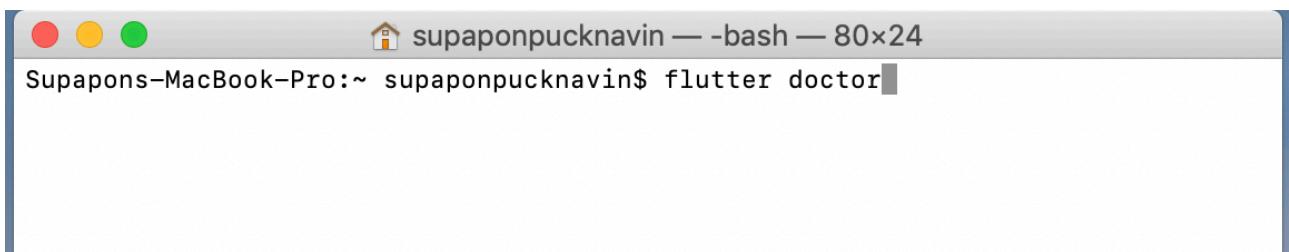


ในบทนี้จะขอแนะนำการพัฒนาแอปด้วย Flutter สำหรับคนที่เคยมีพื้นฐานเขียนแอปมาแล้วนะครับ วิธีการติดตั้งเครื่องมือพัฒนาแอป สามารถเข้าไปดูได้ที่ <https://flutter.dev>

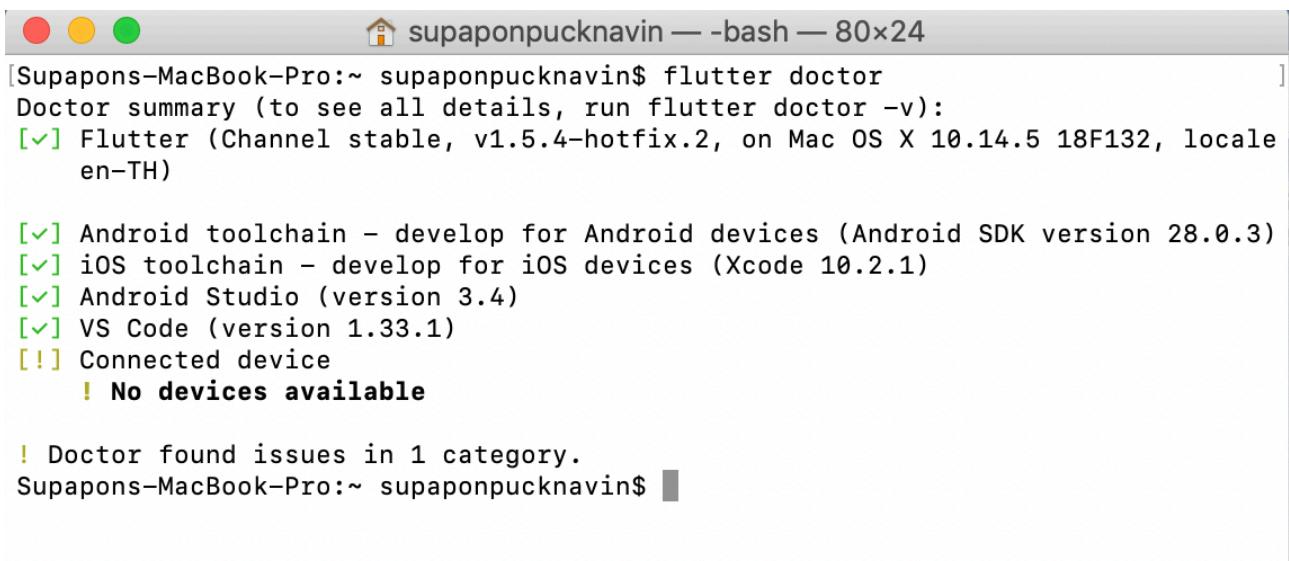
IDE ที่เราจะใช้วนนี้คือ Visual Studio Code (<https://code.visualstudio.com>)

เริ่มเลยดีกว่า เปิด Terminal ขึ้นมาพิมคำสั่ง ‘flutter doctor’ เพื่อตรวจสอบความพร้อมของเครื่องมือ



```
supaponpucknavin — bash — 80x24
Supapons-MacBook-Pro:~ supaponpucknavin$ flutter doctor
```

ขึ้นเขียวหมดอย่างนี้คือ ใช้ได้ ถ้าใครขาดตรงไหนก็ซ่อมตามที่มันบอกได้เลยครับ



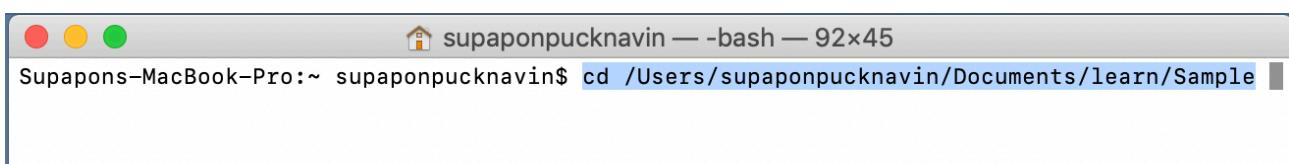
```
supaponpucknavin — bash — 80x24
[Supapons-MacBook-Pro:~ supaponpucknavin$ flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, v1.5.4-hotfix.2, on Mac OS X 10.14.5 18F132, locale en-TH)

[✓] Android toolchain - develop for Android devices (Android SDK version 28.0.3)
[✓] iOS toolchain - develop for iOS devices (Xcode 10.2.1)
[✓] Android Studio (version 3.4)
[✓] VS Code (version 1.33.1)
[!] Connected device
    ! No devices available

! Doctor found issues in 1 category.
Supapons-MacBook-Pro:~ supaponpucknavin$ ]
```

เรามาสร้างโปรเจคใหม่ด้วยคำสั่ง ‘flutter create <project name>’

1. เข้าไปยังโฟลเดอร์ ที่เราต้องการสร้างโปรเจ็คก่อน



```
supaponpucknavin — bash — 92x45
Supapons-MacBook-Pro:~ supaponpucknavin$ cd /Users/supaponpucknavin/Documents/learn/Sample
```

## 2. ใส่คำสั่ง ‘flutter create <project name>’ ตัวอย่างนี้ผมใช้ชื่อว่า flutter01

```
Sample — bash — 92x45
Supapons-MacBook-Pro:Sample supaponpucknavin$ flutter create flutter01

Sample — bash — 92x45
flutter01/android/app/src/main/res/mipmap-xxxhdpi/ic_launcher.png (created)
flutter01/android/app/src/main/res/mipmap-xhdpi/ic_launcher.png (created)
flutter01/android/app/src/main/res/values/styles.xml (created)
flutter01/android/app/src/main/res/mipmap-xhdpi/ic_launcher.png (created)
flutter01/android/app/src/main/AndroidManifest.xml (created)
flutter01/android/app/debug/AndroidManifest.xml (created)
flutter01/android/gradle/wrapper/gradle-wrapper.properties (created)
flutter01/android/gradle.properties (created)
flutter01/android/settings.gradle (created)
flutter01/pubspec.yaml (created)
flutter01/README.md (created)
flutter01/lib/main.dart (created)
flutter01/android/app/build.gradle (created)
flutter01/android/app/src/main/java/com/example/flutter01/MainActivity.java (created)
flutter01/android/build.gradle (created)
flutter01/android/flutter01_android.iml (created)
flutter01/.idea/runConfigurations/main_dart.xml (created)
flutter01/.idea/libraries/Flutter_for_Android.xml (created)
flutter01/.idea/libraries/Dart_SDK.xml (created)
flutter01/.idea/libraries/KotlinJavaRuntime.xml (created)
flutter01/.idea/modules.xml (created)
flutter01/.idea/workspace.xml (created)
Running "flutter packages get" in flutter01...                                13.2s
Wrote 66 files.

All done!
[✓] Flutter is fully installed. (Channel stable, v1.5.4-hotfix.2, on Mac OS X 10.14.5
    18F132, locale en-TH)
[✓] Android toolchain - develop for Android devices is fully installed. (Android SDK version
    28.0.3)
[✓] iOS toolchain - develop for iOS devices is fully installed. (Xcode 10.2.1)
[✓] Android Studio is fully installed. (version 3.4)
[✓] VS Code is fully installed. (version 1.33.1)
[!] Connected device is not available.

Run "flutter doctor" for information about installing additional components.

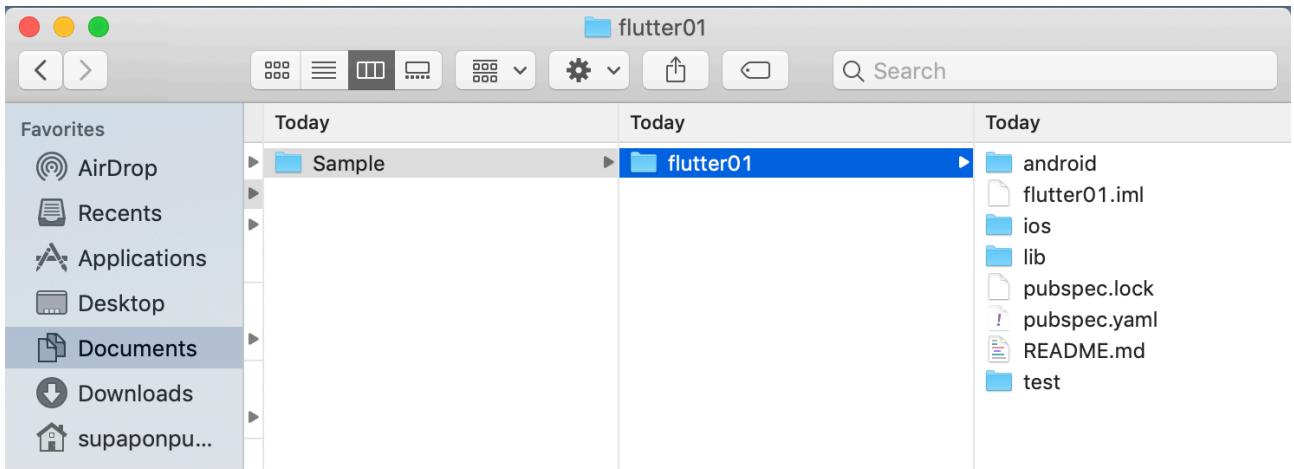
In order to run your application, type:

$ cd flutter01
$ flutter run

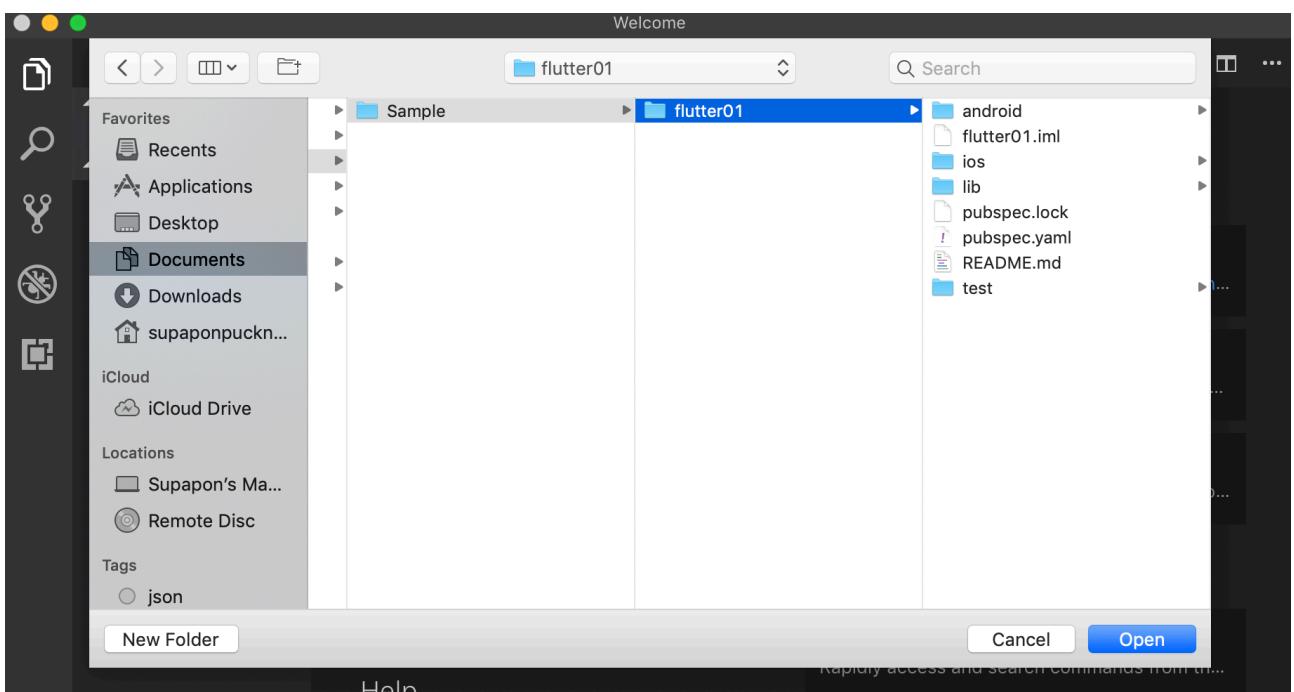
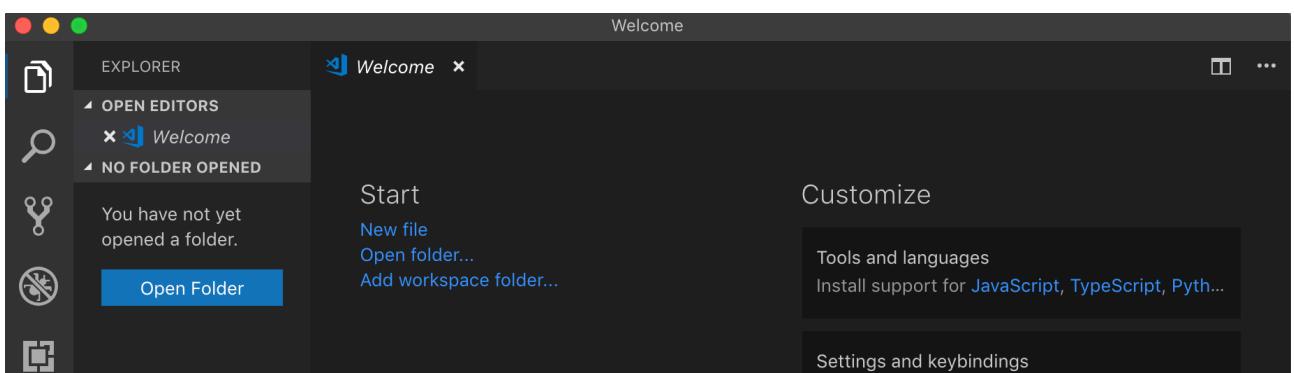
Your application code is in flutter01/lib/main.dart.

Supapons-MacBook-Pro:Sample supaponpucknavin$
```

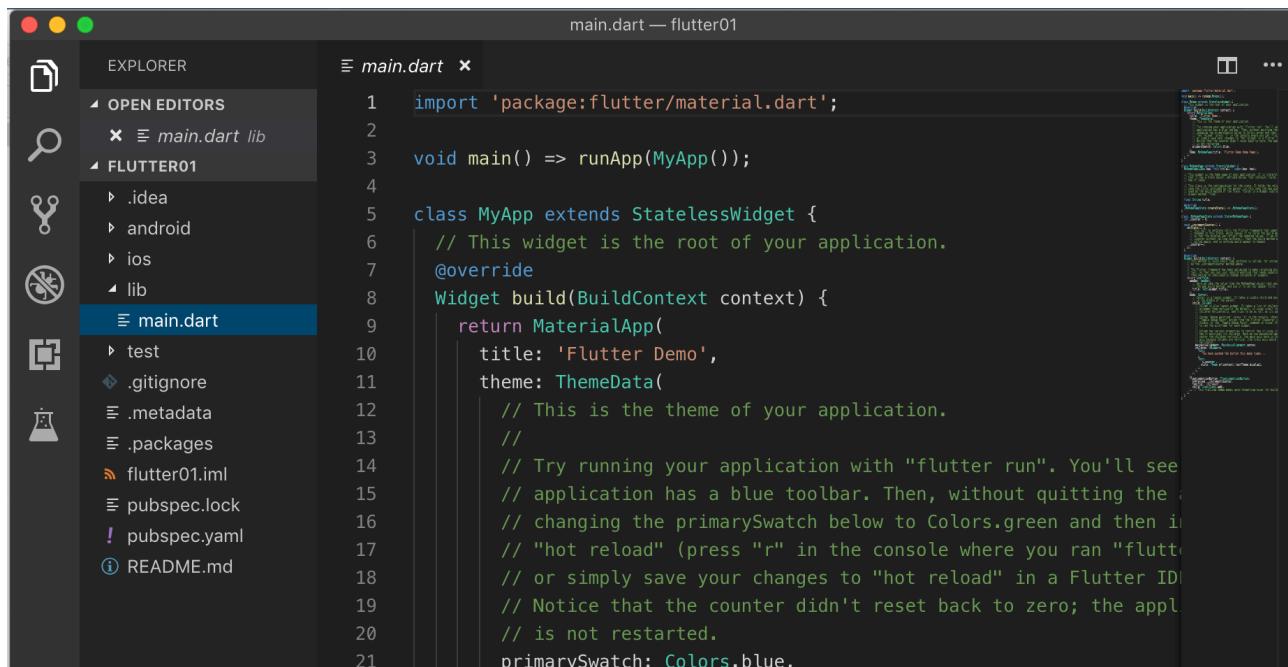
## มันก็จะสร้าง Project เริ่มต้นให้เรา



## เปิดโปรเจ็คขึ้นมาดูเลย (Open Folder)



## ไฟล์ที่เราจะเริ่มกันคือ lib/main.dart



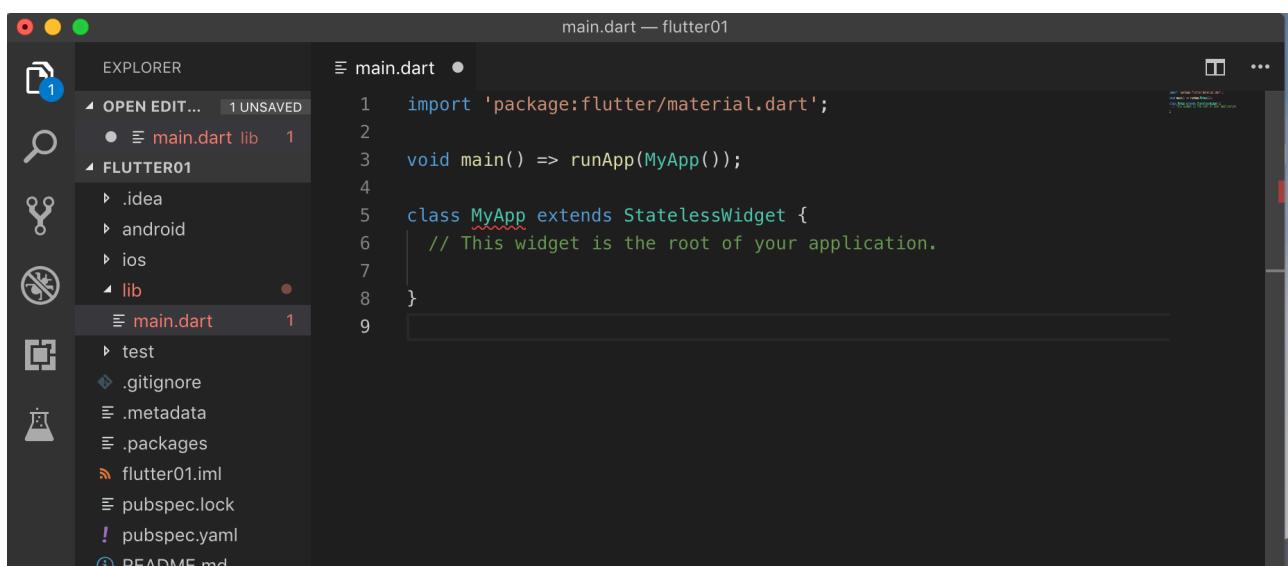
The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows the project structure. The `lib` folder is expanded, and `main.dart` is selected.
- Code Editor:** The file `main.dart` is open, displaying the initial code for a Flutter application.
- Output Panel:** On the right side, there is a panel showing the output of the build process, including logs and build statistics.

```
main.dart — flutter01
EXPLORER
OPEN EDITORS
FLUTTER01
lib
main.dart
test
.gitignore
.metadata
.packages
flutter01.iml
pubspec.lock
! pubspec.yaml
README.md

1 import 'package:flutter/material.dart';
2
3 void main() => runApp(MyApp());
4
5 class MyApp extends StatelessWidget {
6     // This widget is the root of your application.
7     @override
8     Widget build(BuildContext context) {
9         return MaterialApp(
10             title: 'Flutter Demo',
11             theme: ThemeData(
12                 // This is the theme of your application.
13                 //
14                 // Try running your application with "flutter run". You'll see
15                 // application has a blue toolbar. Then, without quitting the
16                 // changing the primarySwatch below to Colors.green and then
17                 // "hot reload" (press "r" in the console where you ran "flutter
18                 // or simply save your changes to "hot reload" in a Flutter IDE).
19                 // Notice that the counter didn't reset back to zero; the appli
20                 // is not restarted.
21             primarySwatch: Colors.blue,
```

ลบโค้ดที่ให้มาเหลือตามรูปเลยครบ  
ทุกสิ่งเริ่มต้นที่ main()



The screenshot shows the VS Code interface after the `MyApp` class has been removed from the code editor. The code now only contains the `void main()` function and its imports.

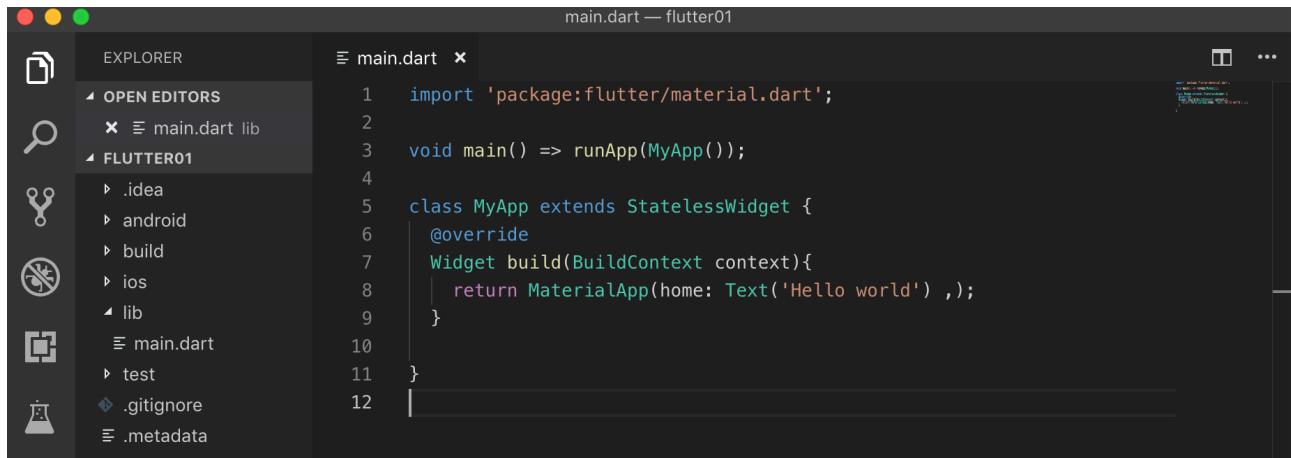
```
main.dart — flutter01
EXPLORER
OPEN EDIT... 1 UNSAVED
FLUTTER01
lib
main.dart 1
test
.gitignore
.metadata
.packages
flutter01.iml
pubspec.lock
! pubspec.yaml
README.md

1 import 'package:flutter/material.dart';
2
3 void main() => runApp(MyApp());
```

เมื่อเริ่มโปรแกรม Flutter จะเริ่มต้นที่ พังก์ชัน `main()` จาก โค้ดคือ พังก์ชัน `main()` ก็จะส่ง `MyApp` กลับไปให้  
`MyApp` คืออะไร? `MyApp` คือ `Widget`.  
`Widget` คืออะไร? `Widget` เป็นเหมือน `View` ใน iOS นั้นๆ  
แนวคิดการสร้างแอฟของ Flutter คือการประกอบ `Widget` หลายๆ อันเข้าด้วยกัน เรา  
มาลองสร้างกันดีกว่า

เมื่อ คลาส Widget ถูกเรียก มันจะวิ่งมาหา พังก์ชั่นชื่อ build() เราจะสร้าง Widget ที่ พังก์ชั่นนี้

MerrialApp() เป็น Widget หลักที่จะควบคุมรูปแบบ สีลักษณะของแอปเรา ขั้นแรกเราจะแสดง Hello world ขึ้นมา



The screenshot shows the VS Code interface with the main.dart file open in the editor. The code defines a simple Flutter application:

```
main.dart — flutter01
EXPLORER
OPEN EDITORS
main.dart lib
FLUTTER01
.idea
android
build
ios
lib
main.dart
test
.gitignore
.metadata
main.dart
import 'package:flutter/material.dart';
void main() => runApp(MyApp());
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context){
    return MaterialApp(home: Text('Hello world') ,);
}
}
```

main.dart

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context){
    return MaterialApp(home: Text('Hello world') ,);
}

}
```

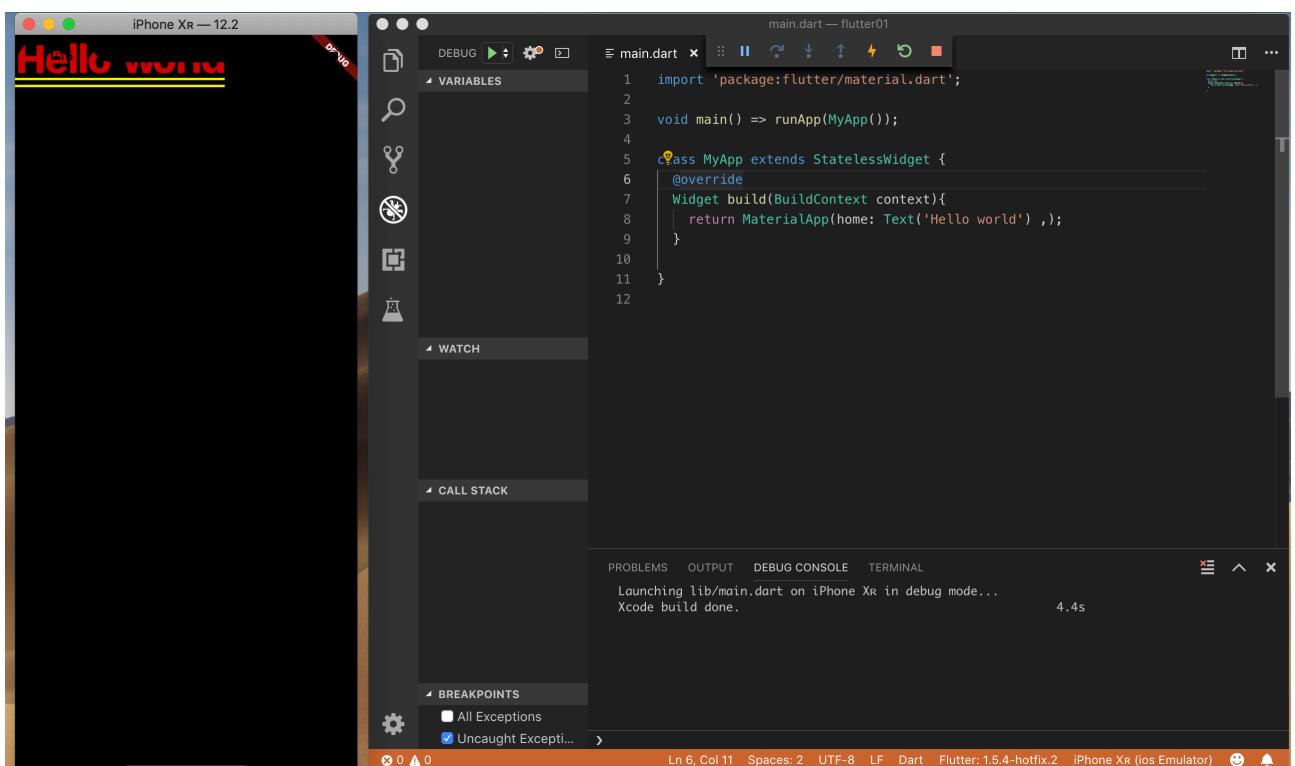
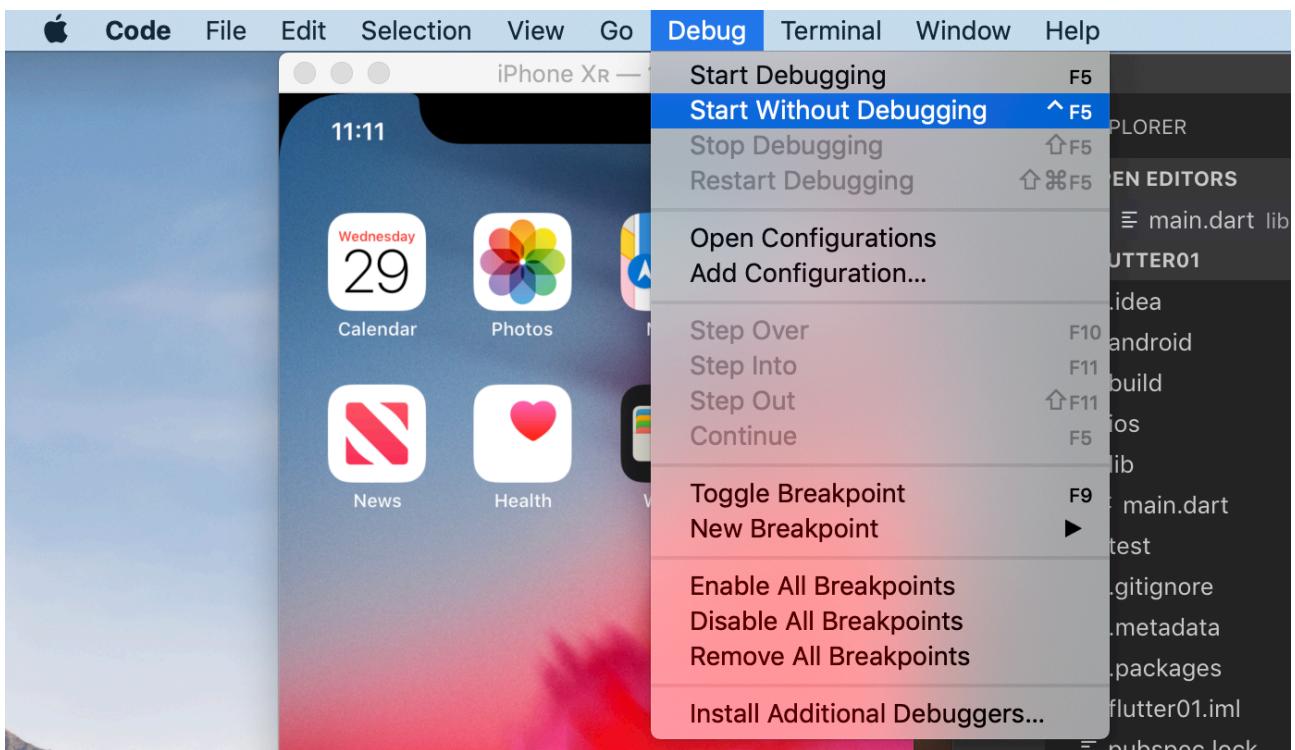
เปิด Terminal ใส่คำสั่ง ‘open -a Simulator’ เพื่อเปิด Simulator ขึ้นมาเพื่อทดสอบ โค้ด



The screenshot shows a Mac OS X terminal window titled 'Sample — -bash — 92x45'. The user has typed the command 'open -a Simulator' into the terminal.

```
Sample — -bash — 92x45
Supaporn-MacBook-Pro:Sample supaponpucknavin$ open -a Simulator
```

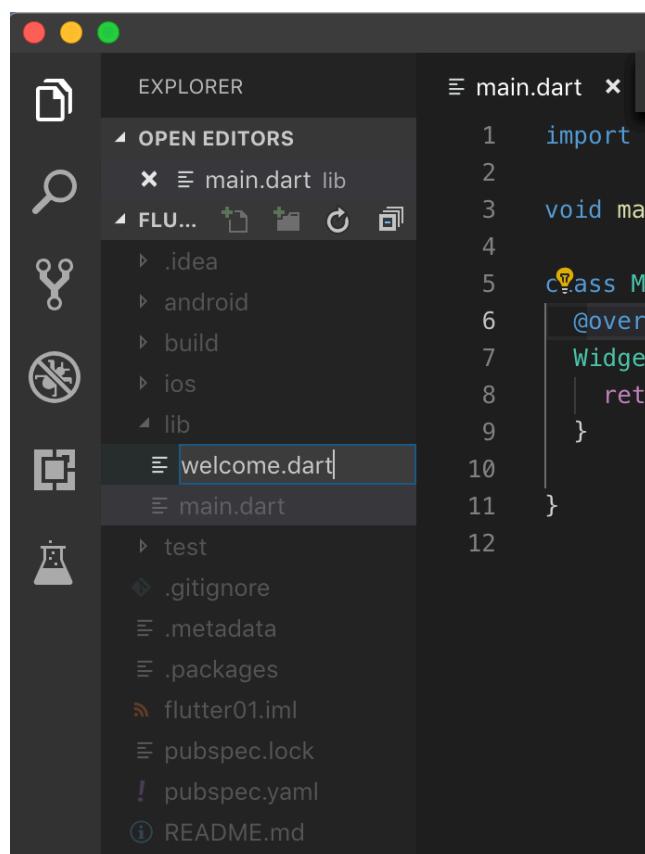
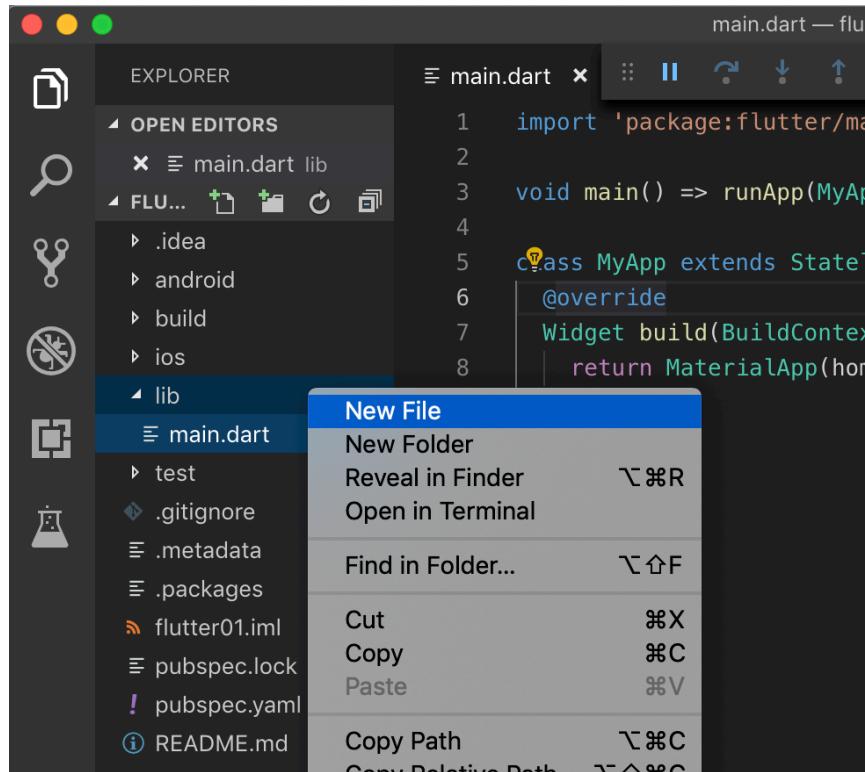
## ทดสอบโปรแกรม โดยไปที่ Debug > Start Debugging



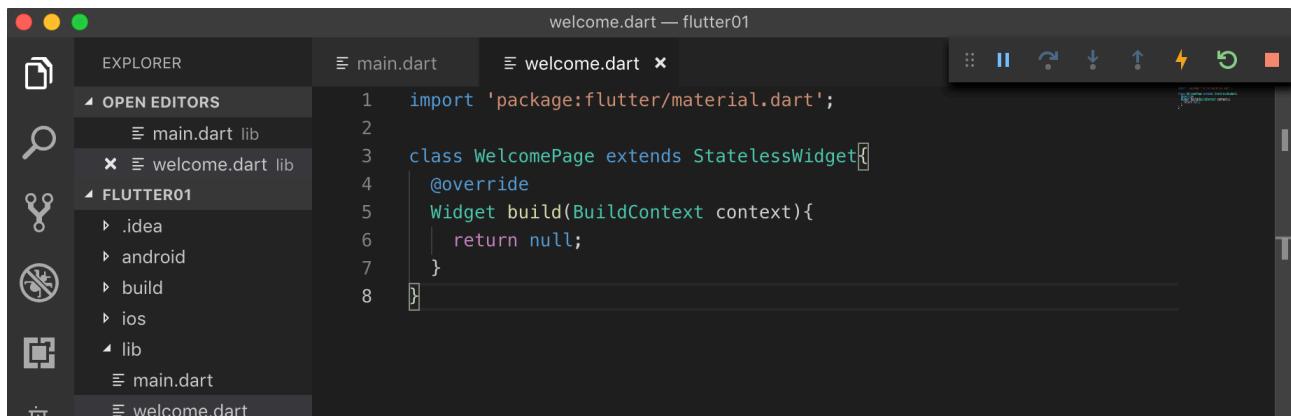
เห็น Hello world นั่นมั้ย โดนกล้องบังอยู่ ชาaha

ເອົາໃໝ່ ເຮັດວຽກສ້າງหน້າຕ້ອນຮັບຜູ້ໃຊ້ຂຶ້ນມາຈຳກັດໜ້າທີ່

ສ້າງໄຟລ໌ໃໝ່ຂຶ້ນມາຊື່ ‘welcome.dart’



สร้างคลาส 'WelcomePage' ขึ้นมา ก็อปจาก MyApp มาเลยรูปแบบมาตรฐาน



The screenshot shows a code editor interface with a dark theme. On the left is an 'EXPLORER' sidebar containing files like 'main.dart lib', 'welcome.dart lib', and 'FLUTTER01' which includes '.idea', 'android', 'build', 'ios', and 'lib' subfolders. The main area has tabs for 'main.dart' and 'welcome.dart'. The 'welcome.dart' tab is active, displaying the following code:

```
1 import 'package:flutter/material.dart';
2
3 class WelcomePage extends StatelessWidget{
4   @override
5     Widget build(BuildContext context){
6       return null;
7     }
8 }
```

### welcome.dart

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

class WelcomePage extends StatelessWidget{
  @override
  Widget build(BuildContext context){
    return null;
  }
}
```

ตอนนี้ return null ไปก่อน กลับไปแก้โค้ดที่ main.dart

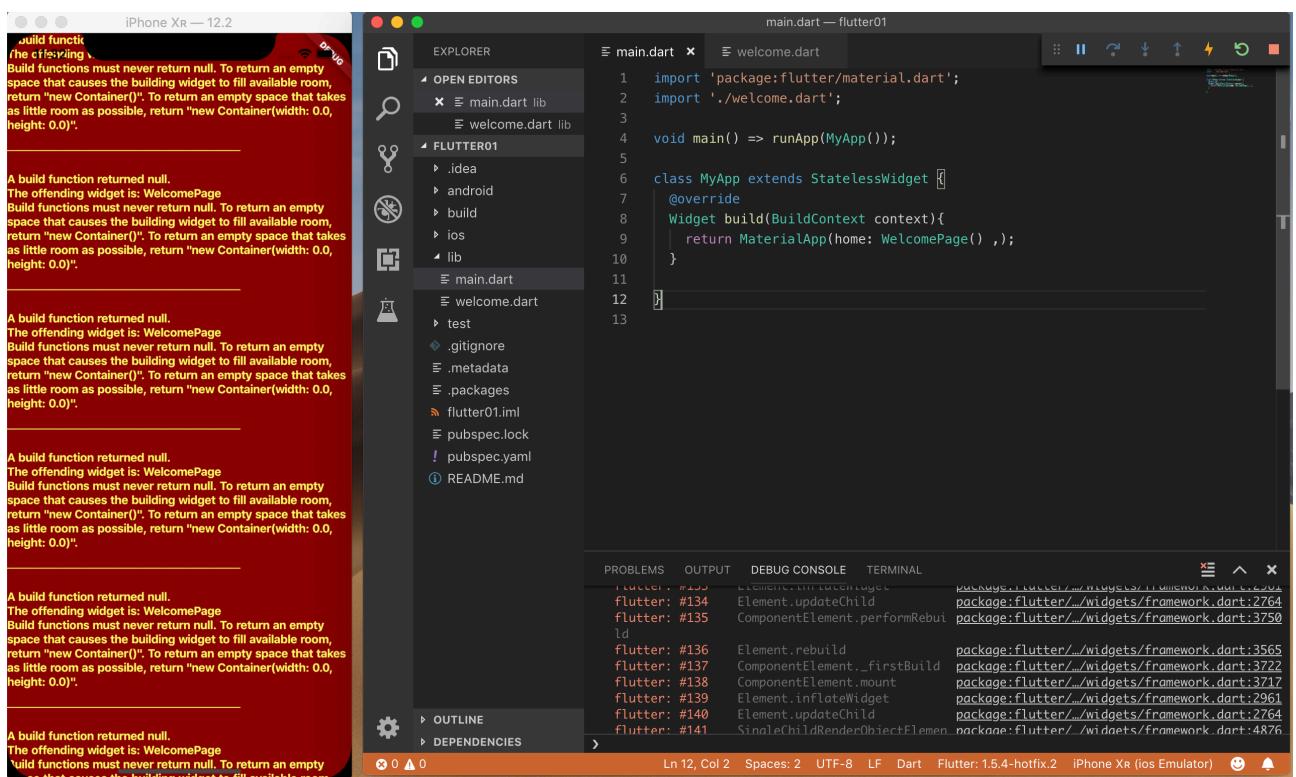
main.dart

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
    @override
    Widget build(BuildContext context){
        return MaterialApp(home: WelcomePage() ,);
    }
}
```

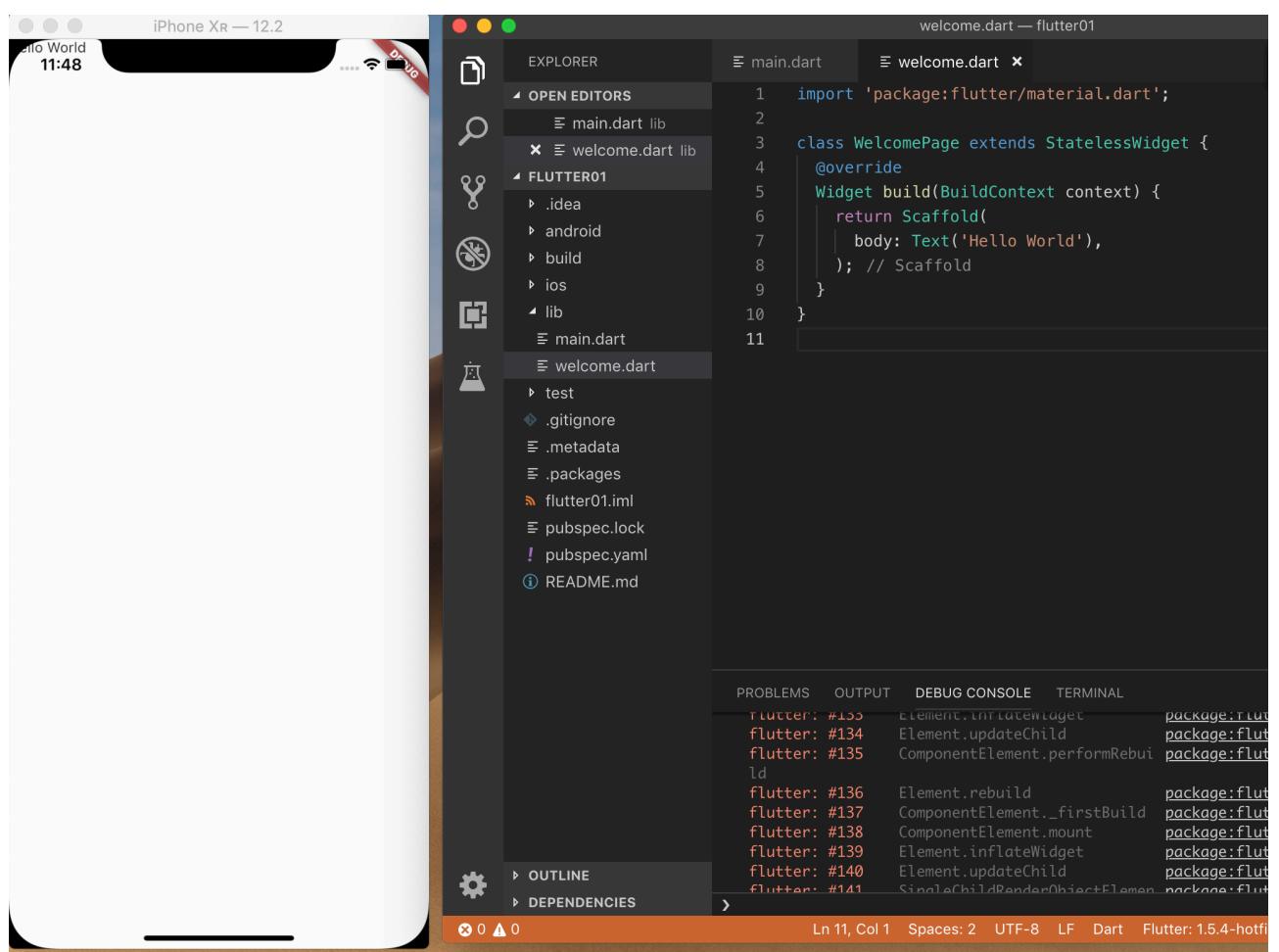
ลองทดสอบ ใหม่จะเห็นแอปเราแจ้ง error ขึ้นเต็มไปหมด ก็ແນລະ return null นินา



เพิ่มโค้ดที่ welcome.dart

welcome.dart > class WelcomePage

```
class WelcomePage extends StatelessWidget {  
    @override  
    Widget build(BuildContext context) {  
        return Scaffold(  
            body: Text('Hello World'),  
        );  
    }  
}
```



เห็น Hello world ที่มุมซ้ายบนนั่นนี้มั้ย

Scaffold เป็น Widget พื้นฐานชนิดนึงที่ใช้บ่อยมาก เราสามารถดูได้ว่ามี Widget อะไรให้เราใช้งานได้ที่ <https://flutter.dev/docs/development/ui/widgets> โดยที่ต่อไปเราจะสร้างหน้าจอแสดงตัวเลขและปุ่มกด

เพิ่มฟังก์ชันใน welcome.dart > class WelcomePage

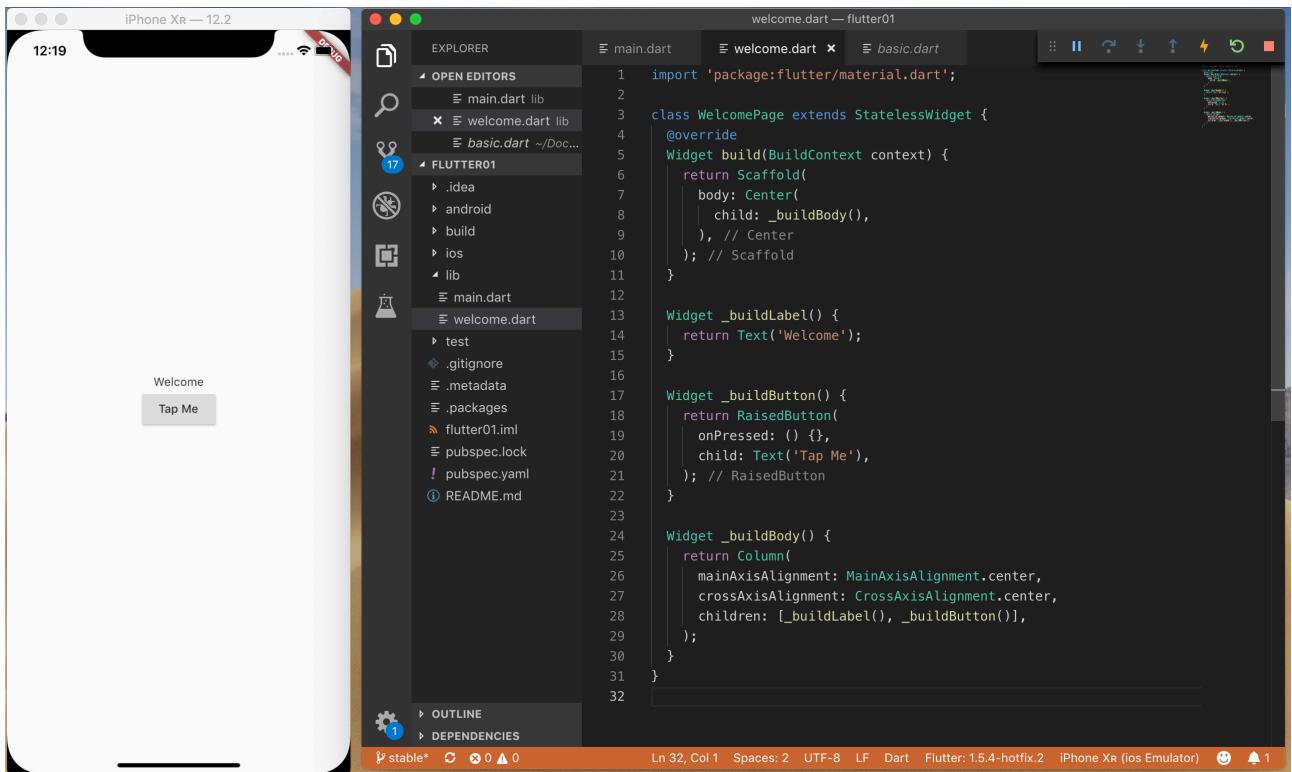
```
Widget _buildLabel() {
    return Text('Welcome');
}

Widget _buildButton() {
    return RaisedButton(
        onPressed: () {},
        child: Text('Tap Me'),
    );
}

Widget _buildBody() {
    return Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [_buildLabel(), _buildButton()],
    );
}
```

แก้ไขฟังก์ชัน build()

```
@override
Widget build(BuildContext context) {
    return Scaffold(
        body: Center(
            child: _buildBody(),
        ),
    );
}
```

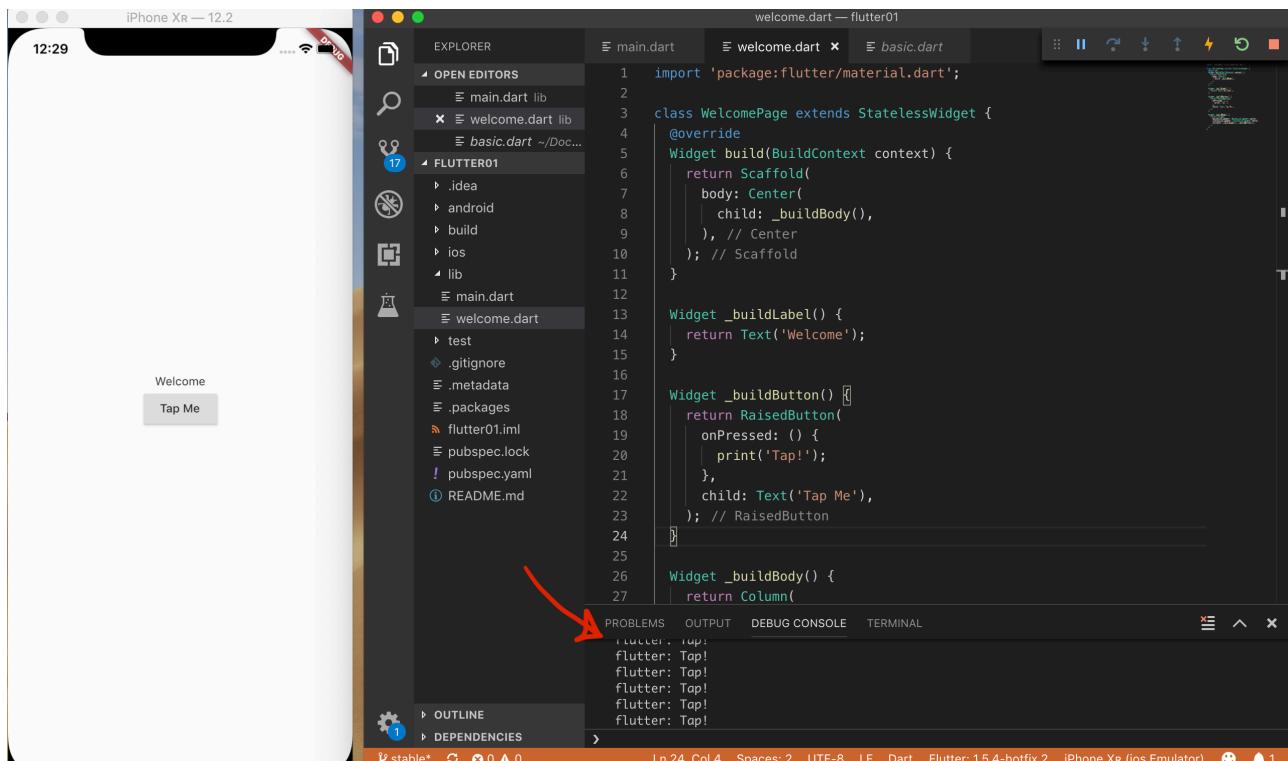


ได้แล้วๆ UI อย่างที่ต้องการ แต่กดปุ่มแล้วก็ยังไม่มีอะไรเกิดขึ้น ไปดูที่ฟังก์ชัน `_buildButton()` เพิ่มโค้ดเข้าไปเป็น

`welcome.dart > Widget _buildButton()`

```
Widget _buildButton() {
    return RaisedButton(
        onPressed: () {
            print('Tap!');
        },
        child: Text('Tap Me'),
    );
}
```

## ກົດ Save ແລ້ວລອງກົດປຸ່ມດູ



ເພີ່ມຕົວແປຣ. int \_count = 0;

```
2
3   class WelcomePage extends StatelessWidget {
4
5     int _count = 0;
6
7     @override
8     Widget build(BuildContext context) {
9       return Scaffold(
```

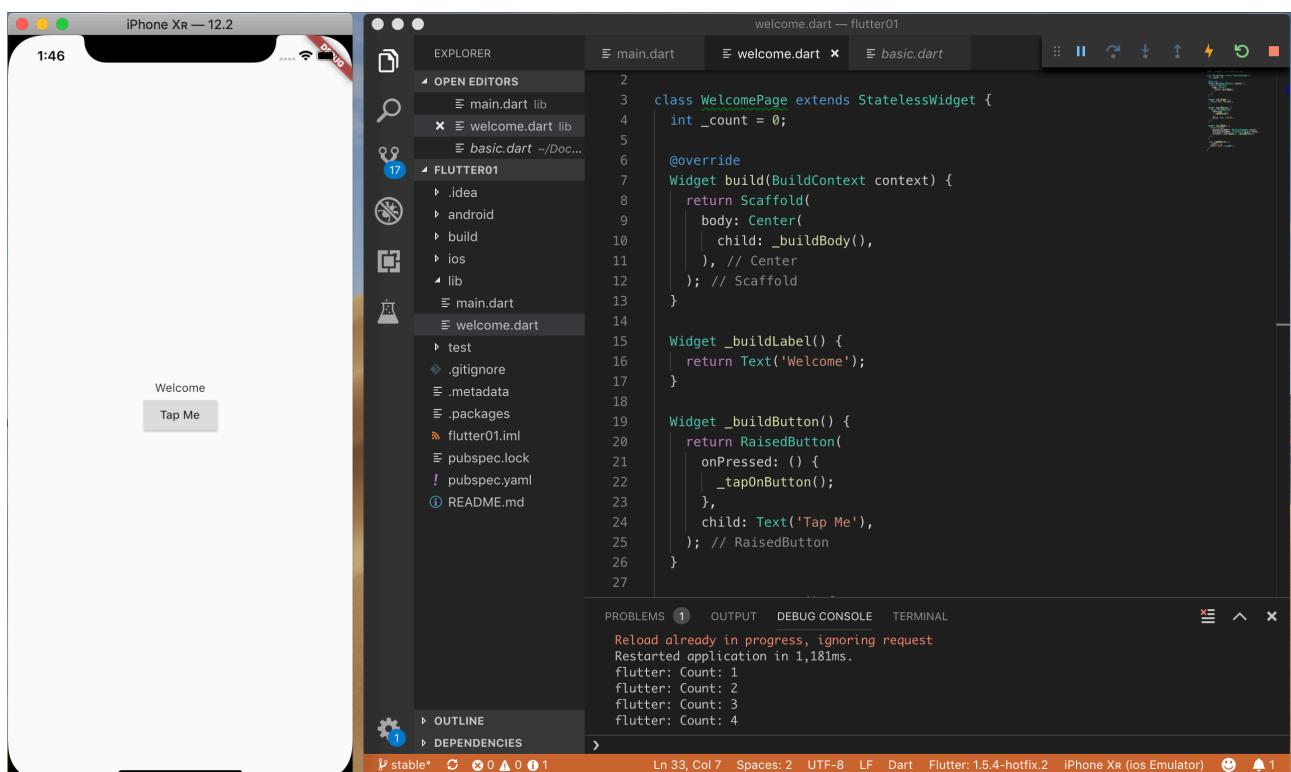
ແລະຝຶກໜັນ \_tapOnButton()

```
void _tapOnButton() {
  _count += 1;
  print("Count: $_count");
}
```

## ແກ້ໄຟກໍຂັນ \_buildButton()

```
Widget _buildButton() {
    return RaisedButton(
        onPressed: () {
            _tapOnButton();
        },
        child: Text('Tap Me'),
    );
}
```

ກົດ ອີໂລດແລ້ວລອງທດສອບອີກຄົງ



## welcome.dart (1/2)

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

class WelcomePage extends StatelessWidget {
    int _count = 0;

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Center(
                child: _buildBody(),
            ),
        );
    }

    Widget _buildLabel() {
        return Text('Welcome');
    }

    Widget _buildButton() {
        return RaisedButton(
            onPressed: () {
                _tapOnButton();
            },
            child: Text('Tap Me'),
        );
    }

    Widget _buildBody() {
        return Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [_buildLabel(), _buildButton()],
        );
    }
}
```

## ต่อ welcome.dart (2/2)

```
void _tapOnButton() {  
    _count += 1;  
    print("Count: $_count");  
}  
}
```

ถึงตรงนี้จะเห็นว่าเมื่อกดปุ่ม `_count` ก็จะเพิ่มค่าที่ละ 1 อย่างที่เราต้องการแล้ว ต่อไปก็การแสดงผล

ถ้าเป็นปกติเราจะทำอะไรย่างนี้

```
Widget myLabel = _buildBody();  
myLabel.text = "Count: $_count";
```

แต่กับ Flutter ทำอย่างนี้ไม่ได้ หากอยากเปลี่ยนแปลงการแสดงผลเราต้องสร้างใหม่ทั้งก้อน กลับไปดูที่ตอนประกาศ class Welcome

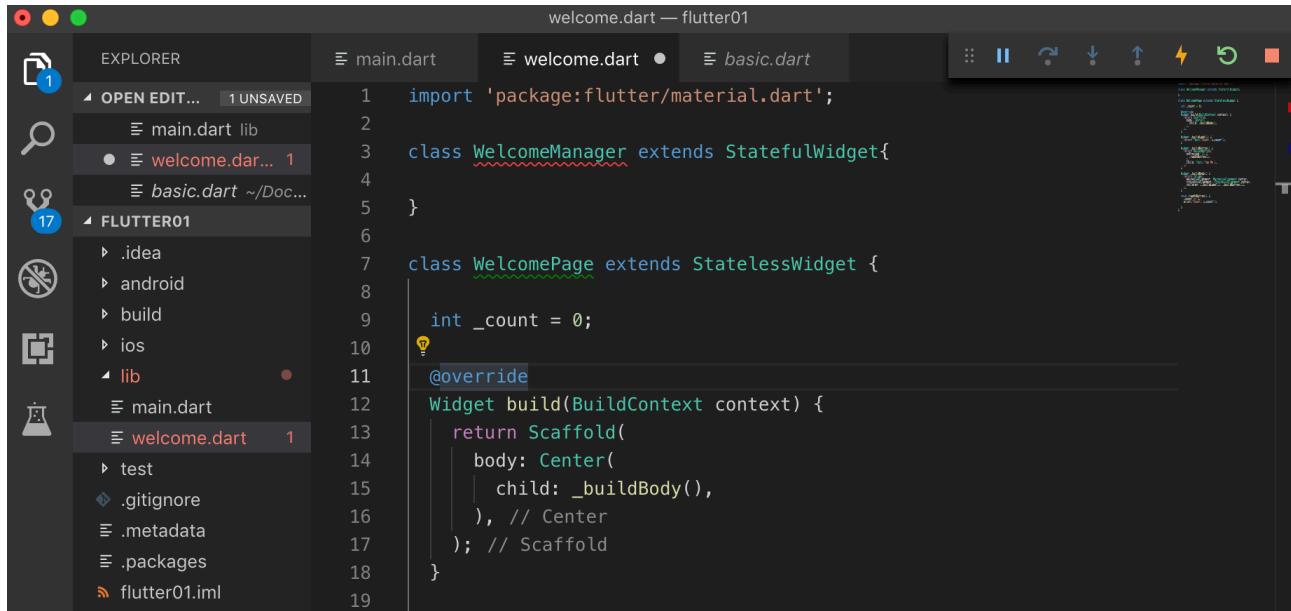
```
2  
3     class WelcomePage extends StatelessWidget {  
4         |
```

‘StatelessWidget’ จะเป็น Widget ชนิดที่ไม่สามารถเปลี่ยนแปลงได้ คือเมื่อถูกสร้างในฟังก์ชัน `build()` และก็จบแค่นั้น

หากต้องการให้สามารถเปลี่ยนแปลงการแสดงผลได้ต้องใช้ ‘StatefulWidget’

ทำการสร้าง class `WelcomeManager` ขึ้นที่ส่วนต้นของไฟล์ `welcome.dart`

## welcome.dart



```
welcome.dart — flutter01
EXPLORER          main.dart    welcome.dart • basic.dart
OPEN EDIT...      1 UNSAVED
main.dart lib
• welcome.dart 1
basic.dart ~/Doc...
FLUTTER01
.idea
android
build
ios
lib
main.dart
welcome.dart 1
test
.gitignore
.metadata
.packages
flutter01.iml
```

```
1 import 'package:flutter/material.dart';
2
3 class WelcomeManager extends StatefulWidget{
4 }
5
6
7 class WelcomePage extends StatelessWidget {
8   int _count = 0;
9   @override
10  Widget build(BuildContext context) {
11    return Scaffold(
12      body: Center(
13        child: _buildBody(),
14      ), // Center
15    ); // Scaffold
16  }
17 }
18
19 }
```

เพิ่มฟังก์ชัน createState ที่ class WelcomeManager ฟังก์ชันนี้จะถูกเรียกเมื่อมีการสั่งให้ StatefulWidget โหลดการแสดงผลใหม่

```
2
3 class WelcomeManager extends StatefulWidget{
4   @override
5   State<StatefulWidget> createState() {
6     // TODO: implement createState
7     return null;
8   }
9 }
```

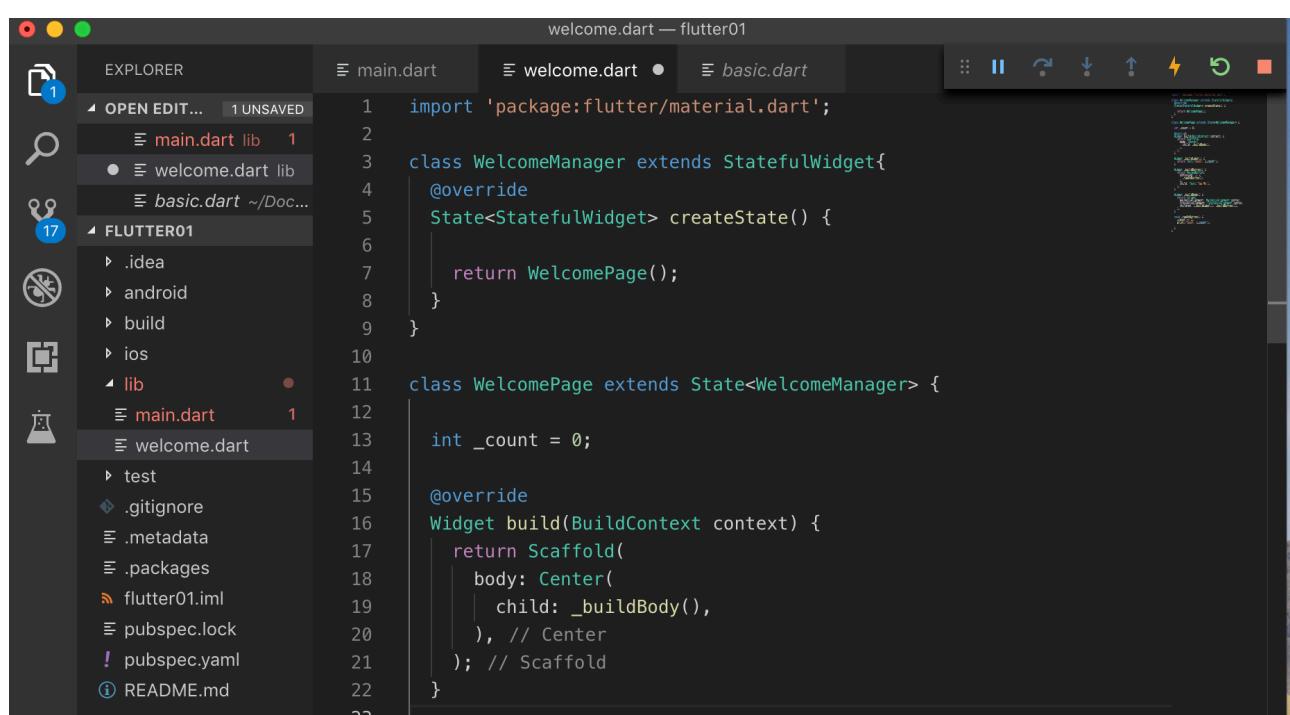
แก้ไข class WelcomePage เป็น State หนึ่งของ class WelcomeManager

```
10
11 class WelcomePage extends State<WelcomeManager> {
12 }
```

กลับไปที่ class WelcomeManager แก้ฟังก์ชัน createState()

welcome.dart > class WelcomeManager > createState

```
class WelcomeManager extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    return WelcomePage();
  }
}
```



แก้ฟังก์ชัน \_buildLabel() และ \_tapOnButton()

welcome.dart > class WelcomePage > \_buildLabel()

```
Widget _buildLabel() {
  return Text("Count: ${_count}");
}
```

welcome.dart > class WelcomePage > \_tapOnButton()

```
void _tapOnButton() {  
    setState(() {  
        _count += 1;  
    });  
}
```

การทำงานของ โปรแกรมคือ เมื่อกดปุ่มจะเรียกฟังก์ชัน ‘setState()’ เพื่อสั่งให้ WelcomeManager โหลดการแสดงผลใหม่ สังเกตว่าค่าที่เราต้องการเปลี่ยนจะอยู่ในฟังก์ชัน ‘setState()’ หลังจากทำงานใน ‘setState()’ จะแล้วก็จะทำการเรียกฟังก์ชัน ‘createState()’ ต่อไป

กลับไปแก้ main.dart

main.dart > class MyApp

```
class MyApp extends StatelessWidget {  
    @override  
    Widget build(BuildContext context) {  
        return MaterialApp(  
            home: WelcomeManager(),  
        );  
    }  
}
```

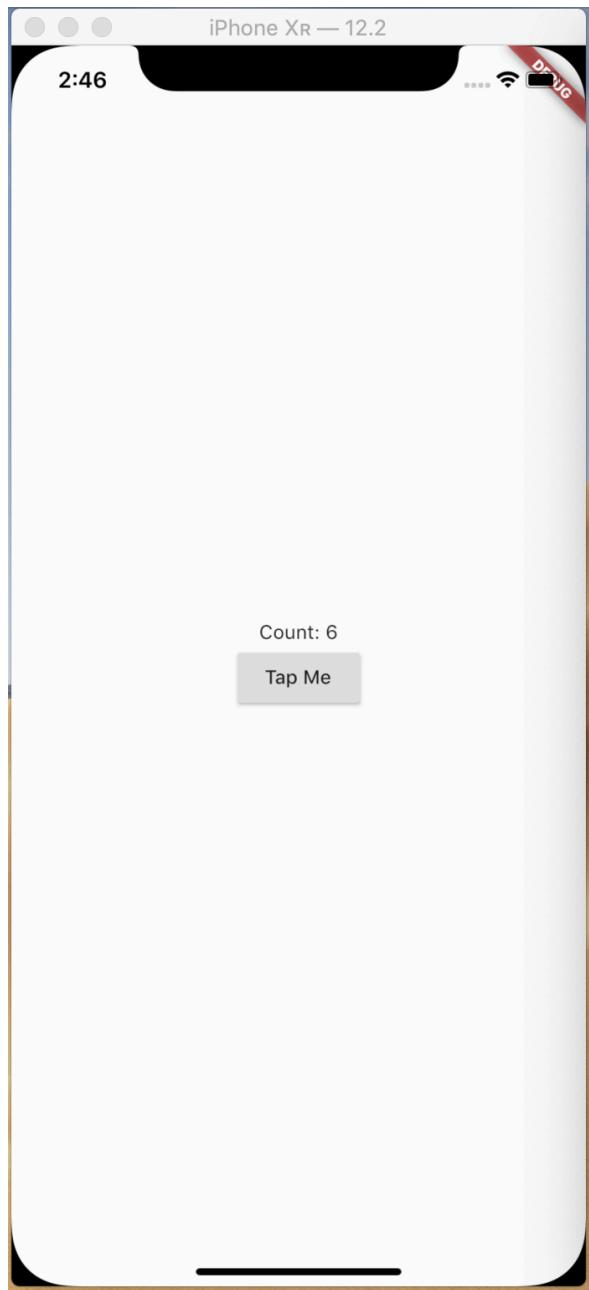
main.dart — flutter01

```
1 import 'package:flutter/material.dart';
2 import './welcome.dart';
3
4 void main() => runApp(MyApp());
5
6 class MyApp extends StatelessWidget {
7   @override
8   Widget build(BuildContext context) {
9     return MaterialApp(
10       home: WelcomeManager(),
11     ); // MaterialApp
12   }
13 }
```

welcome.dart — flutter01

```
1 import 'package:flutter/material.dart';
2
3 class WelcomeManager extends StatefulWidget {
4   @override
5   State<StatefulWidget> createState() {
6     return WelcomePage();
7   }
8 }
9
10 class WelcomePage extends State<WelcomeManager> {
11   int _count = 0;
12
13   @override
14   Widget build(BuildContext context) {
15     return Scaffold(
16       body: Center(
17         child: _buildBody(),
18       ), // Center
19     ); // Scaffold
20   }
21
22   Widget _buildLabel() {
23     return Text("Count: ${_count}");
24   }
25
26   Widget _buildButton() {
27     return RaisedButton(
28       onPressed: () {
29         _tapOnButton();
30       },
31       child: Text('Tap Me'),
32     ); // RaisedButton
33   }
34
35   Widget _buildBody() {
36     return Column(
37       mainAxisAlignment: MainAxisAlignment.center,
38       crossAxisAlignment: CrossAxisAlignment.center,
39       children: [_buildLabel(), _buildButton()],
40     );
41   }
42
43   void _tapOnButton() {
44     setState(() {
45       _count += 1;
46     });
47   }
48 }
```

ทดลอง debug ดูก็จะได้ผลอย่างที่เราต้องการแล้ว.



ถึงตรงนี้เราก็สามารถเข้าใจคร่าวๆถึง การทำงานของ Flutter app การใช้ StatelessWidget และ StatefulWidget เป็นอย่างตื้นแล้ว