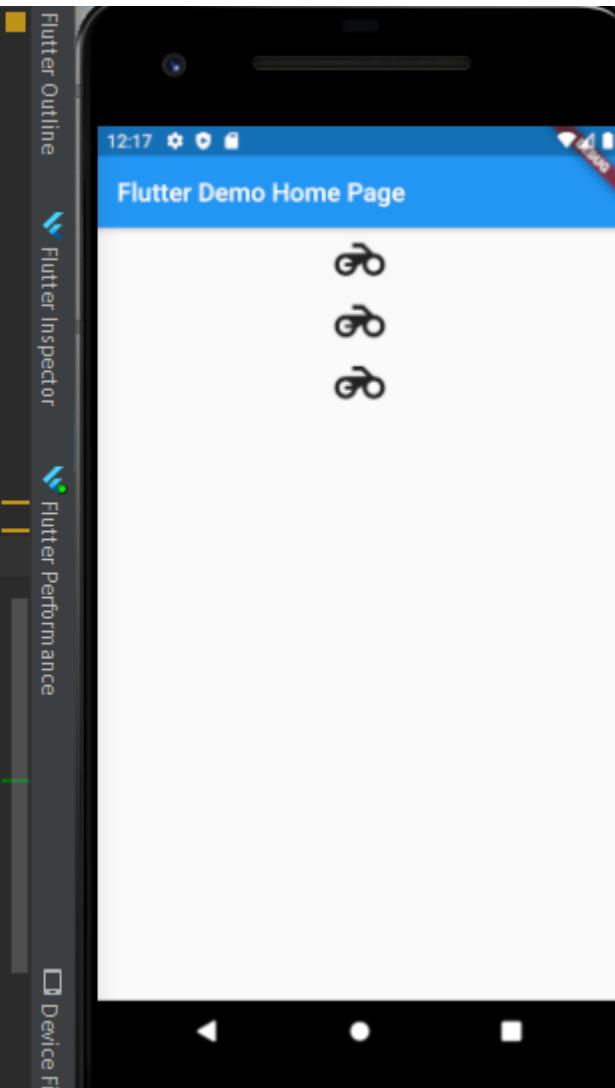
The screenshot shows an Android emulator displaying the 'Flutter Demo Home Page'. At the top, there is a blue header bar with the text 'Flutter Demo Home Page'. Below the header, there is a white content area containing a horizontal row of three motorcycle icons. Each icon is a black silhouette of a motorcycle, with a size of 48 pixels. The icons are evenly spaced along the horizontal axis.

```
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: Text(widget.title),
        ), // AppBar

        body: Center(
            child: Row(
                mainAxisAlignment: MainAxisAlignment.start,
                children: <Widget>[
                    Icon(Icons.motorcycle, size: 48),
                    Icon(Icons.motorcycle, size: 48),
                    Icon(Icons.motorcycle, size: 48),
                ], // <Widget>[]
            ), // Row
        ), // Center
    ); // Scaffold
}
```

# Example Column & MainAxisAlignment.start



```
}

@Override
Widget build(BuildContext context) {

    return Scaffold(
        appBar: AppBar(
            title: Text(widget.title),
        ), // AppBar

        body: Center(
            child: Column(
                mainAxisAlignment: MainAxisAlignment.start,
                children: <Widget>[
                    Icon(Icons.motorcycle, size: 48),
                    Icon(Icons.motorcycle, size: 48),
                    Icon(Icons.motorcycle, size: 48),
                ], // <Widget>[]
            ), // Column
        ),
    ), // Center
); // Scaffold
}
```

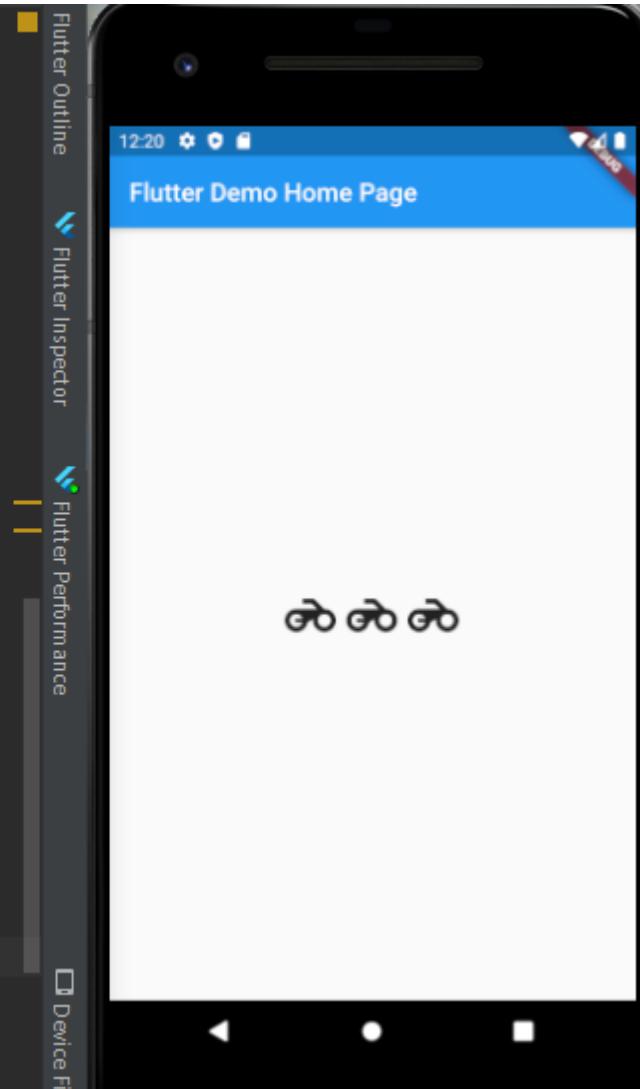
# Example Row & MainAxisAlignment.center

```
}

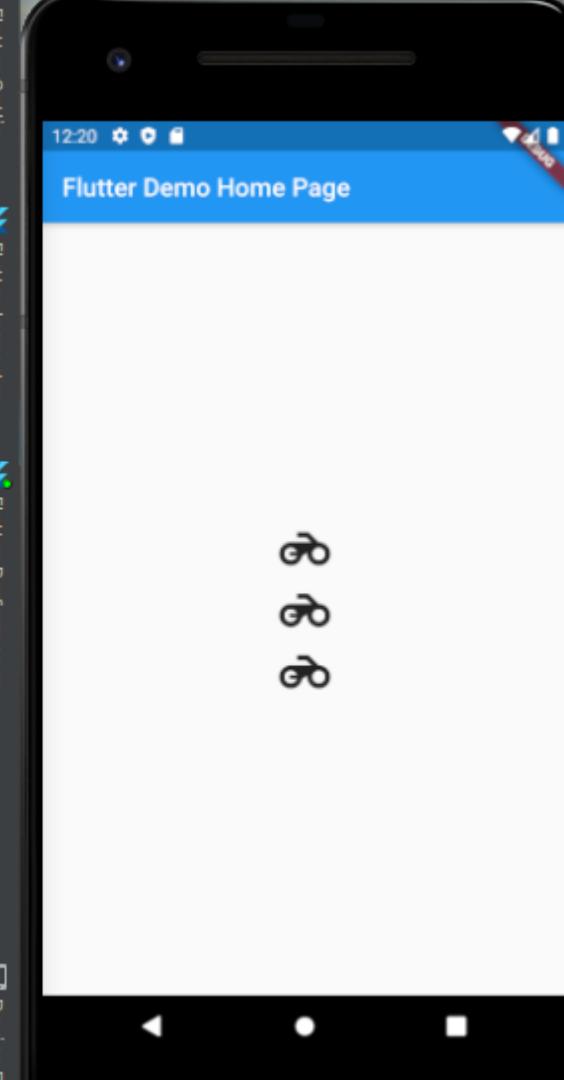
@Override
Widget build(BuildContext context) {

    return Scaffold(
        appBar: AppBar(
            title: Text(widget.title),
        ), // AppBar

        body: Center(
            child: Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                    Icon(Icons.motorcycle, size: 48),
                    Icon(Icons.motorcycle, size: 48),
                    Icon(Icons.motorcycle, size: 48),
                ], // <Widget>[]
            ), // Row
        ), // Center
    ); // Scaffold
}
```



# Example Column & MainAxisAlignment.center



The screenshot shows an Android emulator displaying the 'Flutter Demo Home Page'. The title bar reads 'Flutter Demo Home Page' and the time is 12:20. The main content area contains three motorcycle icons arranged vertically in the center. The Flutter DevTools interface is visible on the left, showing the Dart code for the application.

```
}

@Override
Widget build(BuildContext context) {

    return Scaffold(
        appBar: AppBar(
            title: Text(widget.title),
        ), // AppBar

        body: Center(
            child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                    Icon(Icons.motorcycle, size: 48),
                    Icon(Icons.motorcycle, size: 48),
                    Icon(Icons.motorcycle, size: 48),
                ], // <Widget>[]
            ), // Column
        ),
    ), // Center
}, // Scaffold
}
```

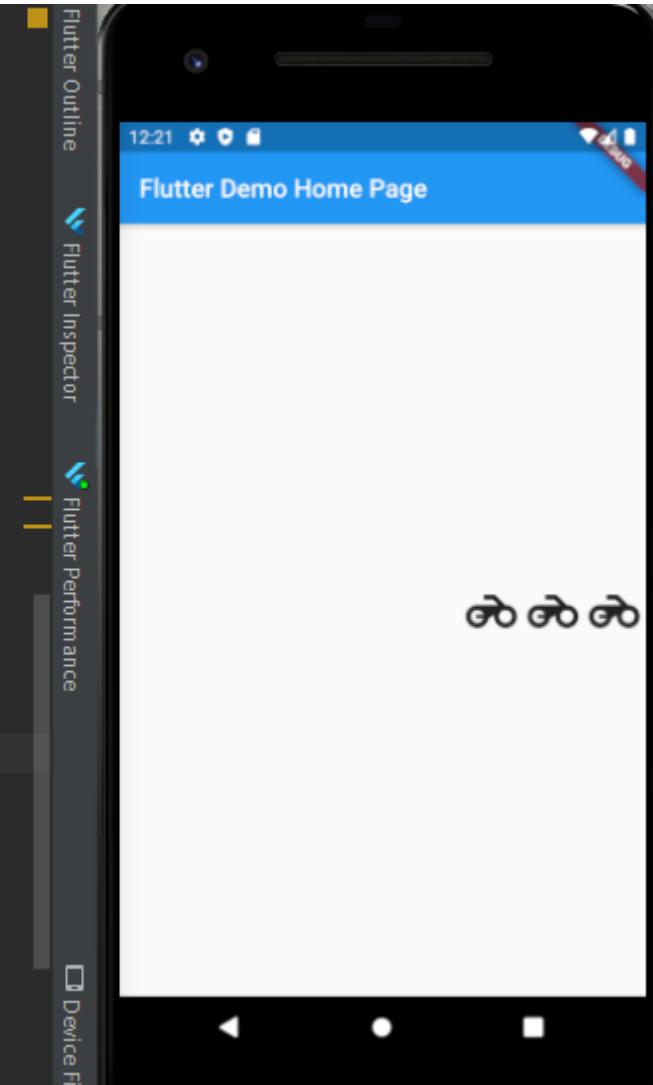
# Example Row & MainAxisAlignment.end

```
}

@Override
Widget build(BuildContext context) {

    return Scaffold(
        appBar: AppBar(
            title: Text(widget.title),
        ), // AppBar

        body: Center(
            child: Row(
                mainAxisAlignment: MainAxisAlignment.end,
                children: <Widget>[
                    Icon(Icons.motorcycle, size: 48),
                    Icon(Icons.motorcycle, size: 48),
                    Icon(Icons.motorcycle, size: 48),
                ], // <Widget>[]
            ), // Row
        ), // Center
    ); // Scaffold
}
```



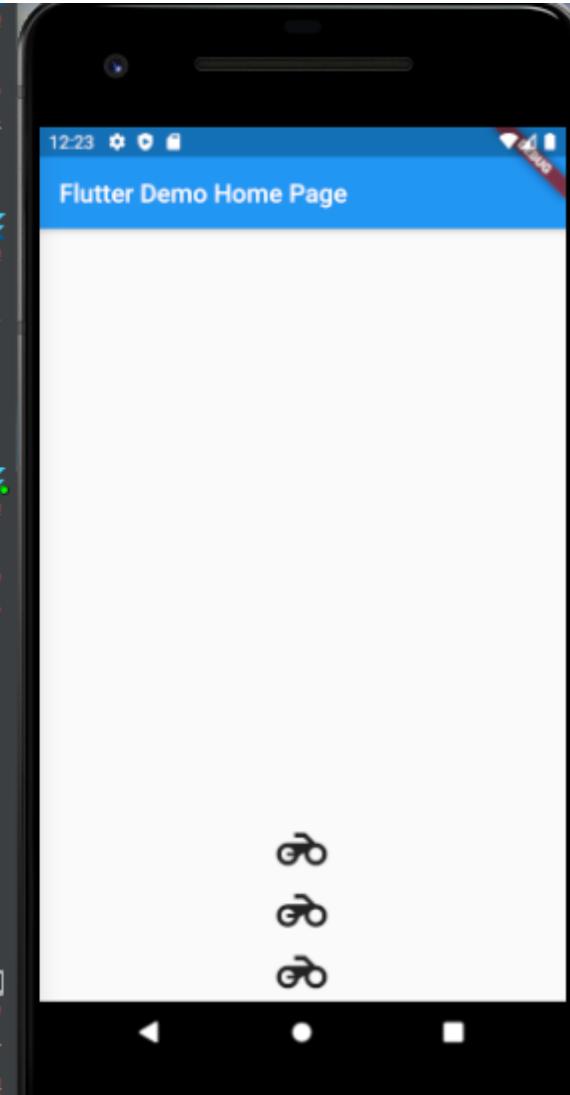
# Example Column & MainAxisAlignment.end

```
}

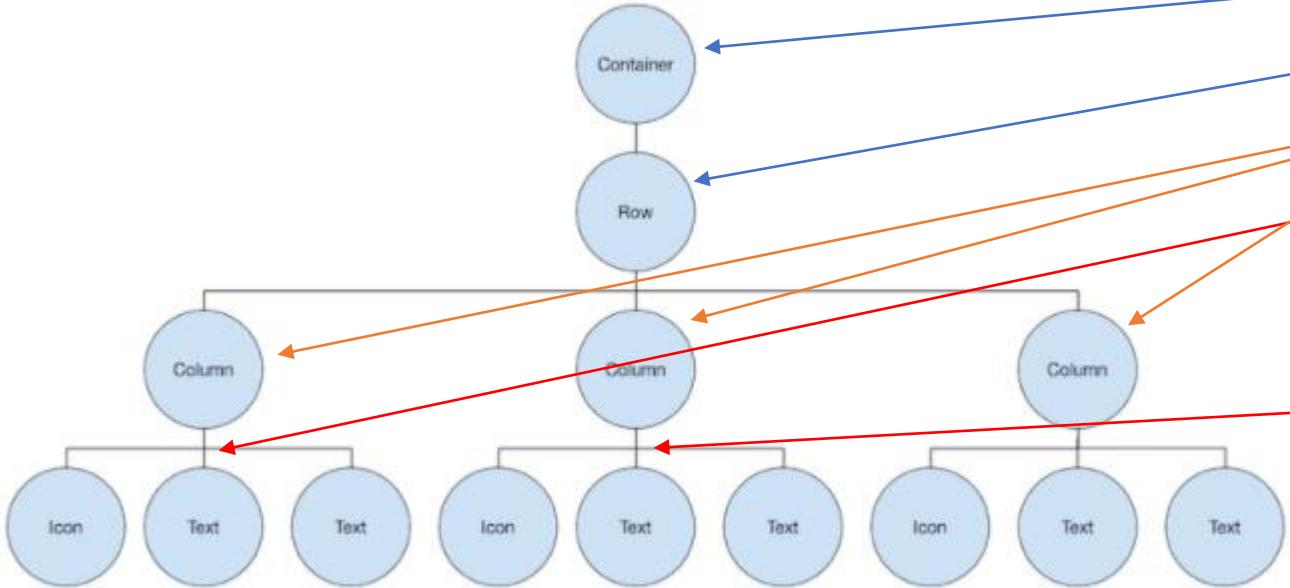
@Override
Widget build(BuildContext context) {

    return Scaffold(
        appBar: AppBar(
            title: Text(widget.title),
        ), // AppBar

        body: Center(
            child: Column(
                mainAxisAlignment: MainAxisAlignment.end,
                children: <Widget>[
                    Icon(Icons.motorcycle, size: 48),
                    Icon(Icons.motorcycle, size: 48),
                    Icon(Icons.motorcycle, size: 48),
                ], // <Widget>[]
            ), // Column
        ), // Center
    ); // Scaffold
}
```



## Child & Children



```
// DefaultTextStyle.merge() allows you to create a default text
// style that is inherited by its child and all subsequent children.
final iconList = DefaultTextStyle.merge(
  style: descTextStyle,
  child: Container(
    padding: EdgeInsets.all(20),
    child: Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Column(
          children: [
            Icon(Icons.kitchen, color: Colors.green[500]),
            Text('PREP:'),
            Text('25 min'),
          ],
        ),
        Column(
          children: [
            Icon(Icons.timer, color: Colors.green[500]),
            Text('COOK:'),
            Text('1 hr'),
          ],
        ),
        Column(
          children: [
            Icon(Icons.restaurant, color: Colors.green[500]),
            Text('FEEDS:'),
            Text('4-6'),
          ],
        ),
      ],
    ),
  );
);
```

Run Tools VCS Window Help flutter\_appweek2\_demo1

main.dart

No Devices



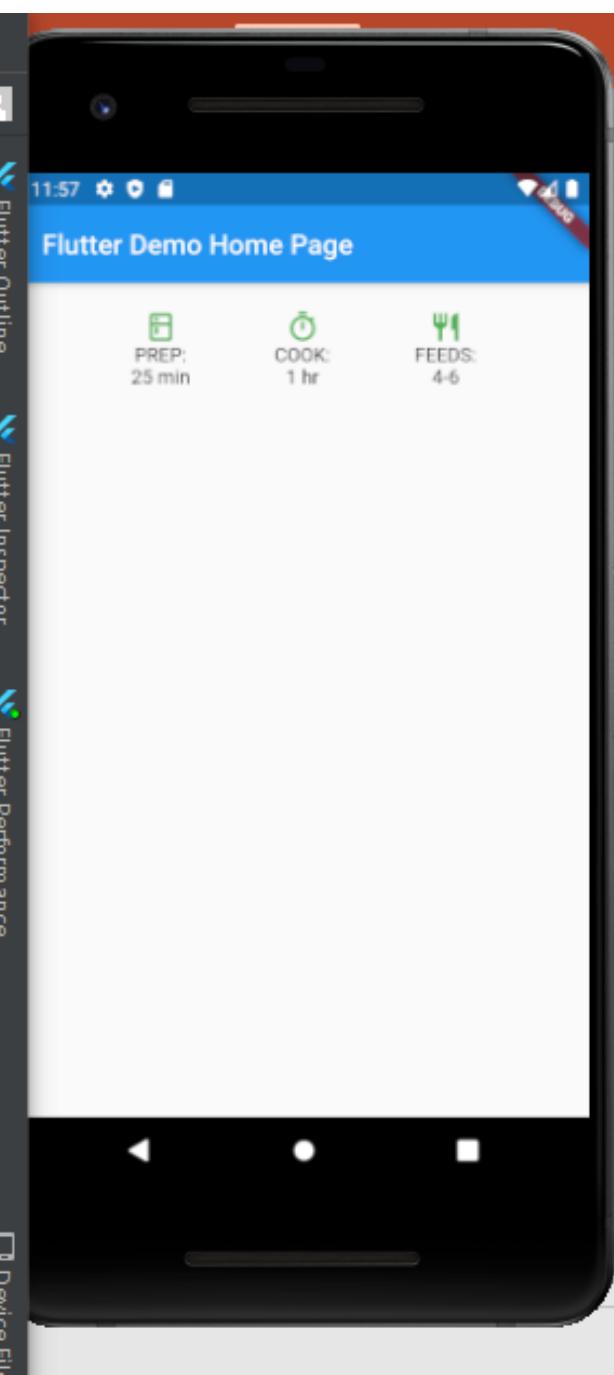
```
padding: EdgeInsets.all(20),
child: Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Column(
      children: [
        Icon(Icons.kitchen, color: Colors.green[500]),
        Text('PREP:'),
        Text('25 min'),
      ],
    ), // Column
    Column(
      children: [
        Icon(Icons.timer, color: Colors.green[500]),
        Text('COOK:'),
        Text('1 hr'),
      ],
    ), // Column
    Column(
      children: [
        Icon(Icons.restaurant, color: Colors.green[500]),
        Text('FEEDS:'),
        Text('4-6'),
      ],
    ), // Column
  ],
), // Row
```

Flutter Outline

Flutter Inspector

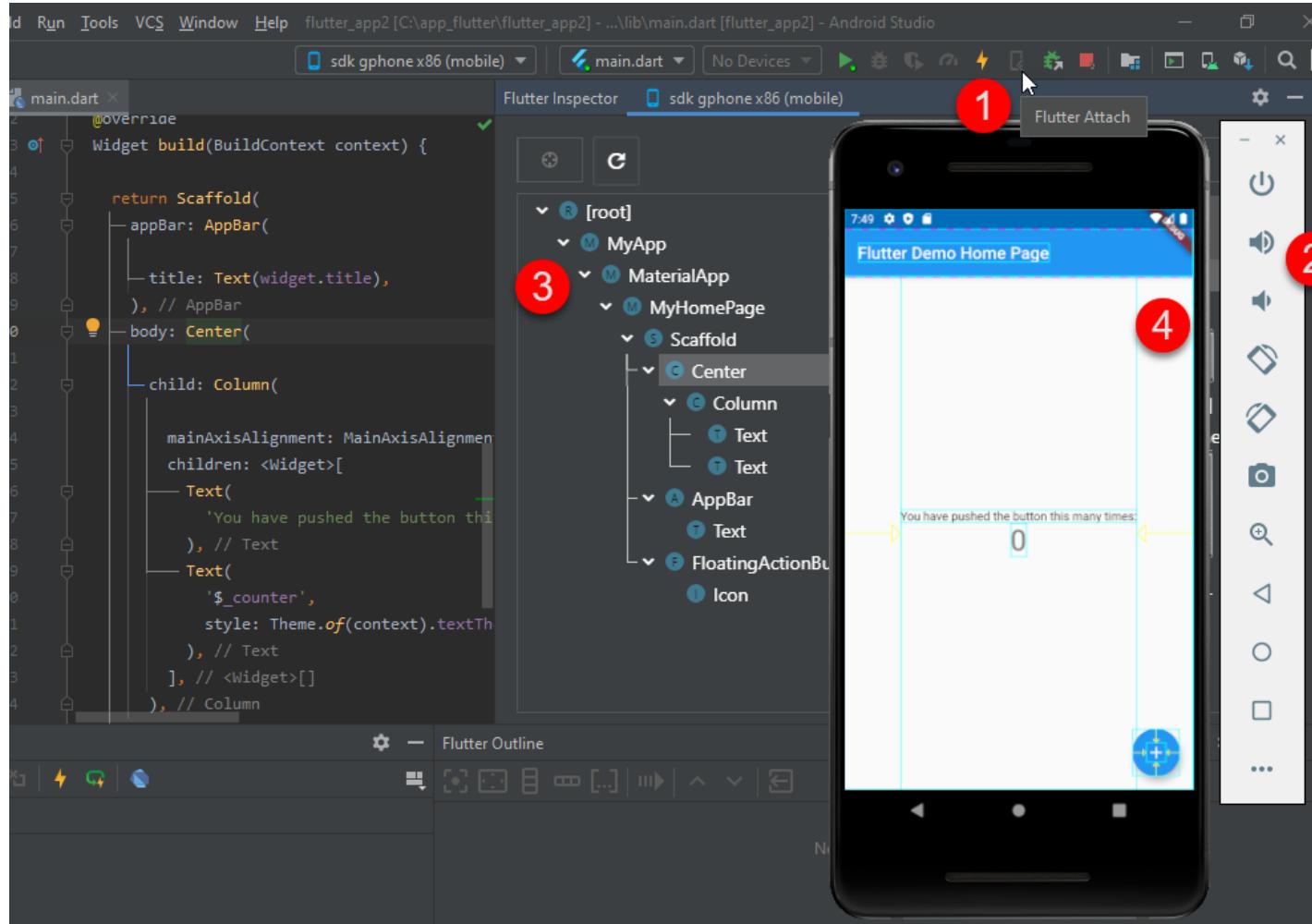
Flutter Performance

Device File



# Flutter บน Android Studio จะมี Tool

- กด Flutter Attach ด้านบนตอน run และมากด Flutter Inspector แล้วมาเลื่อนดู Layout ได้ดังภาพ

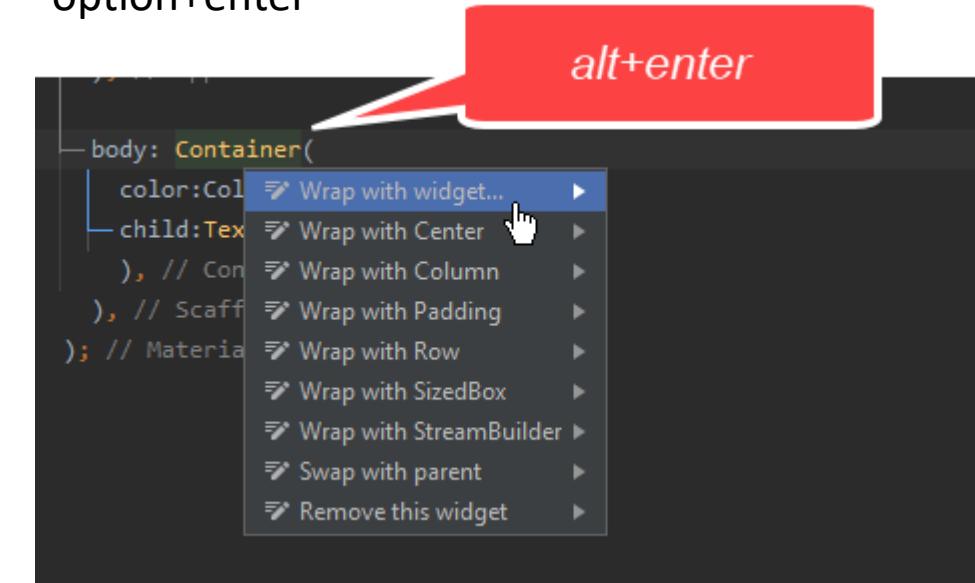


# Home Layout

The screenshot shows the main.dart file in a code editor. The code defines a main function that runs an MyApp widget. The MyApp widget is a StatelessWidget that returns a MaterialApp. The MaterialApp's home is a Scaffold. The Scaffold has an AppBar with the title 'แอปของฉัน'. The body of the Scaffold is a Container with a red color and a child Text('Hello'). The preview window shows a black iPhone X with a blue status bar displaying 'แอปของฉัน' and a white screen with the text 'Hello'.

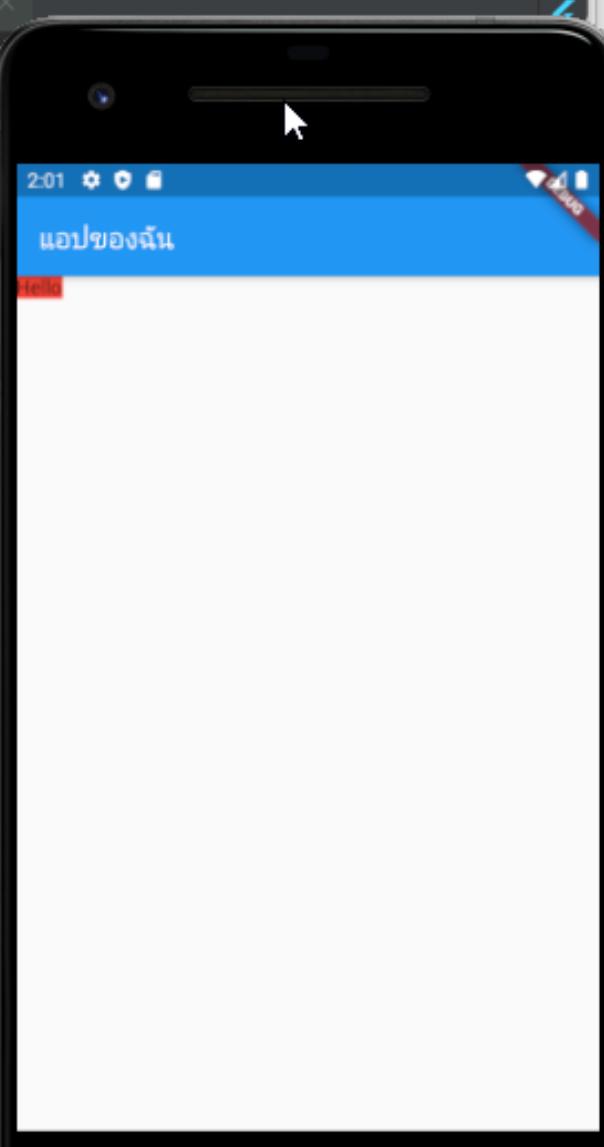
```
2
3     void main() {
4         runApp(MyApp());
5     }
6
7     class MyApp extends StatelessWidget {
8         Widget build(BuildContext context){
9             return MaterialApp(
10                 home: Scaffold(
11                     appBar: AppBar(
12                         title: Text('แอปของฉัน'),
13                     ), // AppBar
14
15                     body: Container(
16                         color:Colors.red,
17                         child:Text('Hello'),
18                     ), // Container
19                 ), // Scaffold
20             ); // MaterialApp
21
22         }
23     }
24 }
```

ถ้าได้โค้ดตามภาพจะสังเกตุว่า ข้อความ Hello จะชิดกับ appBar เราชานารถกำหนด Layout ได้โดยในส่วนของ body:Container ให้กด alt+enter หรือ option+enter

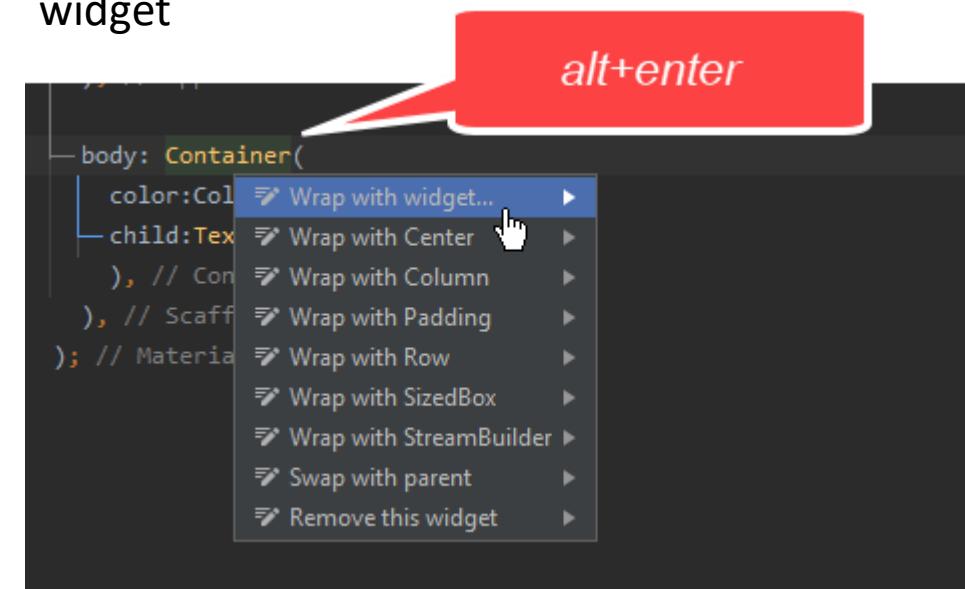


# Home Layout

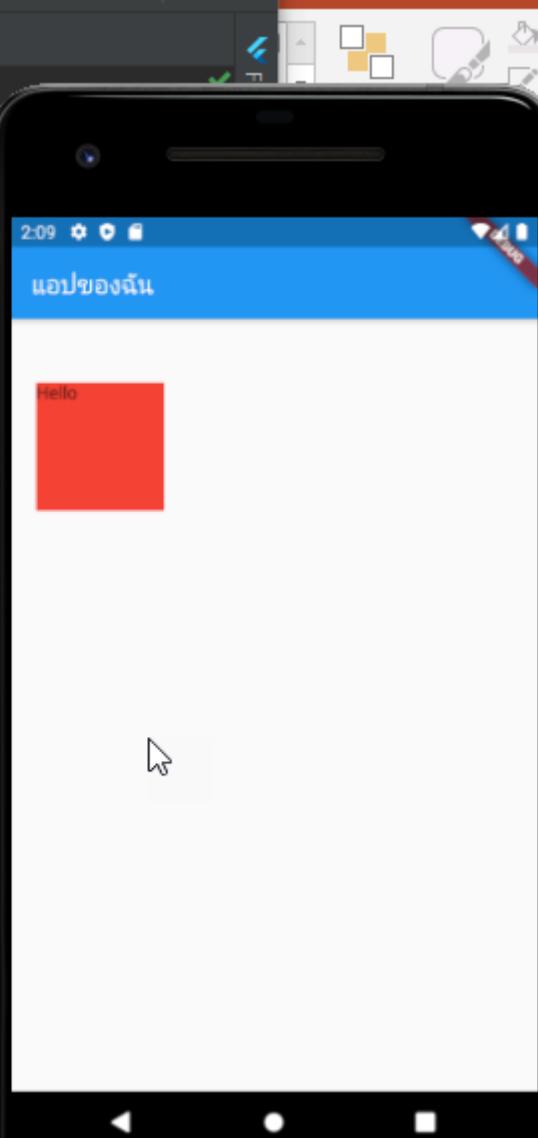
```
2
3     void main() {
4         runApp(MyApp());
5     }
6
7     class MyApp extends StatelessWidget {
8         Widget build(BuildContext context){
9             return MaterialApp(
10                 home: Scaffold(
11                     appBar: AppBar(
12                         title: Text('แอปของฉัน'),
13                     ), // AppBar
14
15                     body: Container(
16                         color:Colors.red,
17                         child:Text('Hello'),
18                     ), // Container
19                 ), // Scaffold
20             ); // MaterialApp
21
22         }
23     }
24 }
```



ถ้าได้โค้ดตามภาพจะสังเกตุว่า ข้อความ Hello จะชิดกับ appBar เราสามารถกำหนด Layout ได้โดยในส่วนของ body:Container ให้กด alt+enter หรือ option+enter เลือกอันแรก wrap width widget



# Home Layout



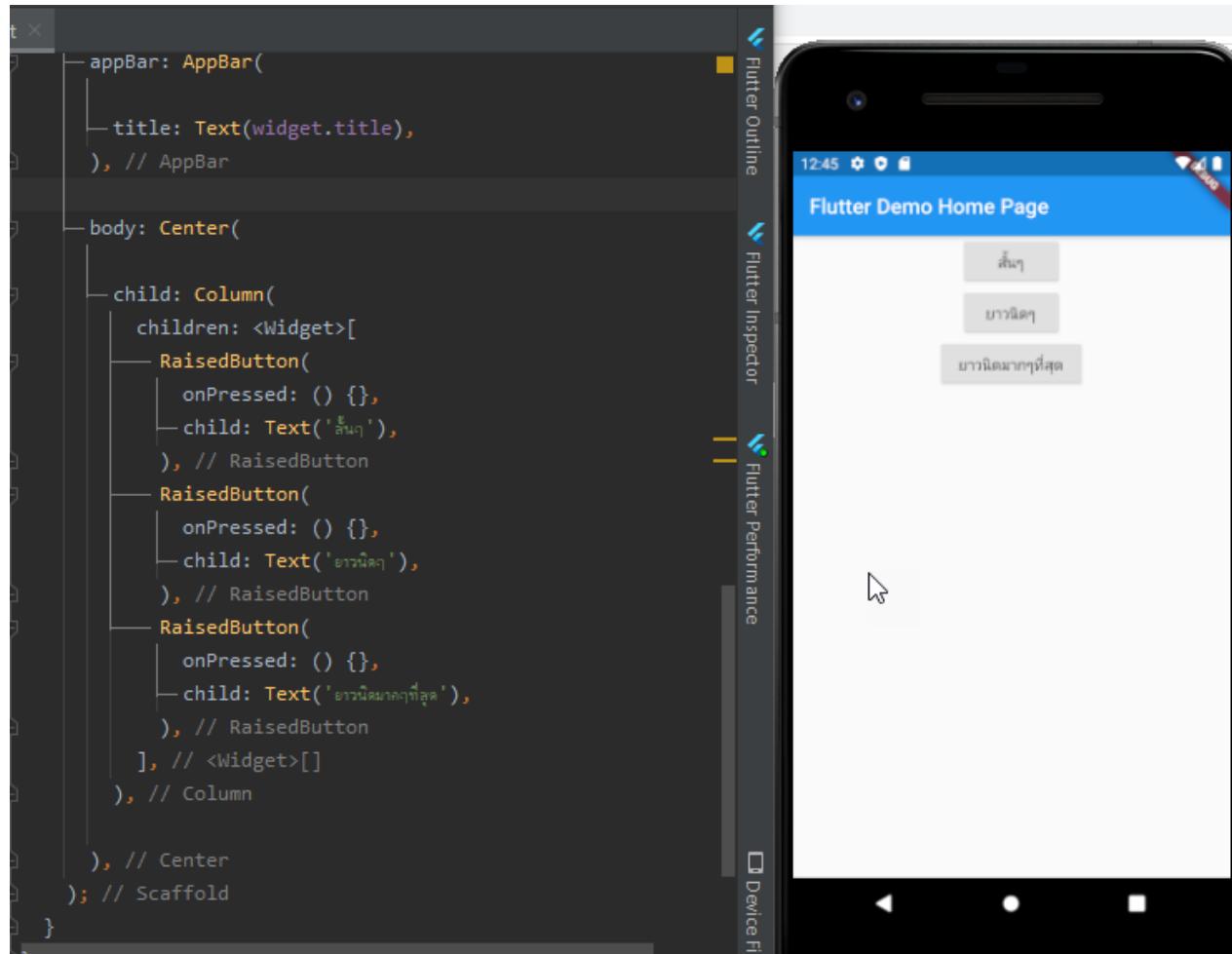
The screenshot shows an Android emulator displaying a simple application. The title bar reads "แอปของฉัน". The main screen contains a single red square with the text "Hello" centered inside it.

```
2 void main() {  
3   runApp(MyApp());  
4 }  
  
5 class MyApp extends StatelessWidget {  
6   Widget build(BuildContext context){  
7     return MaterialApp(  
8       home: Scaffold(  
9         appBar: AppBar(  
10          title: Text('แอปของฉัน'),  
11        ), // AppBar  
12  
13        body: SafeArea(  
14          child: Container(  
15            height:100.0,  
16            width:100.0,  
17            margin: EdgeInsets.fromLTRB(20, 50, 10, 20),  
18            color: Colors.red,  
19            child: Text('Hello'),  
20            ), // Container  
21          ), // SafeArea  
22        ), // Scaffold  
23      ); // MaterialApp  
24  
25    }  
26  
27 }  
28 }
```

หลังจากนั้นให้แก้ widget เป็น **SafeArea** เพื่อให้มี margins และ padding ที่กำหนด ความกว้าง ความสูง และ margin ของ element ได้ตามใจคุณ

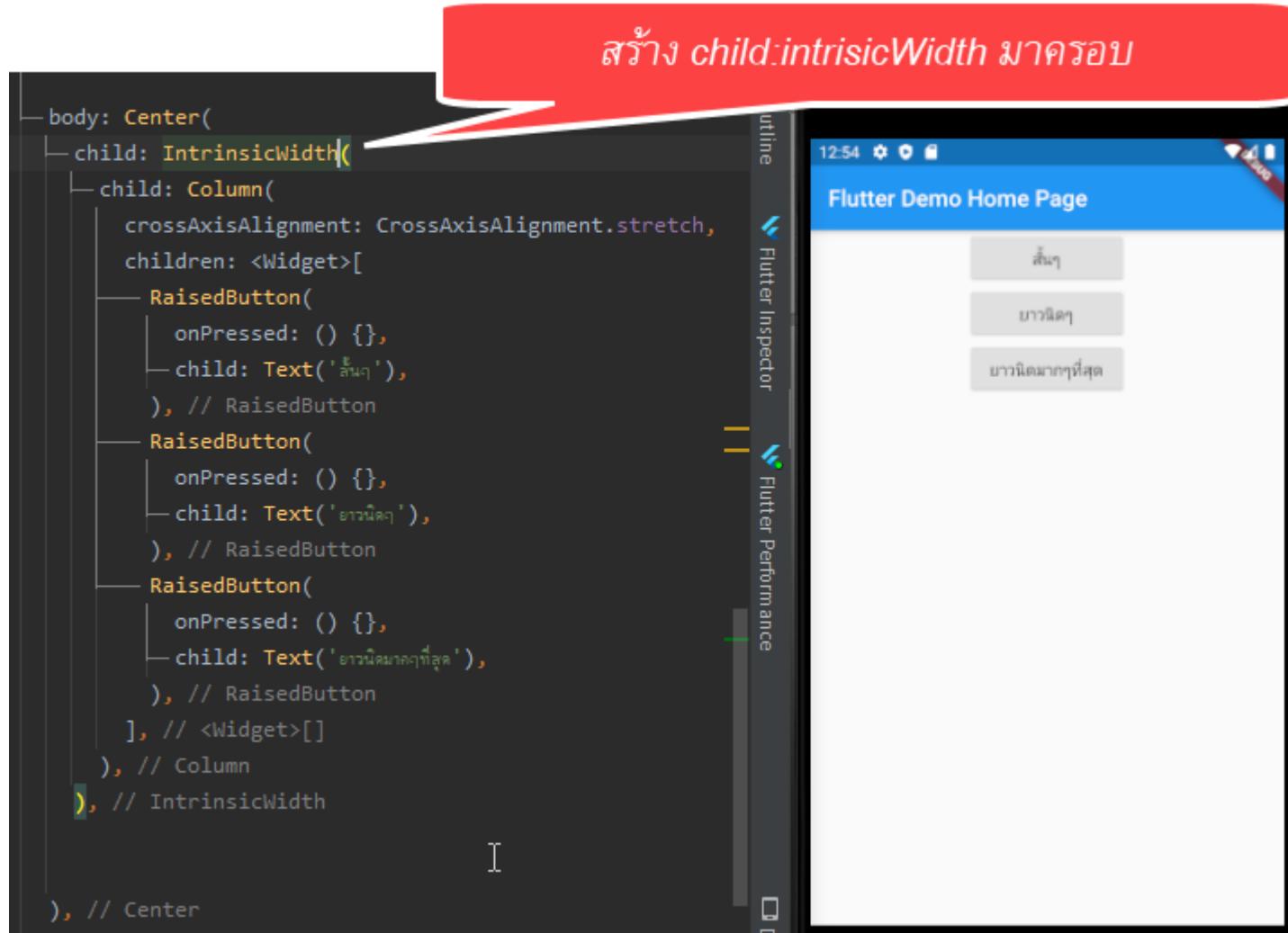
# IntrinsicWidth and IntrinsicHeight

- Intrinsic แปลว่า แท้จริง, ที่อยู่ภายใน ในกรณีที่ต้องการให้ความกว้าง หรือความสูง นั้น กว้างเท่ากับ widget ที่กว้างที่สุด ใน Column หรือสูงเท่ากับ widget ที่สูงที่สุดใน Row เราไม่ต้องยุ่งยากไปใช้วิธีอื่นๆ โดยทั่วไปถ้าเราไม่ใช้ Intrinsic จะเป็นดังภาพด้านล่าง



# IntrinsicWidth and IntrinsicHeight

- พอยใช้ IntrinsicWidth จะได้แบบภาพด้านล่าง



# IntrinsicHeight

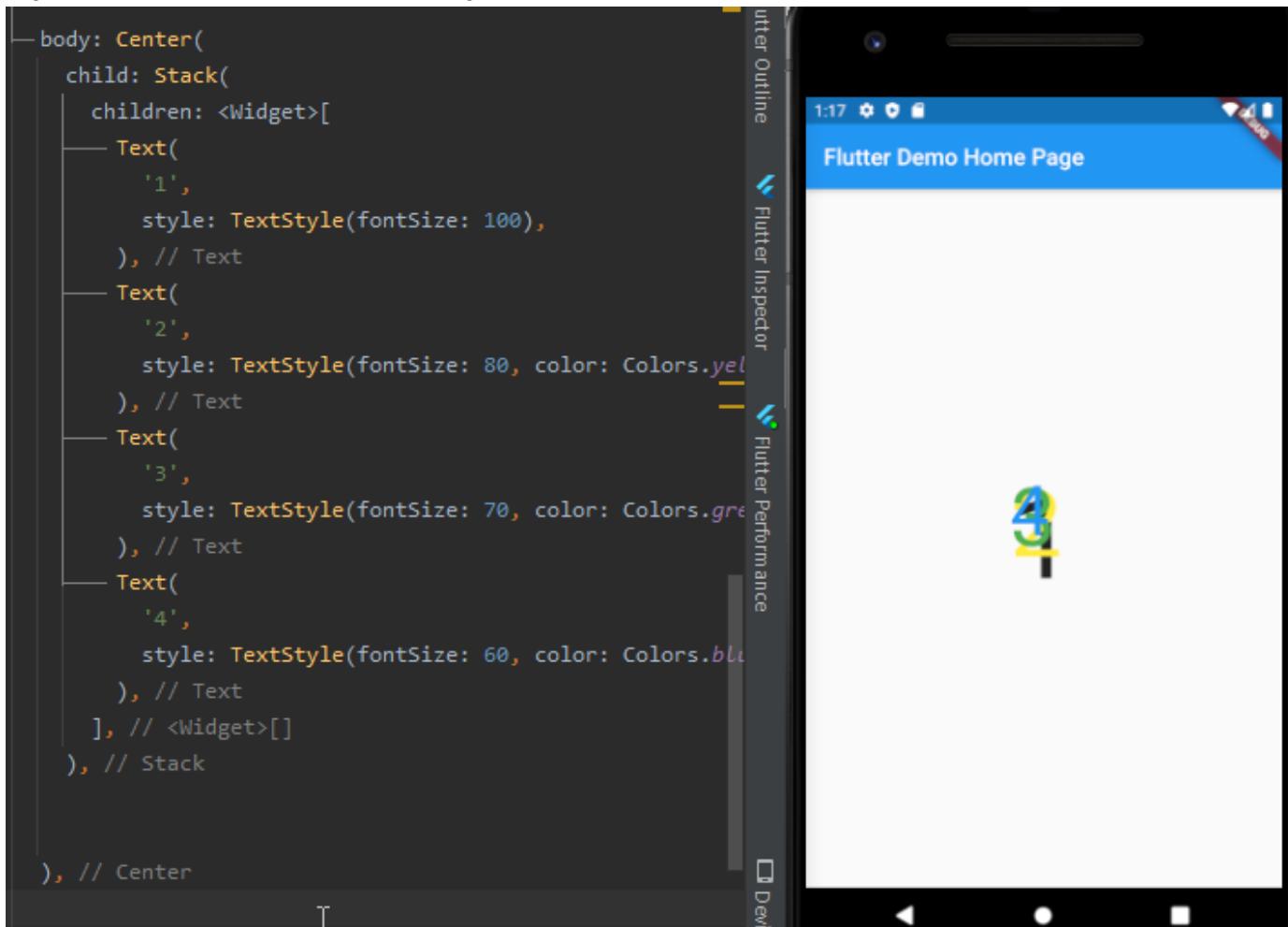


The screenshot shows an Android emulator displaying a Flutter application titled "Flutter Demo Home Page". The screen contains three raised buttons arranged horizontally. The first button has the text "ສັນຕິພາບ", the second has "ຍາວເລີດ", and the third has "ຍາວເລີດມາກຸງທີ່ສຸດ". The Flutter Inspector panel on the right side of the IDE interface is visible, showing details about the current widget tree.

```
body: Center(
  child: IntrinsicHeight(
    child: Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        RaisedButton(
          onPressed: () {},
          child: Text('ສັນຕິພາບ'),
        ), // RaisedButton
        RaisedButton(
          onPressed: () {},
          child: Text('ຍາວເລີດ'),
        ), // RaisedButton
        RaisedButton(
          onPressed: () {},
          child: Text('ຍາວເລີດມາກຸງທີ່ສຸດ'),
        ), // RaisedButton
      ], // <Widget>[]
    ), // Row
  ), // IntrinsicHeight
), // Center
```

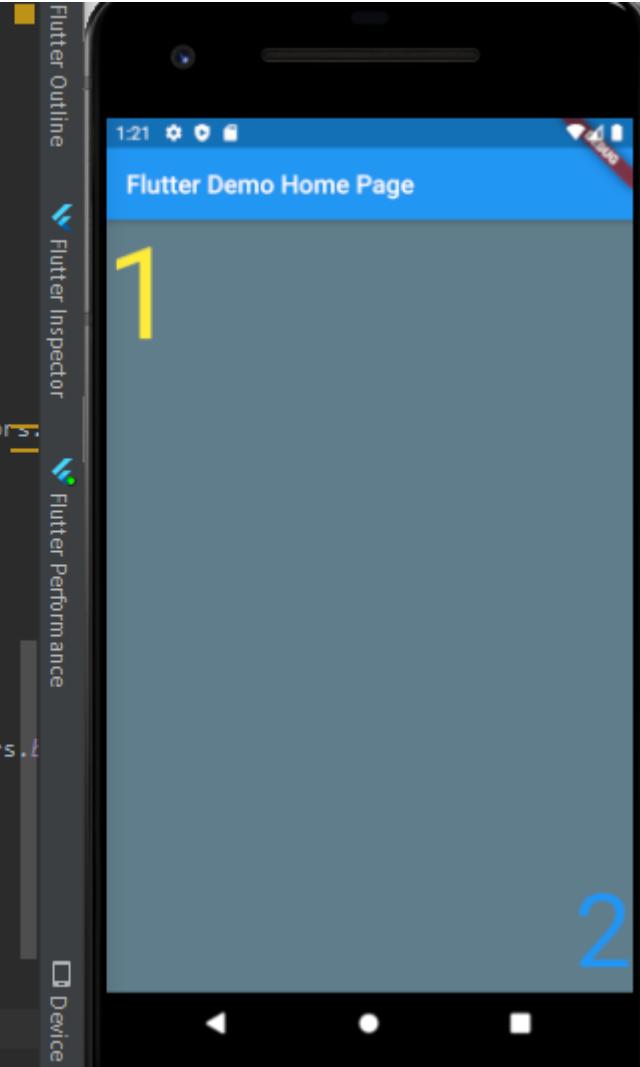
# Stack

- ใช้สำหรับการวาง layout แบบว่าง widget ซ้อนทับกันนั่นเอง โดยใช้ List<Widget> widget ที่อยู่อันดับแรกสุดคืออยู่ล่างสุด widget ที่อยู่อันดับสุดท้ายคือตัวที่อยู่บนสุด



# Stack

- สำหรับกรณีที่ต้องการจัดตำแหน่งสามารถใช้ Positioned ในการวางแผนตามที่ต้องการได้เลย โดยพื้นที่วางนั้นขึ้นอยู่กับขนาดของ widget แม่



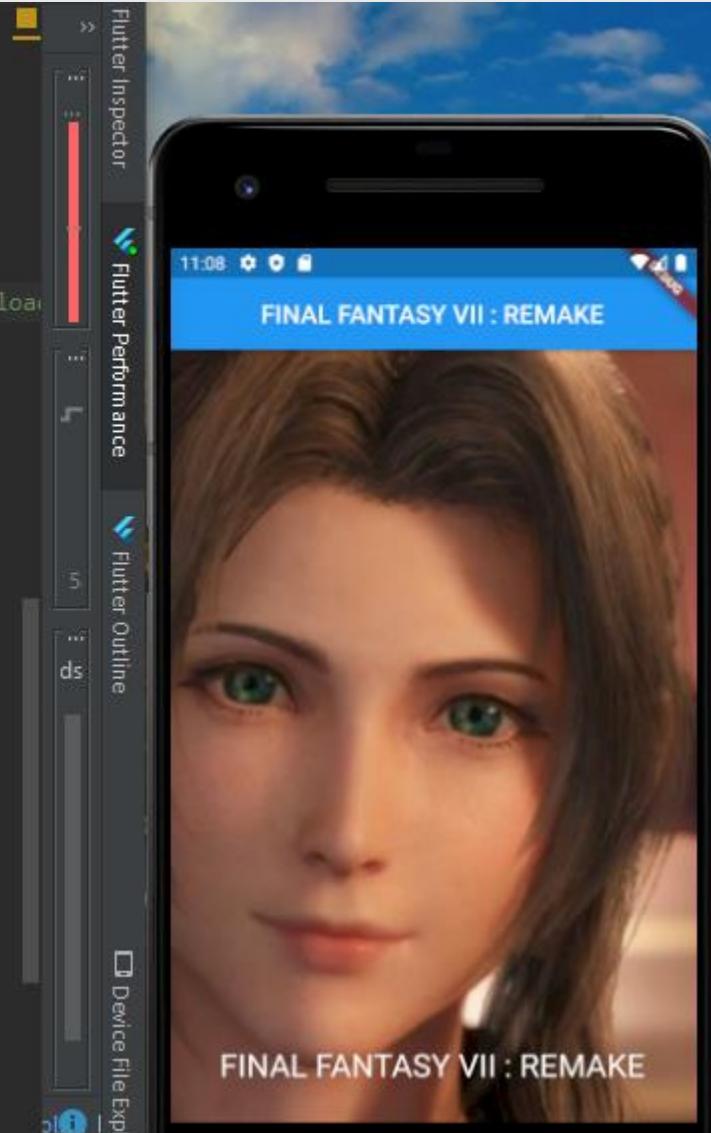
The screenshot shows the Flutter Dev Tools interface. On the left, there's a code editor with the following Dart code:

```
body: Center(
  child: Stack(
    fit: StackFit.expand,
    children: <Widget>[
      Material(color: Colors.blueGrey),
      Positioned(
        top: 0,
        left: 0,
        child: Text(
          '1',
          style: TextStyle(fontSize: 100, color: Colors.white),
        ), // Text
      ), // Positioned
      Positioned(
        bottom: 0,
        right: 0,
        child: Text(
          '2',
          style: TextStyle(fontSize: 80, color: Colors.brown),
        ), // Text
      ), // Positioned
    ], // <Widget>[]
), // Stack
), // Center
```

On the right, an iPhone X simulator displays the resulting UI. The screen has a blue background. In the top-left corner, there is a large yellow '1'. In the bottom-right corner, there is a smaller blue '2'. The text is rendered in a large font size, demonstrating the use of the Positioned widget to place the text at specific coordinates relative to the stack's boundaries.

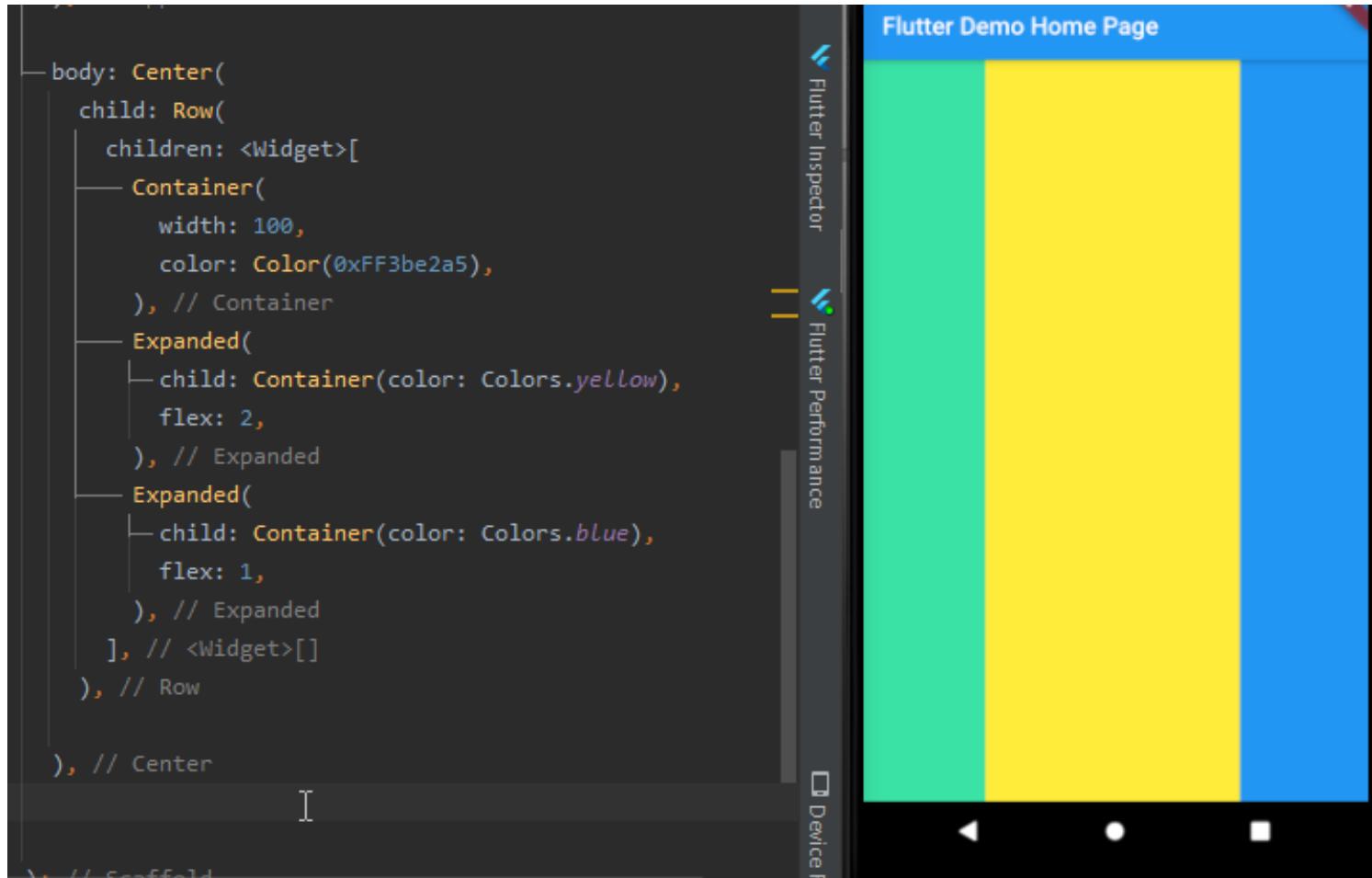
# Stack เօຮງປກາວ ຈອນ Image ໄດ້

```
body: SafeArea(  
  child: Stack(  
    children: <Widget>[  
      // Background Image  
      Container(  
        decoration: BoxDecoration(  
          image: DecorationImage(  
            image: NetworkImage('https://static3.cbrimages.com/wordpress/wp-content/uploads/2019/05/01153333/Clara-De-Clare-From-Final-Fantasy-VII-Remake-Portrait-Image.jpg'),  
            fit: BoxFit.cover,  
          ), // DecorationImage  
        ), // BoxDecoration  
        child: null), // Container  
      Center(  
        child: Column(  
          mainAxisAlignment: MainAxisAlignment.end,  
          children: <Widget>[  
            Padding(  
              padding: const EdgeInsets.only(bottom: 30),  
              child: Text(  
                'FINAL FANTASY VII : REMAKE',  
                style: TextStyle(  
                  color: Colors.white,  
                  fontSize: 25,  
                ), // TextStyle  
              ), // Text  
            ], // <Widget>[]  
        ), // Column
```



# Expanded

- Expanded จะใช้สำหรับกรณีที่ต้องการที่จะใช้พื้นที่ที่เหลือทั้งหมดของ widget แม้ หาก widget แม่นั้นมี Expanded มากกว่า 1 ตัวจะต้องทำการใส่สัดส่วนให้มันผ่านค่า flex เพิ่มเติมเข้าไปด้วย



# Widget คืออะไร

ใน Flutter จะมองทุกอย่างเกือบทั้งหมดเป็น widget

Widget คือ ส่วนที่ถูกใช้สร้างเป็นหน้าตาของ App หรือที่เรียกว่า user interface (UI) โดยนำมาประกอบเรียงกันเป็นลำดับขึ้นชั้นเป็นโครงสร้าง แต่ละ widget จะถูกวางซ้อนอยู่ภายใน Parent widget และได้รับการส่งต่อสีบทอดคุณสมบัติ ต่างๆ จาก Parent อีกที แม้กระทั้ง application object ก็ถือเป็น widget ซึ่งเราเรียกว่า root widget MaterialApp คือ root widget

เราอาจจำแนก Widget ตามการใช้งาน ได้เป็น ดังนี้

- ใช้กำหนดโครงสร้าง (Structural Element) เช่น ปุ่ม button หรือ menu
- ใช้กำหนดลักษณะ หรือรูปแบบ (Stylistic Element) เช่น font หรือ color
- ใช้จัดวาง และกำหนดมุมมองเล้อท์ (Aspect of Layout) เช่น padding หรือ alignment



# StatelessWidget และ StatefulWidget คืออะไร

ใน App ของเราจะมี widget อยู่ 2 ประเภทหลัก ที่ใช้งานคือ stateless และ stateful widget โดย state ก็คือสภาพของสิ่งนั้นๆ stateless จึงหมายถึง widget ที่ไม่มี state หรือไม่มีสภาพการเปลี่ยนแปลง หรือไม่จำเป็นต้องใช้งานการเปลี่ยนแปลง จึงใช้งาน widget นี้ ส่วน stateful หมายถึง widget ที่มี state หรือมีสภาพการเปลี่ยนแปลง ไปตามข้อมูลที่ได้รับหรือจากการกำหนดจากผู้ใช้ข้อแตกต่างที่สำคัญของห้องสองส่วนนี้คือ stateful widget จะมี State object ที่ใช้ในการเก็บข้อมูล state และทำการส่งต่อสำหรับใช้งานในกระบวนการสร้าง widget ใหม่เมื่อมีการเปลี่ยนแปลง ทำให้ค่า state ไม่ได้หายไปไหน

## การใช้งาน StatelessWidget

Stateless widget ใน Flutter เป็น widget ที่ไม่จำเป็นที่ต้องมีการเปลี่ยนแปลง state เกิดขึ้น โดยเราจะใช้ StatelessWidget สำหรับสร้าง widget แบบคงที่ เหมาะสำหรับใช้ในการสร้าง และกำหนดส่วนของ UI ซึ่งจะปรับแต่งเฉพาะค่าข้อมูลของ ตัว widget เท่านั้น เช่น Text widget ก็ถือเป็น StatelessWidget ที่เป็น subclass ของ StatelessWidget

# StatelessWidget ตามโค้ดด้านล่าง

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

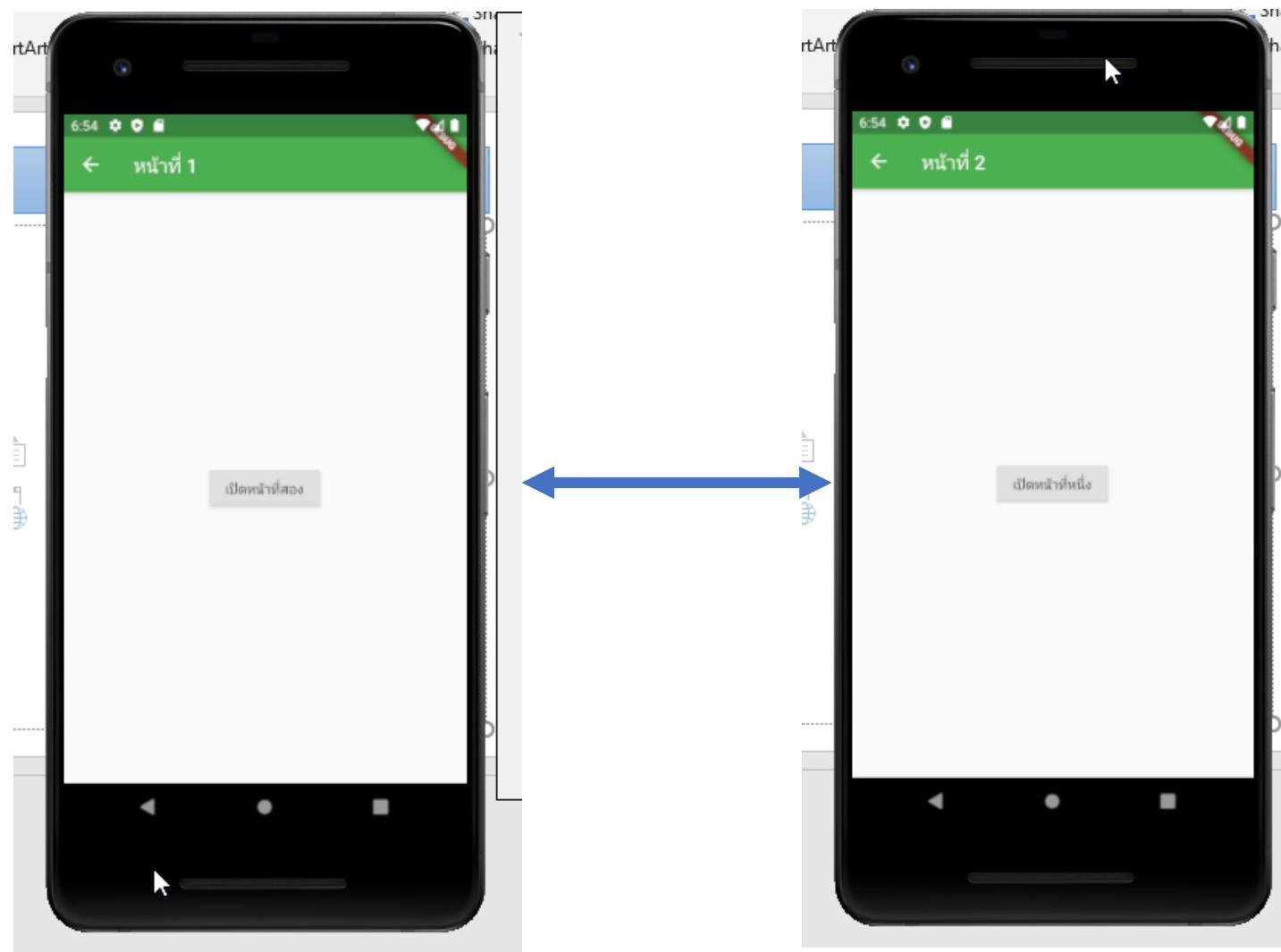
void main(){runApp(
    My StatelessWidget(text: ' StatelessWidget Example to show immutable data')
);
}

class StatelessWidget extends StatelessWidget {
    final String text;
    // constructor
    StatelessWidget({Key key, this.text}) : super(key: key);

    @override
    Widget build(BuildContext context) {
        return Center(
            child: Text(
                text,
                textDirection: TextDirection.ltr,
            ), // Text
        ); // Center
    }
}
```



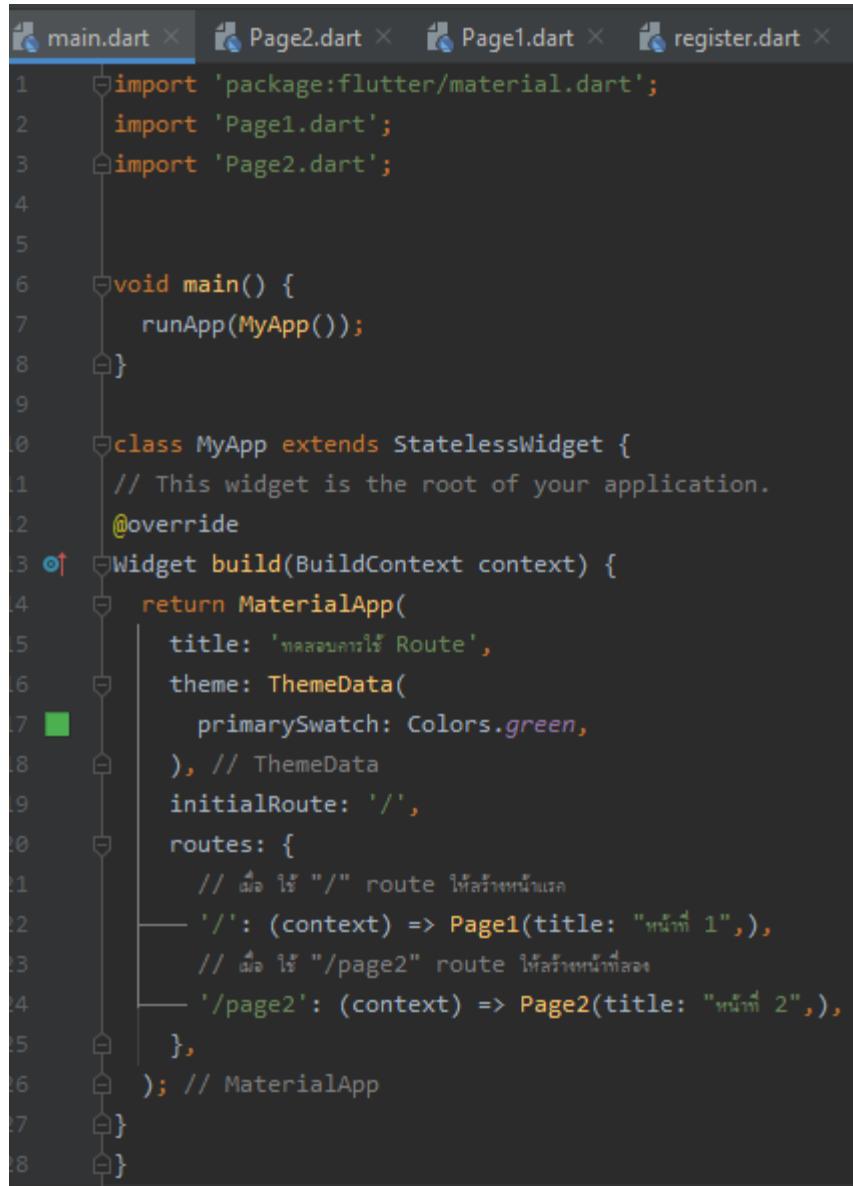
# การใช้ StatefulWidget การทำ route เพื่อเปิดอีกหน้า



เมื่อกดปุ่มแล้วจะเปลี่ยนไปอีกหน้า

# การทำ route เพื่อเปิดอีกหน้า

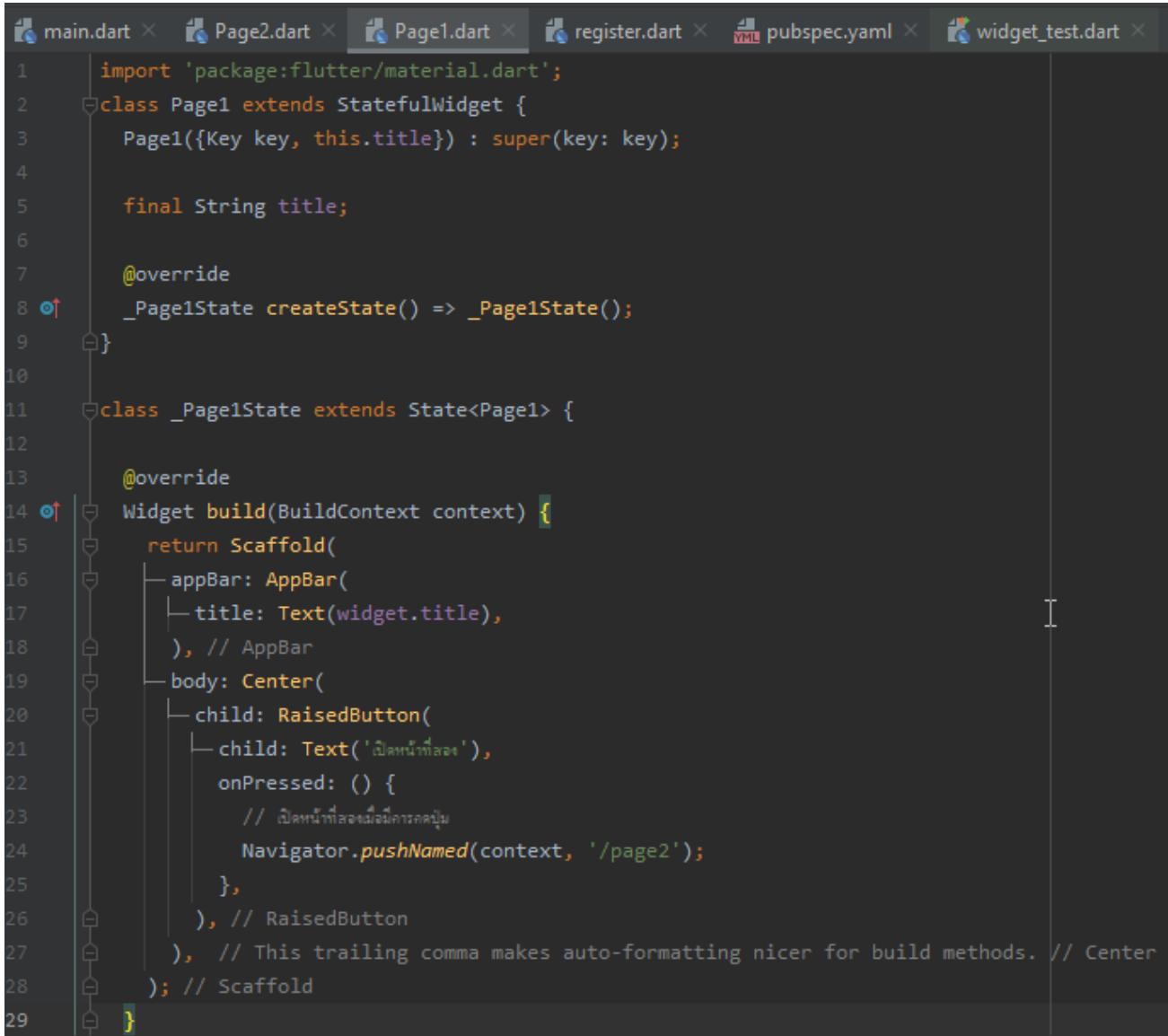
- Main.dart



```
main.dart x Page2.dart x Page1.dart x register.dart x
1 import 'package:flutter/material.dart';
2 import 'Page1.dart';
3 import 'Page2.dart';
4
5
6 void main() {
7   runApp(MyApp());
8 }
9
10 class MyApp extends StatelessWidget {
11   // This widget is the root of your application.
12   @override
13   Widget build(BuildContext context) {
14     return MaterialApp(
15       title: 'ทดสอบการ Route',
16       theme: ThemeData(
17         primarySwatch: Colors.green,
18       ), // ThemeData
19       initialRoute: '/',
20       routes: {
21         // ถ้า เป็น "/" route ให้รันหน้าแรก
22         '/': (context) => Page1(title: "หน้า 1"),
23         // ถ้า เป็น "/page2" route ให้รันหน้าที่สอง
24         '/page2': (context) => Page2(title: "หน้า 2"),
25       },
26     ); // MaterialApp
27 }
28 }
```

# การทำ route เพื่อเปิดอีกหน้า (ต่อ)

- Page1.dart

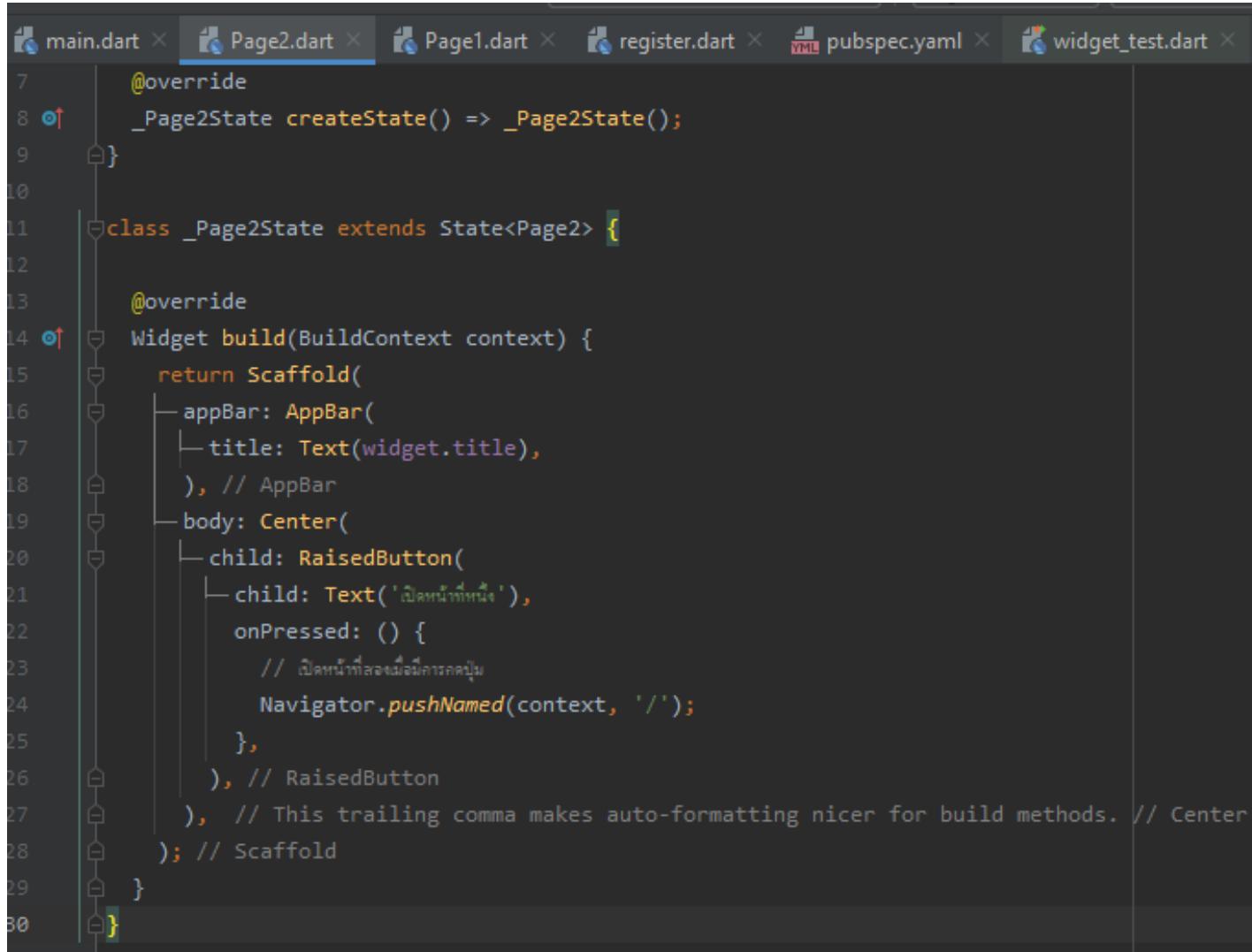


The screenshot shows a Flutter project in an IDE with multiple files listed in the tab bar: main.dart, Page2.dart, Page1.dart (selected), register.dart, pubspec.yaml, and widget\_test.dart.

```
1 import 'package:flutter/material.dart';
2 class Page1 extends StatefulWidget {
3     Page1({Key key, this.title}) : super(key: key);
4
5     final String title;
6
7     @override
8     _Page1State createState() => _Page1State();
9 }
10
11 class _Page1State extends State<Page1> {
12
13     @override
14     Widget build(BuildContext context) {
15         return Scaffold(
16             appBar: AppBar(
17                 title: Text(widget.title),
18             ), // AppBar
19             body: Center(
20                 child: RaisedButton(
21                     child: Text('ไปหน้าที่สอง'),
22                     onPressed: () {
23                         // ไปหน้าที่สองเมื่อคลิกปุ่ม
24                         Navigator.pushNamed(context, '/page2');
25                     },
26                 ), // RaisedButton
27             ), // This trailing comma makes auto-formatting nicer for build methods. // Center
28         ); // Scaffold
29 }
```

# การทำ route เพื่อเปิดอีกหน้า (ต่อ)

- Page2.dart



```
7     @override
8     _Page2State createState() => _Page2State();
9   }
10
11  class _Page2State extends State<Page2> {
12
13    @override
14    Widget build(BuildContext context) {
15      return Scaffold(
16        appBar: AppBar(
17          title: Text(widget.title),
18        ), // AppBar
19        body: Center(
20          child: RaisedButton(
21            child: Text('按我跳到第2頁'),
22            onPressed: () {
23              // 按我跳到第2頁
24              Navigator.pushNamed(context, '/');
25            },
26          ), // RaisedButton
27        ), // This trailing comma makes auto-formatting nicer for build methods. // Center
28      ); // Scaffold
29    }
30  }
```

# BottomAppBar

The screenshot shows the Flutter Inspector interface with the following details:

- Flutter Inspector View:** Shows the widget tree structure:
  - [root]
  - MyApp
  - MaterialApp
  - MyHomePage
  - Scaffold
    - Center
      - Column
        - Text
        - Text
    - AppBar
      - Text
      - Text
    - BottomAppBar
      - Text
      - FloatingActionButton
- Code View:** Displays the Dart code corresponding to the widget tree:

```
Text('You have pushed the button this many times:'), // Text
Text('$_counter', style: Theme.of(context).textTheme.headline4), // Text
], // <Widget>[]
), // Column
), // Center
bottomNavigationBar: BottomAppBar(
color: Colors.blue,
child: Text('Button'),
), // BottomAppBar
floatingActionButton: FloatingActionButton(
onPressed: _incrementCounter,
tooltip: 'Increment',
child: Icon(Icons.add),
), // This trailing comma makes auto-formatting nicer for build methods. // S
); // Scaffold
```
- Red Callout:** A red callout box with white text points to the `bottomNavigationBar` and `FloatingActionButton` sections of the code.
- Text in Callout:** The text in the callout is in Thai:

BottomAppBar ຈະอยู่ล่าง Body  
ก่อน FloatingActionButton

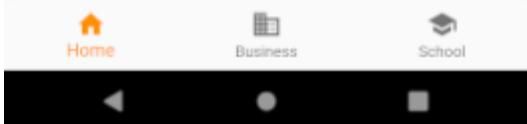
# BottomNavigationBar



Index 0: Home



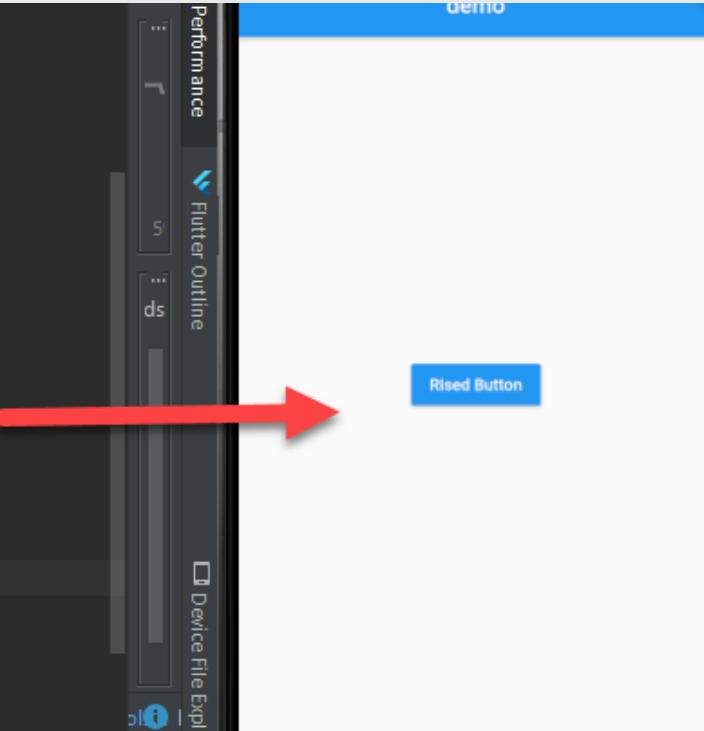
Index 1: Business



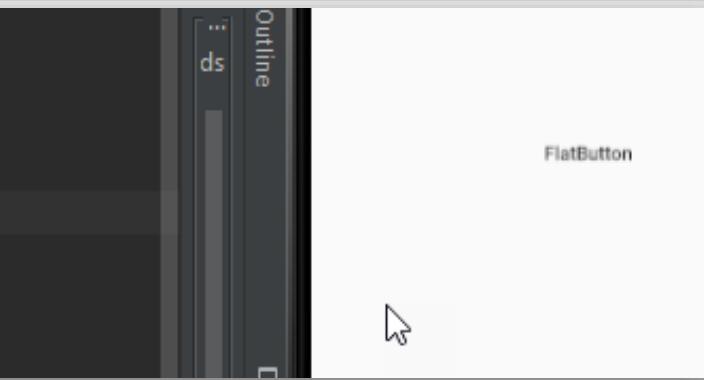
- <https://api.flutter.dev/flutter/widgets/Icon-class.html> หาเปลี่ยนไอคอนได้จากที่นี่
- <https://api.flutter.dev/flutter/material/Icons-class.html>

# Button

```
class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Center(child: Text('demo')),
      ), // AppBar
      body: Center(
        child: RaisedButton(
          onPressed: (){},
          color:Colors.blue,
          child:Text(
            'Rised Button',
            style: TextStyle(color: Colors.white),
          ) // Text
        ), // RaisedButton
      ); // Center, Scaffold
    }
}
```

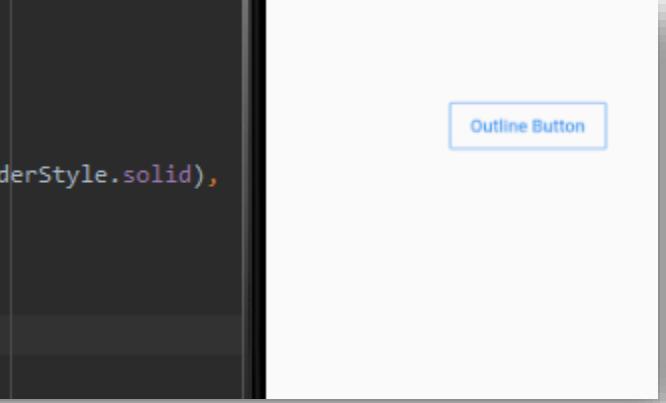


```
body: Center(
  child: FlatButton(
    onPressed: (){},
    child:Text(
      'FlatButton',
    ) // Text
  ), // FlatButton
)); // Center, Scaffold
```



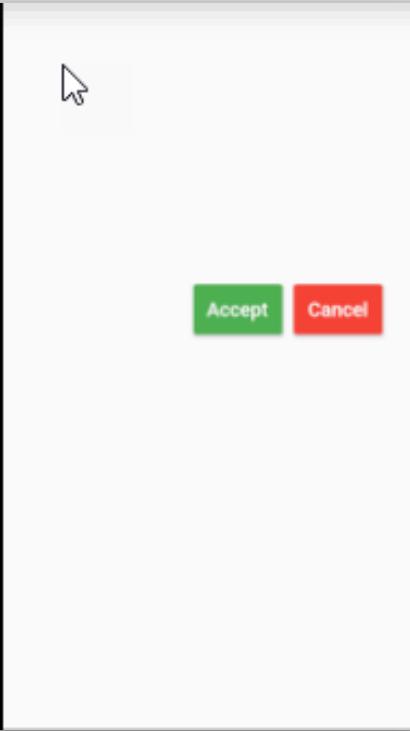
# Button

```
- body: Center(  
  child: OutlineButton(  
    onPressed: () => {},  
    textColor: Colors.blue,  
    borderSide: BorderSide(color: Colors.blue, width: 1.0, style: BorderStyle.solid),  
    child: Text(  
      'Outline Button',  
      ), // Text  
    ), // OutlineButton  
) // Center, Scaffold
```



Outline Button

```
- body: Center(  
  child: ButtonBar(  
    alignment: MainAxisAlignment.center,  
    children: <Widget>[  
      RaisedButton(  
        onPressed: () => {},  
        color: Colors.green,  
        child: Text('Accept', style: TextStyle(color: Colors.white),),  
      ), // RaisedButton  
      RaisedButton(  
        onPressed: () => {},  
        color: Colors.red,  
        child: Text('Cancel', style: TextStyle(color: Colors.white),),  
      ), // RaisedButton  
    ], // <Widget>[]  
) // ButtonBar  
) // Center, Scaffold
```



Accept Cancel

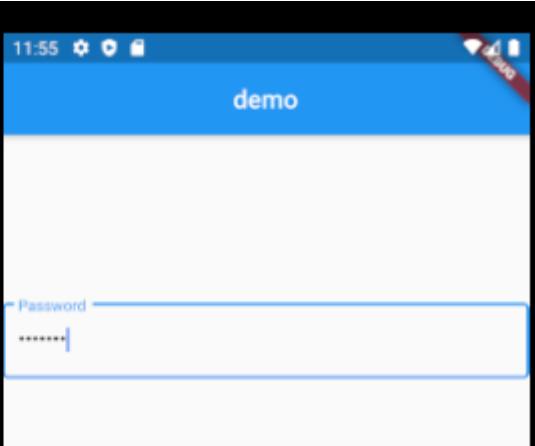
# TextField

TextField คือ Widget Input ที่จะรับค่าจากการกด key ตัวอักษรลงไปบนระบบ

```
— body: Center(  
|   — child: TextField(  
|     ) // TextField  
| ); // Center, Scaffold
```

สามารถกำหนดรูปแบบเมื่อกรอก password ได้

```
— body: Center(  
|   — child: TextField(  
|     obscureText: true,  
|     decoration: InputDecoration(  
|       border: OutlineInputBorder(),  
|       labelText: 'Password',  
|     ), // InputDecoration  
|   ) // TextField  
| ); // Center, Scaffold
```



# ตัวอย่างการกรอกข้อมูลใน TextField และกดปุ่มให้แสดงค่าที่กดได้

The image shows a Flutter application running on an Android emulator. The app has a blue header bar with the text 'input'. Below it is a white screen containing a grey button labeled 'update' and a text field. The text field displays the value '\_input'. The entire interface is contained within a black scaffold.

The code in the editor is as follows:

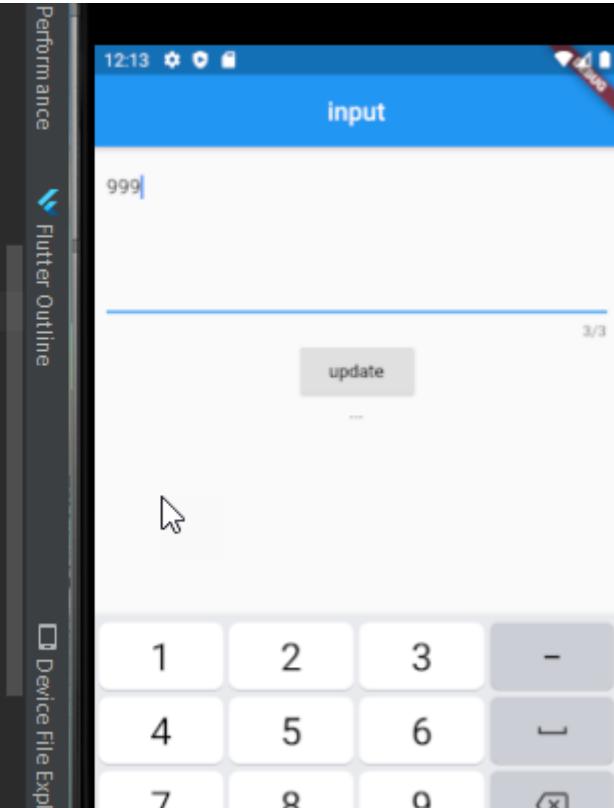
```
class _MyHomePageState extends State<MyHomePage> {
  String _input = "..."; ←
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Center(child: Text('input')),
      ), // AppBar
      body: Container(
        padding: EdgeInsets.all(10),
        child: Column(
          children: <Widget>[
            TextField(
              keyboardType: TextInputType.number, ←
              ), // TextField
            RaisedButton(child: Text('update'), onPressed: () {
              setState(() { ←
                _input = '';
              });
            }), // RaisedButton
            Text('_input'), ←
            ], // <Widget>[]
        ) // Column
      );
    ); // Container, Scaffold
  }
}
```

Red arrows point to several parts of the code and UI to highlight specific components or logic:

- A red arrow points to the variable declaration `_input = "...";`.
- A red arrow points to the `keyboardType: TextInputType.number,` line in the `TextField` constructor.
- A red arrow points to the `onPressed: () {` part of the `RaisedButton` constructor.
- A red arrow points to the `Text('_input')` in the `Column`.

# TextField มี Properties ที่น่าสนใจดังนี้

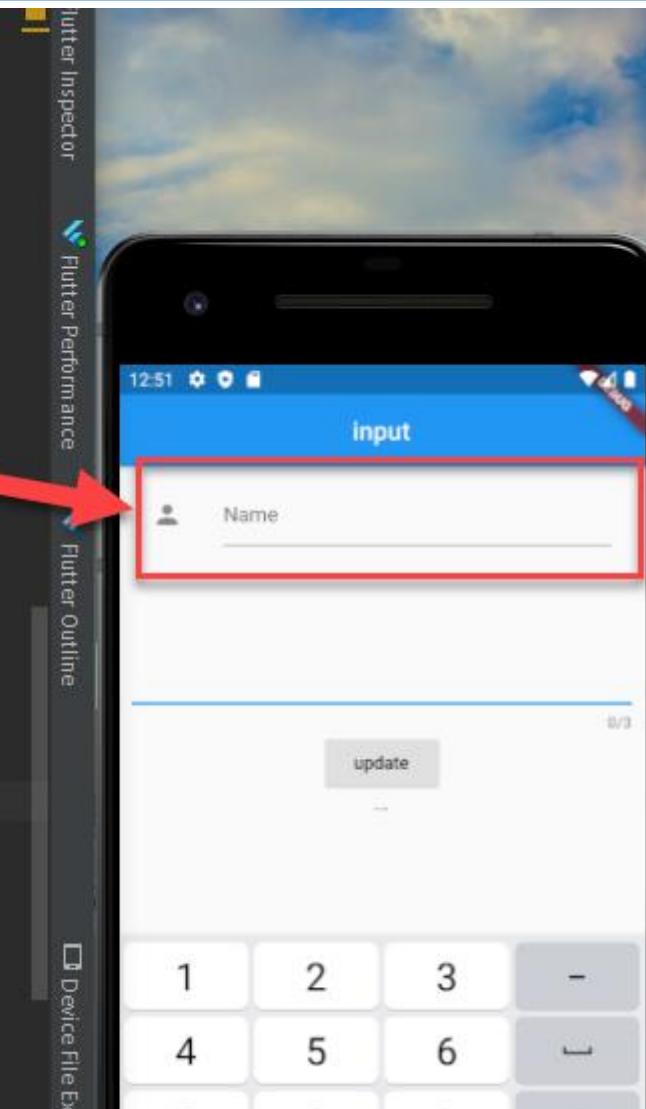
```
- body: Container(  
    padding: EdgeInsets.all(10),  
    child: Column(  
        children: <Widget>[  
            TextField(  
                keyboardType: TextInputType.number,  
                maxLength: 3,  
                maxLines: 5,  
            ), // TextField  
            RaisedButton(child: Text('update'), onPressed: (){  
                setState(() {  
  
                });  
            }), // RaisedButton  
            Text('${_input}'),  
        ], // <Widget>[]  
    ) // Column  
  
)); // Container, Scaffold
```



- keyboardType กำหนดรูปแบบการกรอกข้อมูล เช่นกรอกแต่ตัวเลข กรอกอีเมล
- maxLength กำหนดขนาดความยาวของตัวอักษร
- maxLines กำหนดจำนวนไลน์ที่สามารถกรอกได้ (จำนวนบรรทัดในการกรอก)
- TextInputAction สามารถกำหนดให้มีรูปแบบของคีย์บอร์ดใน keyboard ได้ (TextInputAction=search)

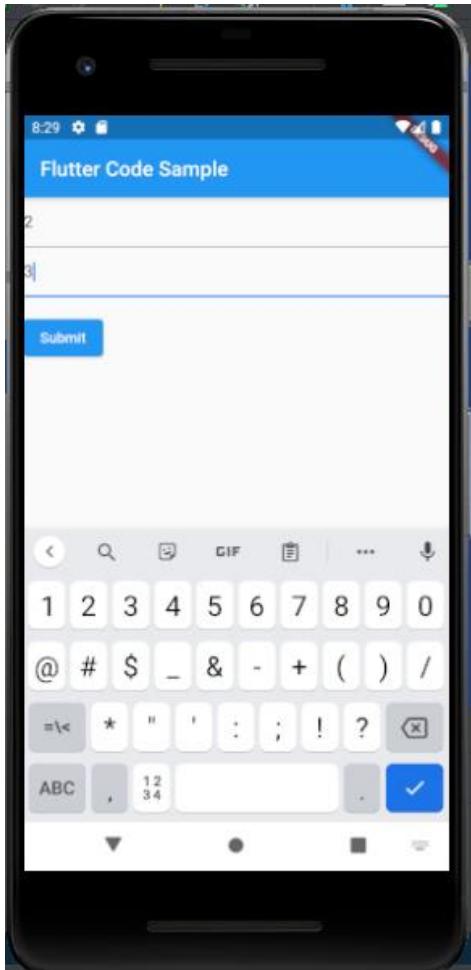
# ListTile widget ที่สามารถใส่ leading และ textfield ได้

```
body: Container(
  padding: EdgeInsets.all(10),
  child: Column(
    children: <Widget>[
      ListTile(
        leading: Icon(Icons.person),
        title: TextField(
          decoration: InputDecoration(
            hintText: "Name",
          ),
        ),
        ),
      ),
      TextField(
        keyboardType: TextInputType.number,
        maxLength: 3,
        maxLines: 5,
      ),
      RaisedButton(child: Text('update'), onPressed: () {
        setState(() {
          });
      }),
      Text('${_input}'),
    ],
  ),
)
```



# Input Text

- การตั้งค่า input ของ text



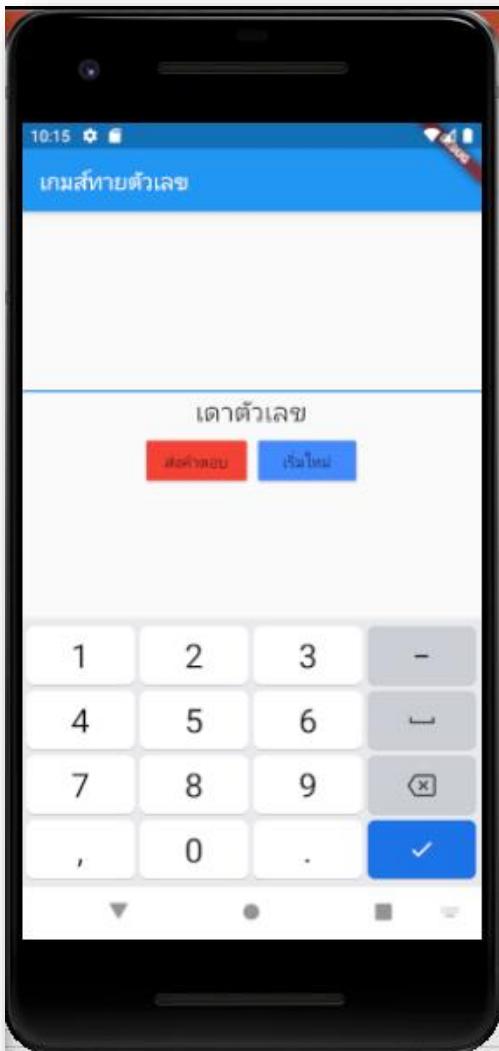
สร้างจะใช้ **TextField** และใช้ **Controller** เป็น id

```
crossAxisAlignment: CrossAxisAlignment.start,  
children: <Widget>[  
  -> TextField(  
    controller: _ageController,  
    decoration: const InputDecoration(  
      hintText: 'Enter your age',  
    ), // InputDecoration  
    validator: (value) {  
      if (value.isEmpty) {  
        return 'Please enter some text';  
      }  
      return null;  
    },  
  ), // TextField  
  -> TextField(  
    controller: _heightController,  
    decoration: const InputDecoration(  
      hintText: 'Enter your height',  
    ), // InputDecoration
```

และจะสร้างตัวแปลง **controller** มารับค่า **input** ที่สร้าง

```
/// This is the private State class that goes with My StatefulWidget.  
class _My StatefulWidget extends State<My StatefulWidget> {  
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();  
  final TextEditingController _ageController = TextEditingController();  
  final TextEditingController _heightController = TextEditingController();  
  @override  
  Widget build(BuildContext context) {  
    return Form(  
      key: _formKey,  
      child: Column(  
        crossAxisAlignment: CrossAxisAlignment.start,
```

# App 1 แอพทำนายตัวเลข



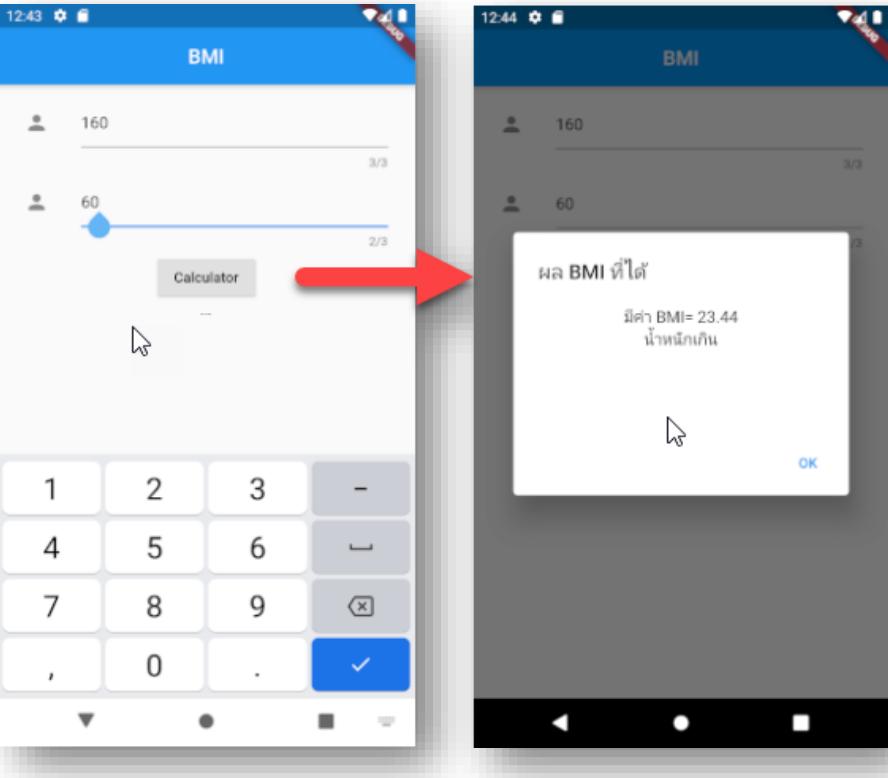
## ขั้นตอนการทำ

- สร้างแอพ Flutter ชื่อ week3\_random\_number โดยมี Widget ต่าง ๆ ดังนี้
  - สร้าง TextField ให้กรอกได้เฉพาะตัวเลข
  - สร้าง Text เพื่อแสดงข้อความผลลัพธ์ที่ได้
  - สร้างปุ่ม ส่งคำตอบสีแดง และ ปุ่มเริ่มใหม่สีน้ำเงิน
- วิธีเล่นคือการกรอกตัวเลขที่คิดว่าจะเป็นคำตอบลงไปใน TextField และกดปุ่มส่งคำตอบ ถ้าส่งผิด จะขึ้นข้อความว่า ผิดคำตอบที่ถูกคือ (ตัวเลขที่แอพแรนdomขึ้นมา) ถ้าส่งถูกจะขึ้นว่าสุดยอด
- กดปุ่มเล่นใหม่ เพื่อเริ่มเดาตัวเลขใหม่อีกครั้ง

## สิ่งที่ได้

- สามารถวางแผน Layout ได้
- เรียนรู้เรื่อง State และการสร้างตัวแปร ตลอดจนการกำหนด Controller ทำงานกับ TextField และปุ่มกด
- การเรียก Import class math ในภาษา dart มาเพื่อใช้ random number

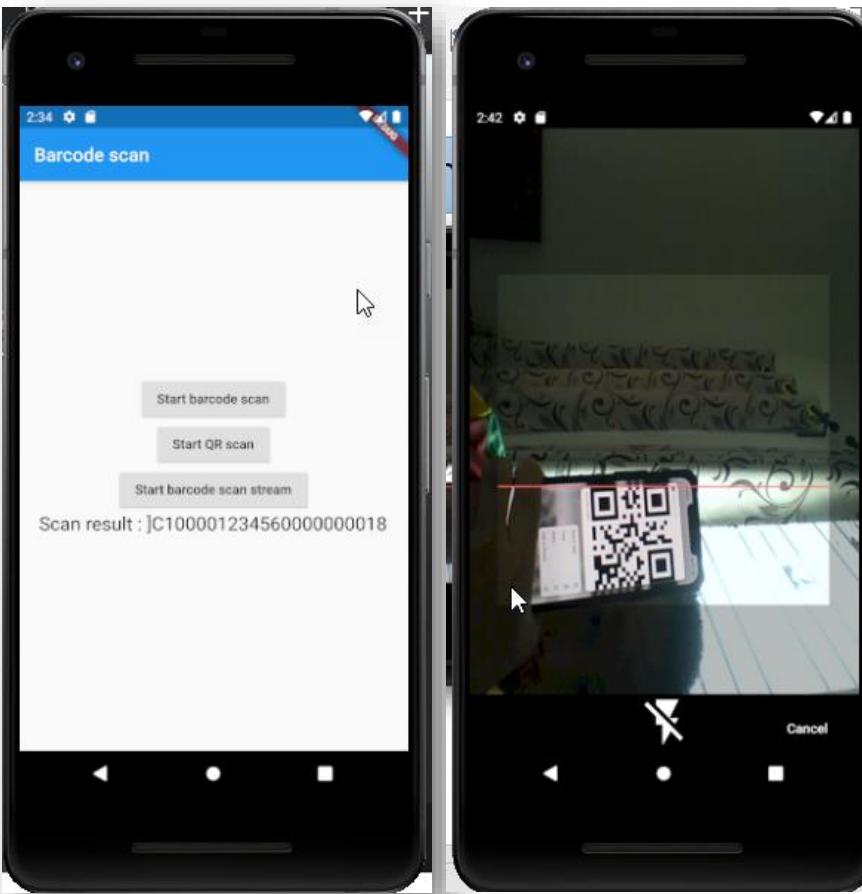
# App 2 การคำนวน BMI



## ขั้นตอนการทำ

- สร้างแอพ Flutter ชื่อ week3\_scanbarcode โดยมี Widget ต่าง ๆ ดังนี้
    - สร้าง TextField เก็บค่า Width และ Height
      - กำหนดให้กรอกได้แต่ตัวเลข 3 ตัวอักษร
    - สร้าง ปุ่ม กด Calculator
  - การใช้งานกรอกน้ำหนักส่วนสูง และกดปุ่ม Calculator คำนวนค่า BMI และแสดง Popup ขึ้นมา
- ## สิ่งที่ได้
- สามารถถาง Layout ได้
  - เรียนรู้เรื่อง State เรียนรู้การใช้ showDialog และใช้ Widget AlertDialog ที่สามารถปรับแต่งความสวยงามได้ และเรียนรู้การส่งค่าตัวแปล และการตัดจุดทศนิยม รวมไปถึง การใช้ Navigator.pop เพื่อยกเลิก Widget AlertDialog

# App 3 Scan Bar Code



## ขั้นตอนการทำ

- สร้างแอพ Flutter ชื่อ week3\_scanbarcode โดยมี Widget ต่าง ๆ ดังนี้
  - สร้างปุ่ม Scan Barcode
  - สร้างปุ่ม Scan QR Code
  - สร้างปุ่ม Scan Barcode แบบ StreamMode
  - สร้าง Text แสดงผลการ Scan
- การใช้งานกดปุ่มที่ต้องการสแกน กล้องเปิดแล้วนำไฟล์ภาพ Barcode หรือ QrCode มาสแกน

## ลิ่งที่ได้

- สามารถดู Layout ได้
- เรียนรู้เรื่อง State เรียนรู้การใช้ Future และ Asynchronous การใช้ plugin flutter\_barcode\_scanner