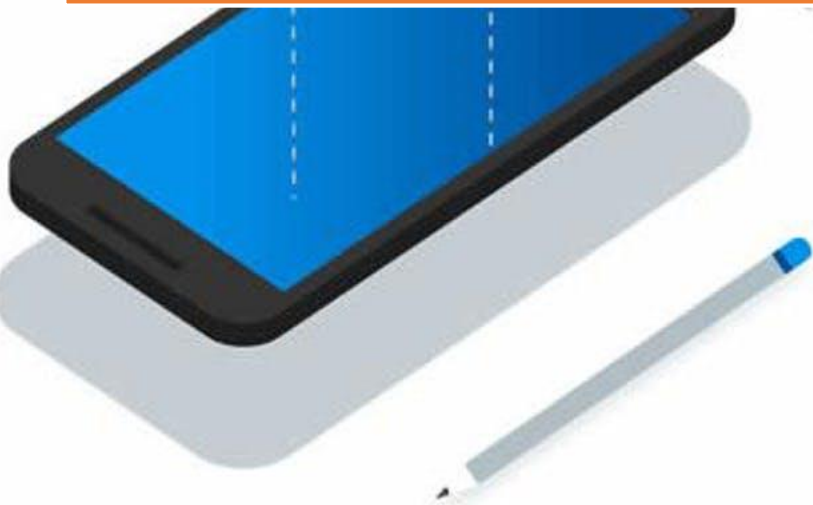




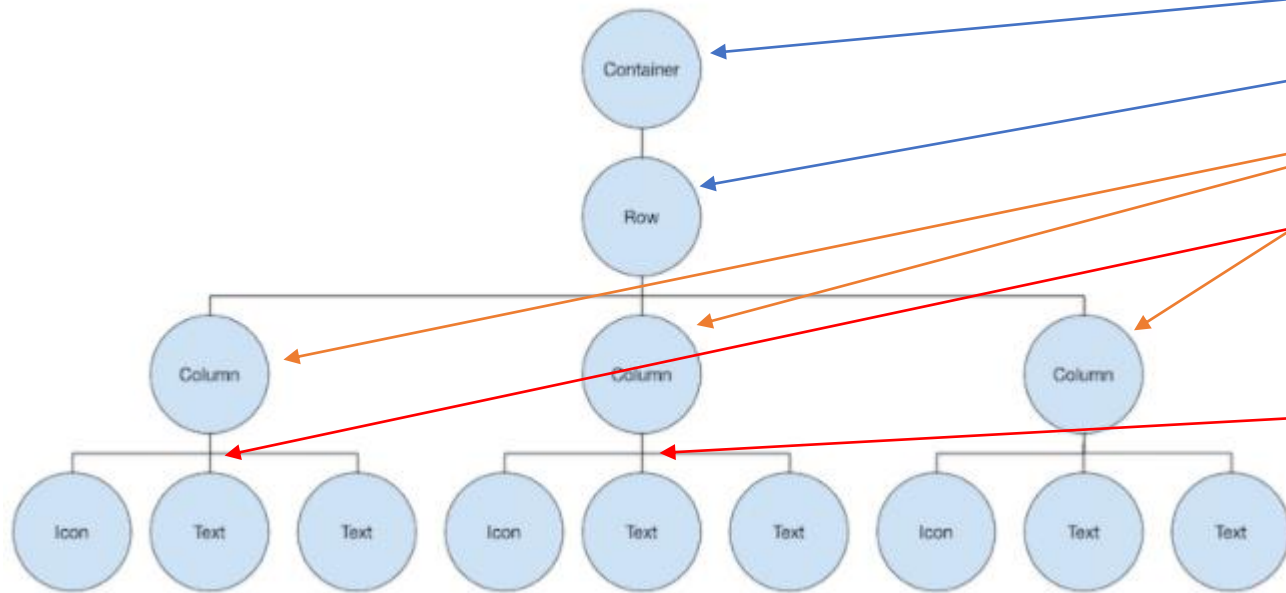
Flutter

Week 2

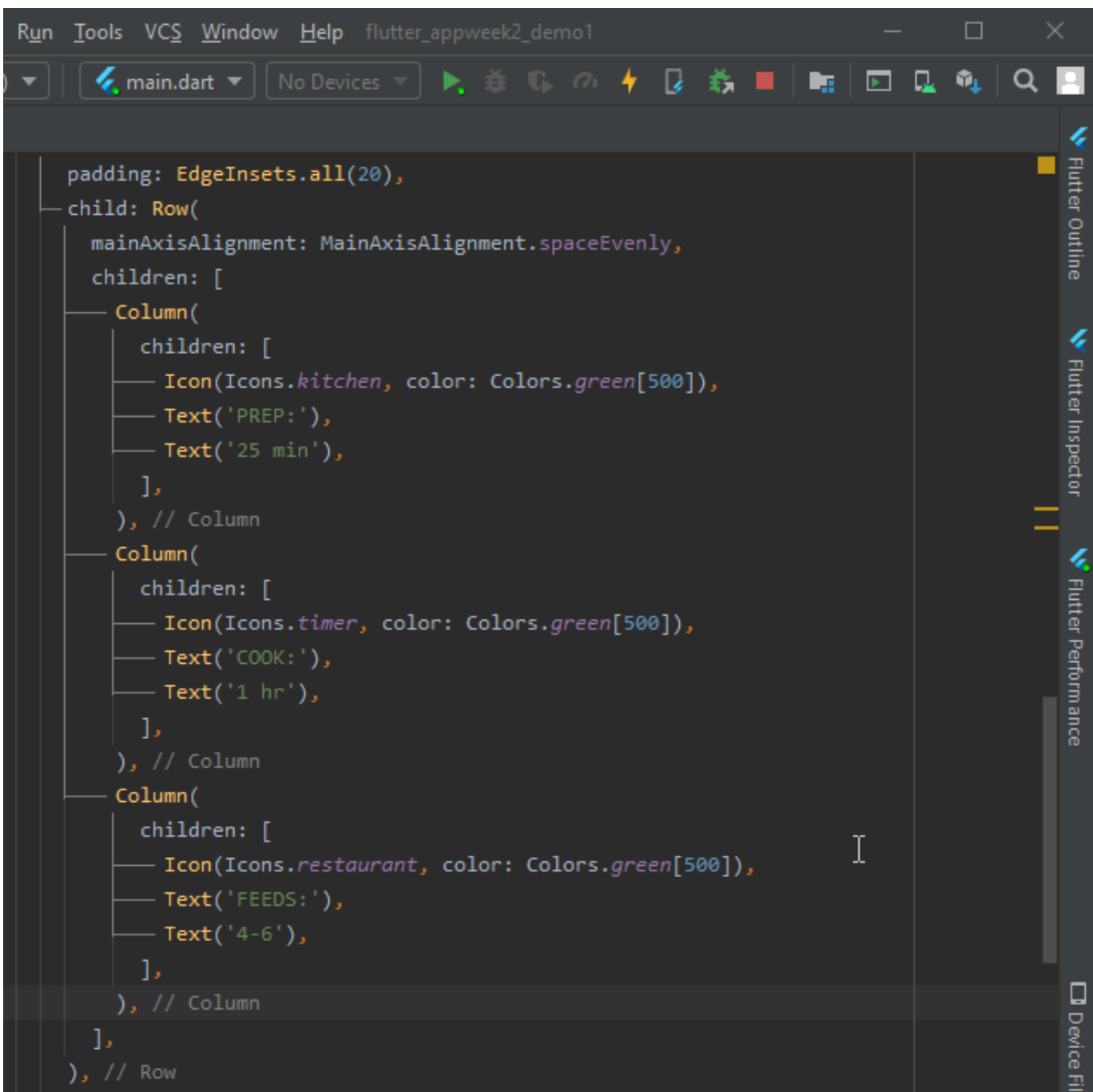
4123006 การพัฒนาโปรแกรมประยุกต์บน
อุปกรณ์เคลื่อนที่ 3 (2-2-5)



Child & Children

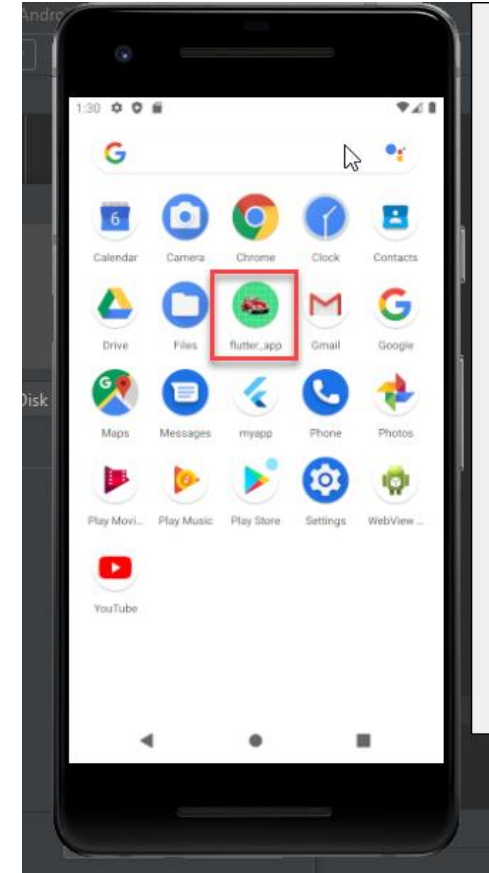
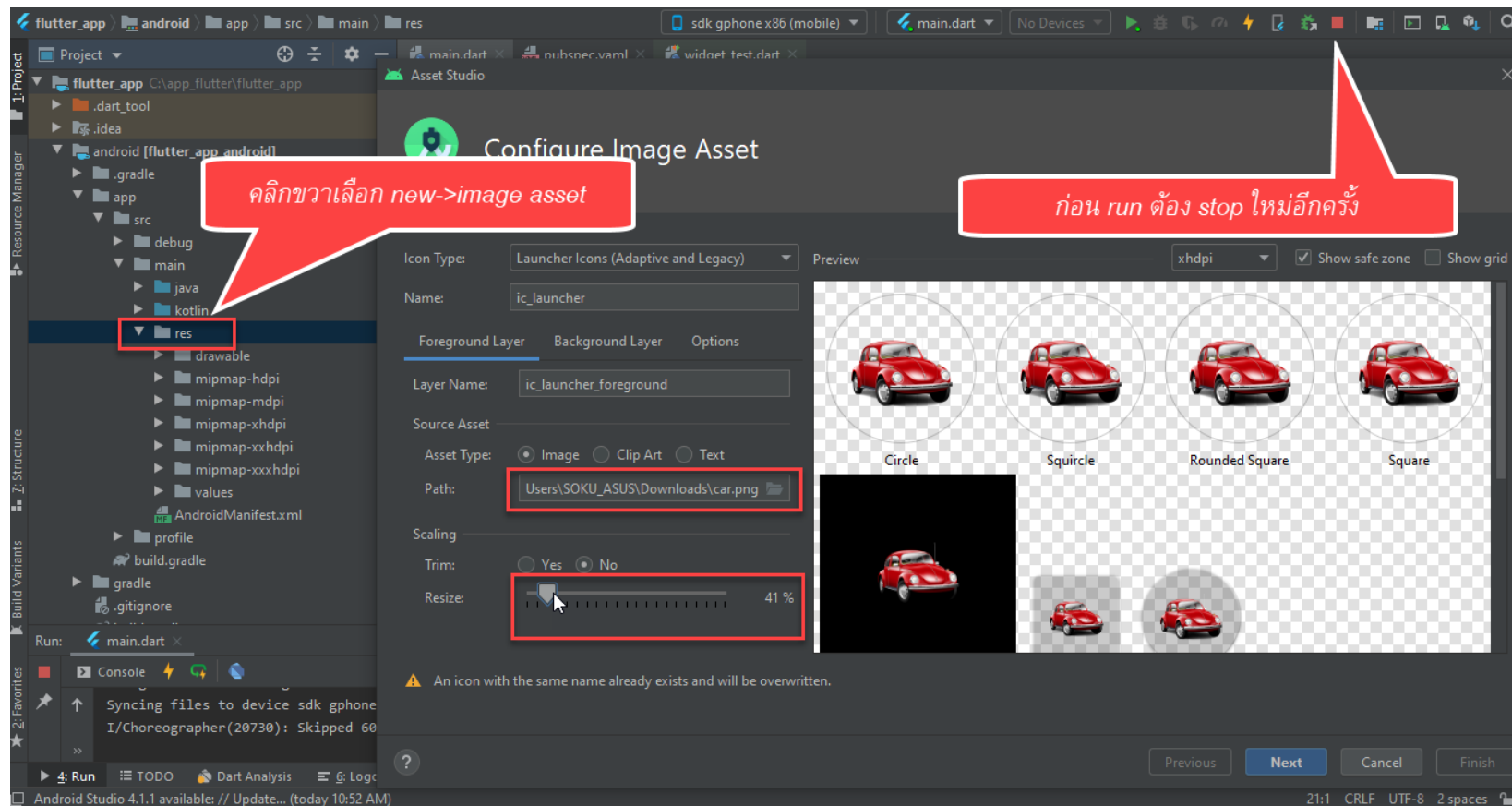


```
// DefaultTextStyle.merge() allows you to create a default text
// style that is inherited by its child and all subsequent children.
final iconList = DefaultTextStyle.merge(
  style: descTextStyle,
  child: Container(
    padding: EdgeInsets.all(20),
    child: Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      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'),
          ],
        ),
      ],
    ),
  ),
);
```



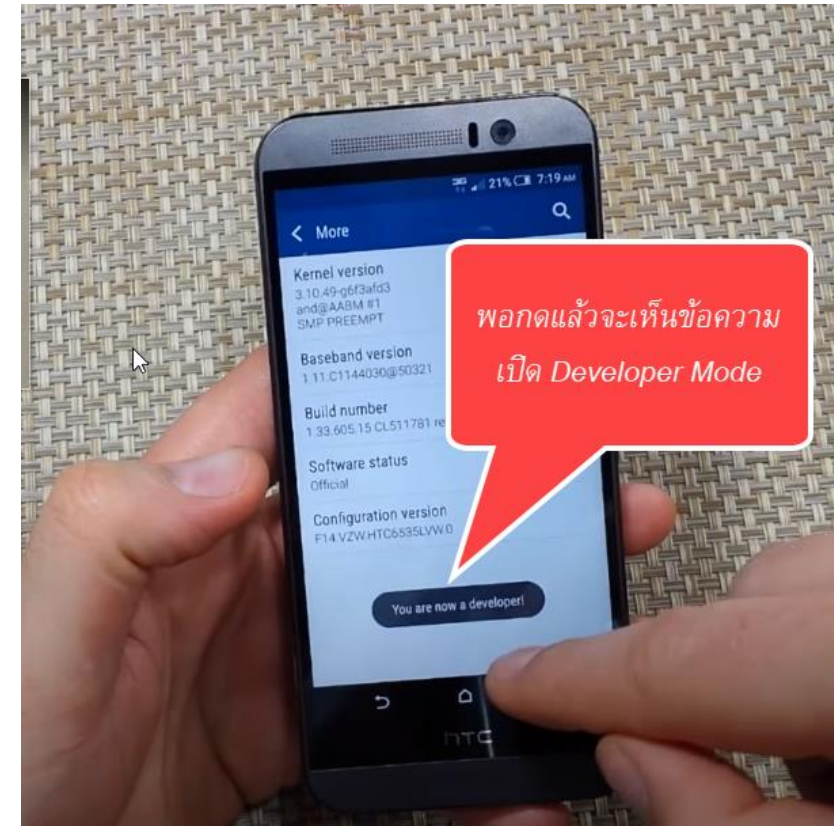
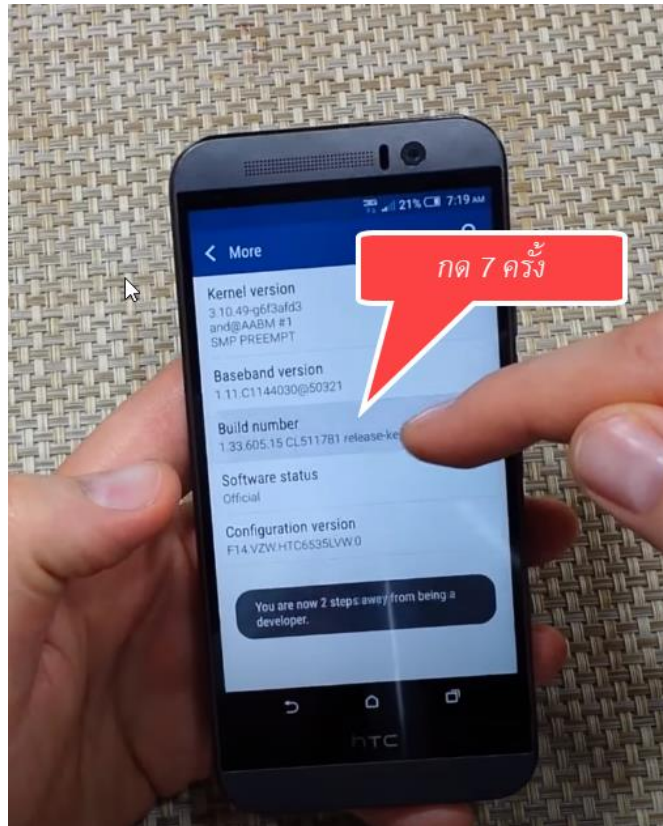
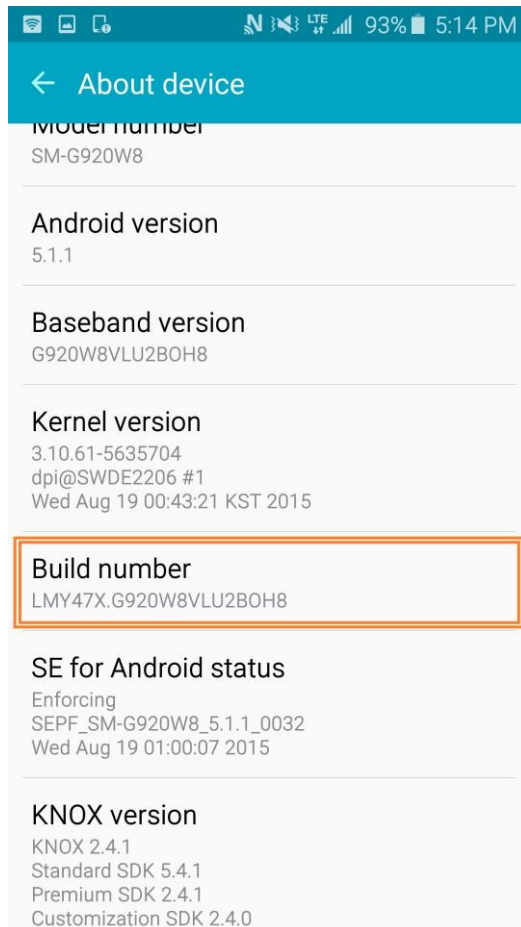
การสร้าง icon บน ios และ android

- หารูป icon ที่ต้องการหรือไปดาวน์โหลดมาจากเว็บ iconarchive นามสกุล png
- ใน android studio เลือก android->app->src->main->res แล้วคลิกขวาที่ res เลือก new เลือก image assets
- ให้เลือกไฟล์ icon ที่จะสร้าง แล้วปรับขนาดรูปภาพจากช่องด้านขวา แล้วก็กด next -> finish แล้วรอ run ดู



การ Deploy app สู่ physical android device

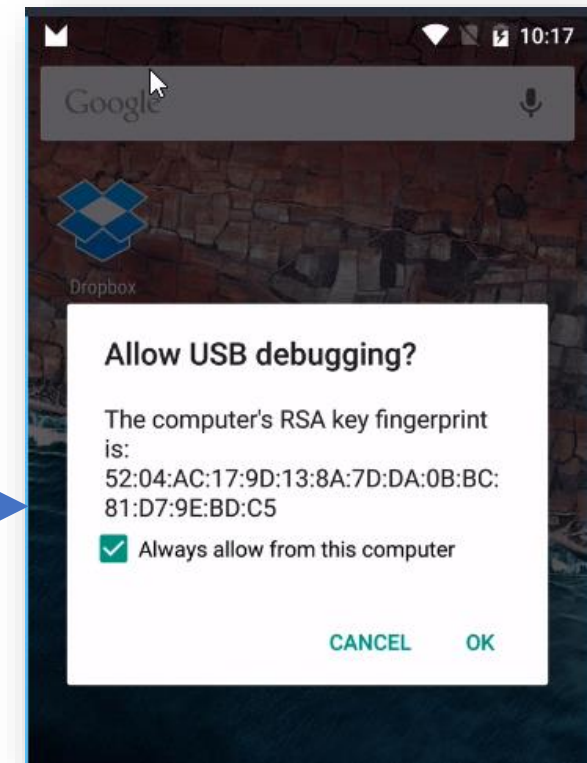
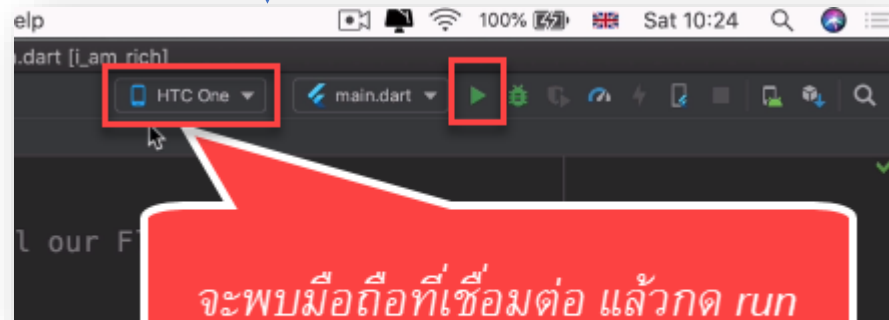
- ต้องนำเครื่องมือถือ android เข้าไปใน setting ไปหา build number กด 7 ครั้งเพื่อเปิด Developer Mode เพื่อให้อุปกรณ์ android สามารถเสียบสาย usb แล้ว debug app ได้ เพื่อจะได้ลงแอปที่ทำจากโปรแกรมเราได้



การ Deploy app สู่ physical android device (ต่อ)

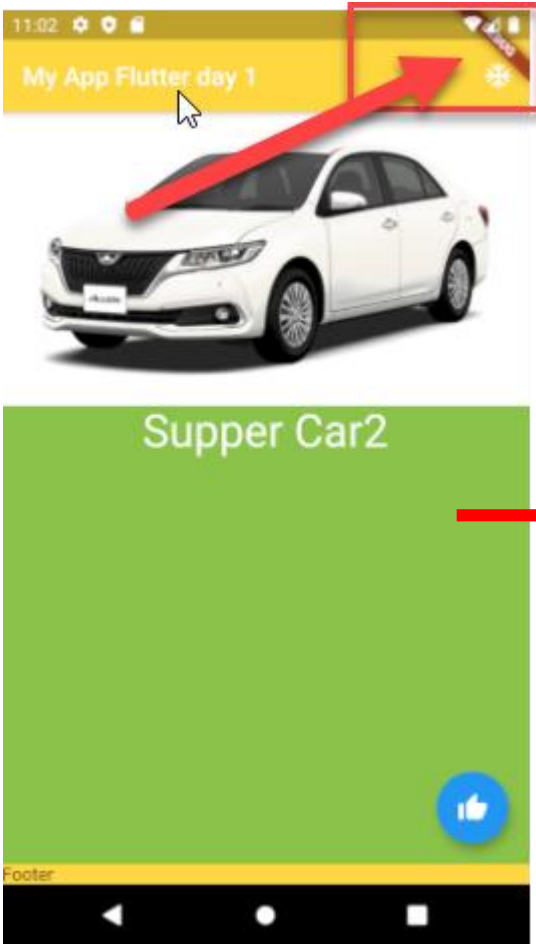
Steps

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 ใน หน้าจอมือถือจะแสดงข้อความให้กด **allow usb debugging**

การเอา Icon Debug Banner ออกจากโค้ด Flutter



ให้ใส่ debugShowCheckedModeBanner: false, ก่อน home

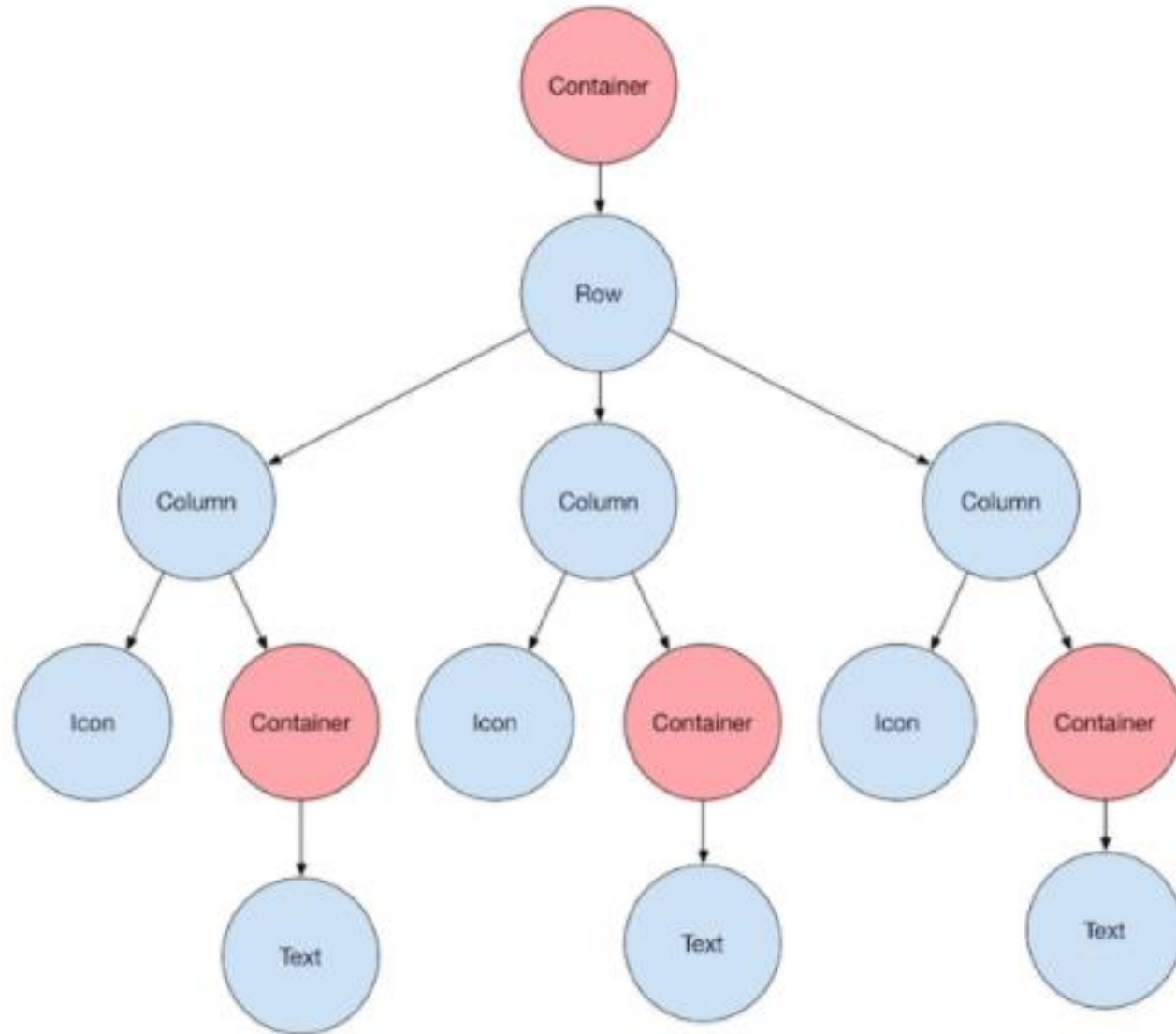
```
import 'package:flutter/material.dart';

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

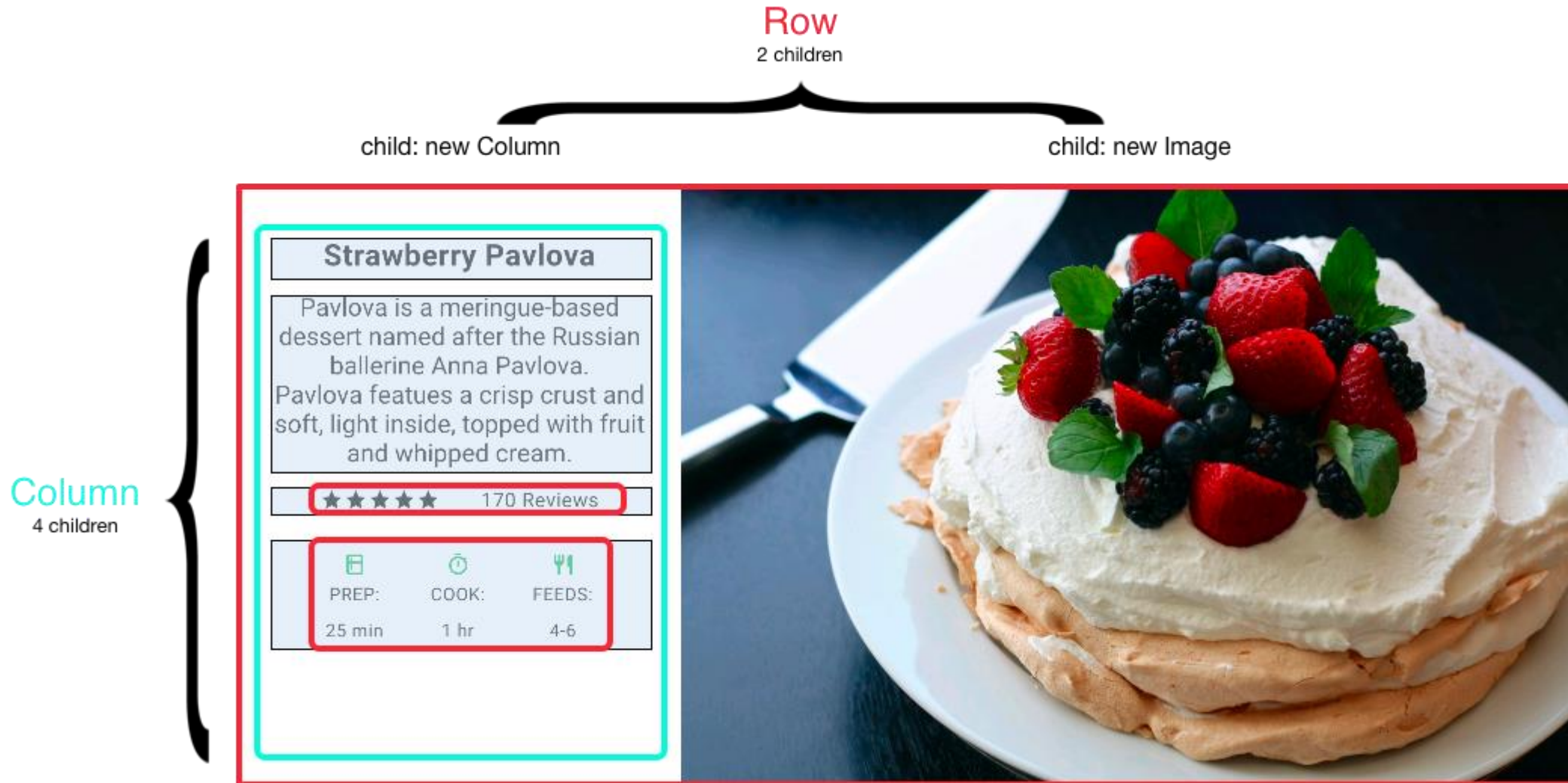
class MyApp extends StatelessWidget {
  Widget build(BuildContext context) {
    var hello='สวัสดี';
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home:Scaffold(
        appBar: AppBar(
          backgroundColor: Colors.amberAccent,
          title:Text('My App Flutter day 1'),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



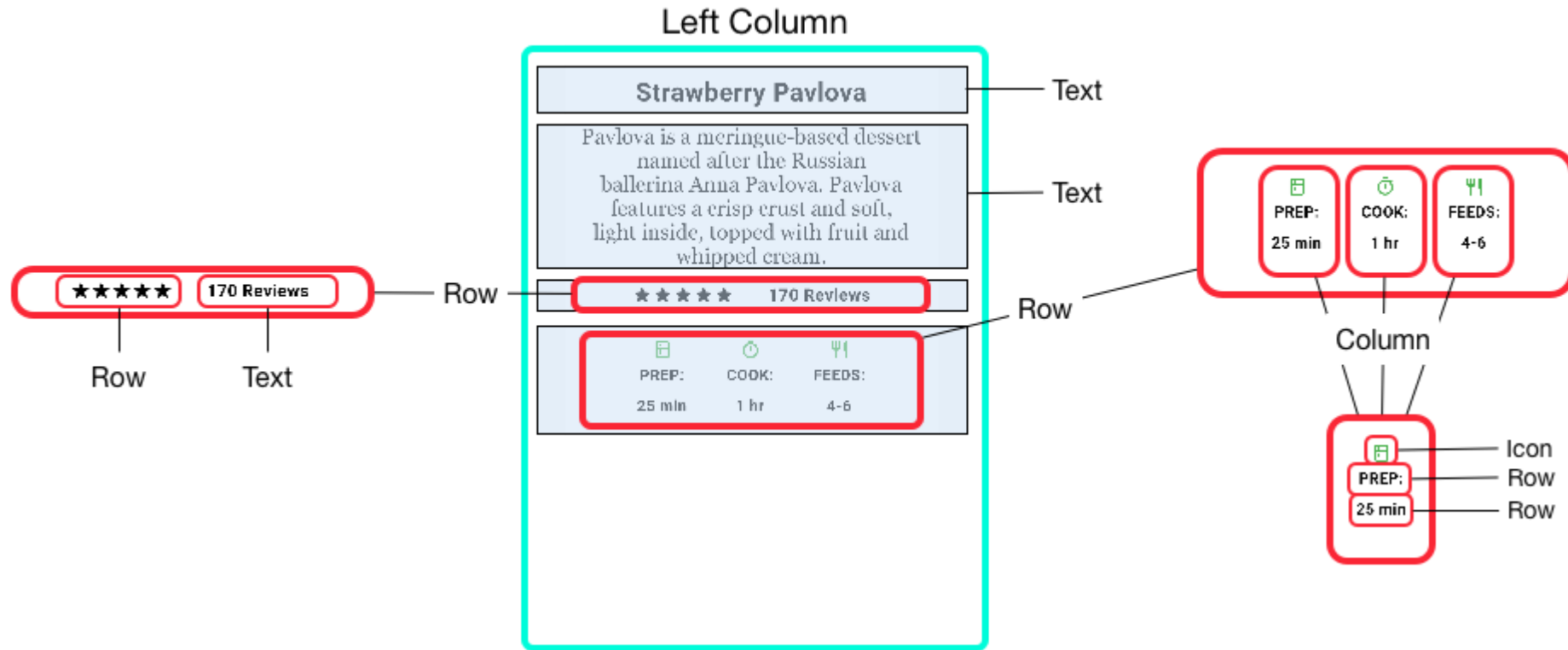
Layouts in Flutter



Layouts in Flutter Row & Column

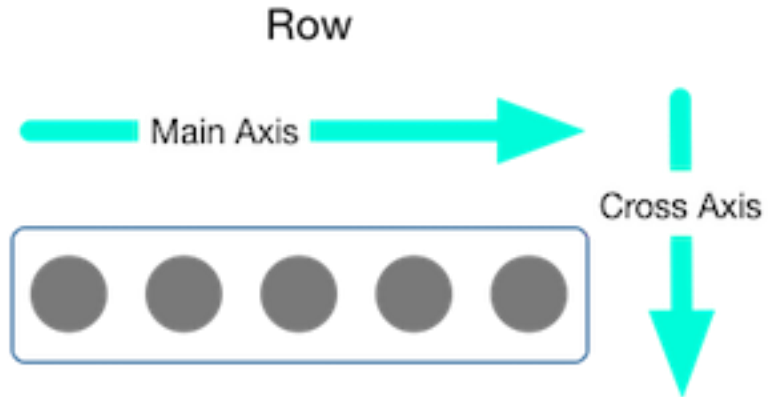


Layouts in Flutter Row & Column



- จากภาพจะสังเกตว่า column และ row อาจจะซ้อนอยู่ข้างในของกันและกันได้ ให้จำง่ายๆ ว่าคอลัมน์เรียงบนลงล่าง ส่วน row เรียงจากซ้ายไปขวา

การจัด Aligned Widget



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

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

App source: [row_column](#)



App source: [row_column](#)



Layout

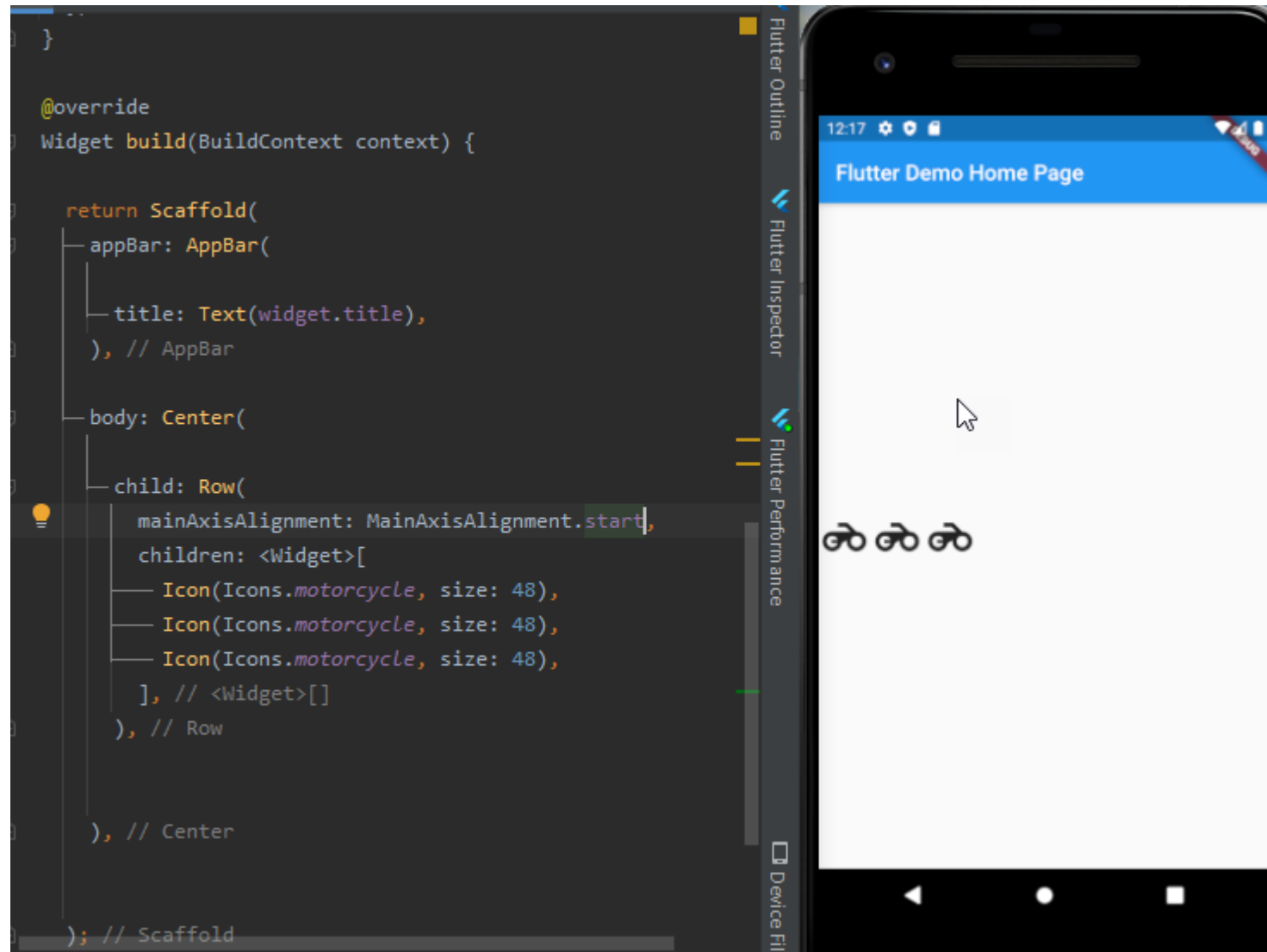


MainAxisAlignment

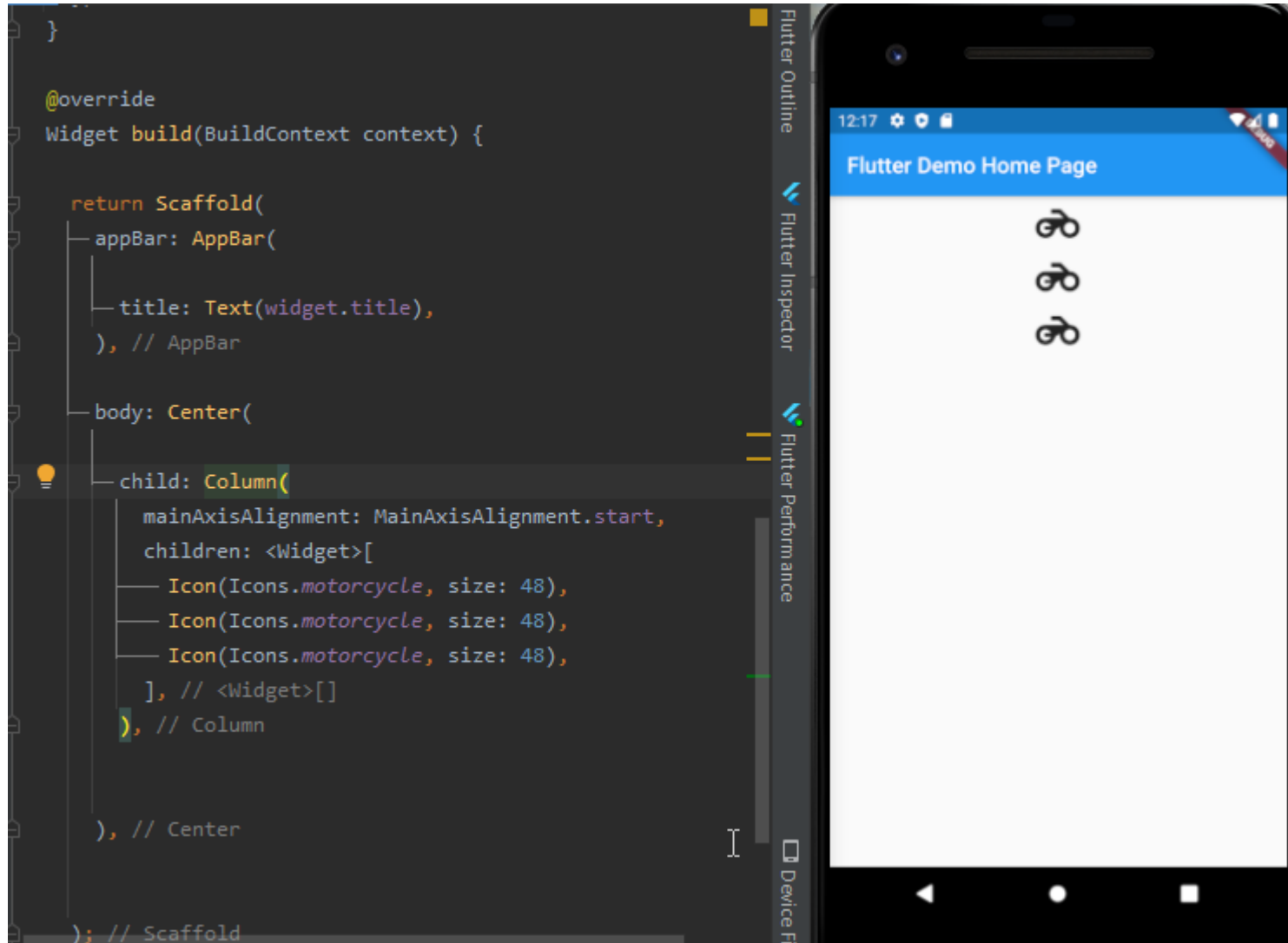
การจัดในแกนหลัก เช่น ถ้าใช้ Row คือ การจัดในแนวนอน, ถ้าใช้ Column คือ การจัดในแนวตั้ง

นอกจาก Main แล้วจะมี **CrossAxisAlignment** คือแนวนอนตัด แกน X

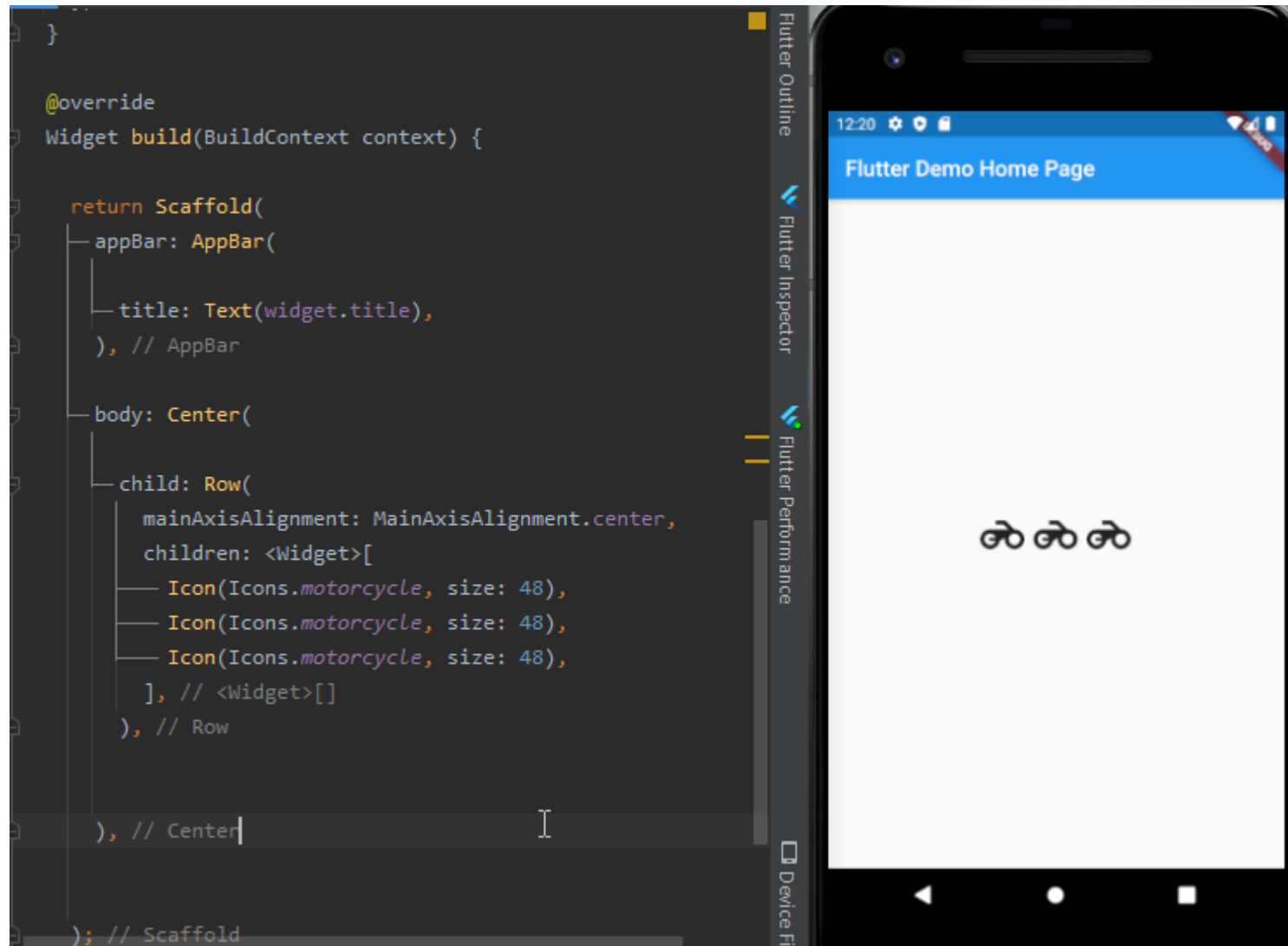
Example Row & MainAxisAlignment.start



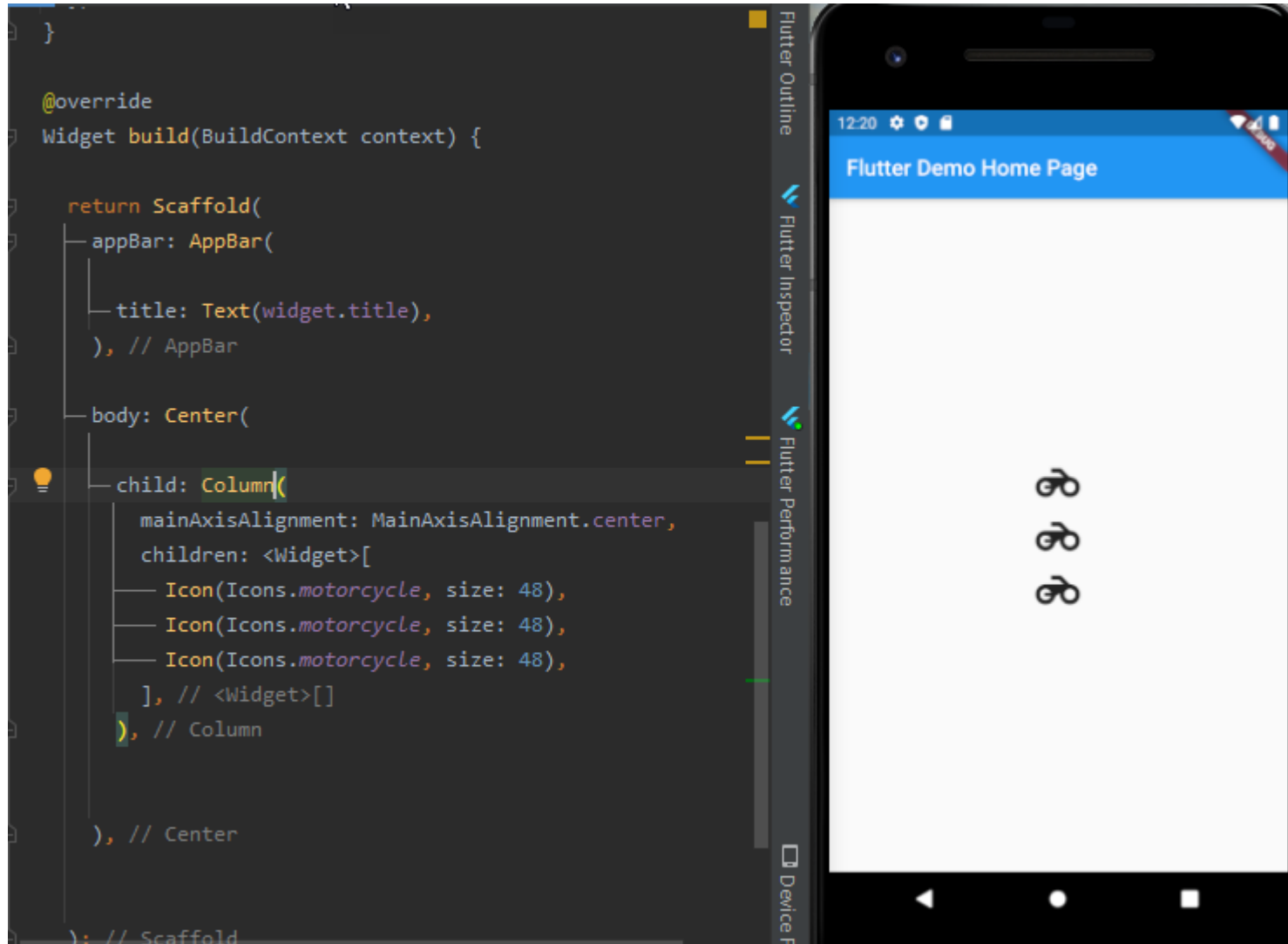
Example Column & MainAxisAlignment.start



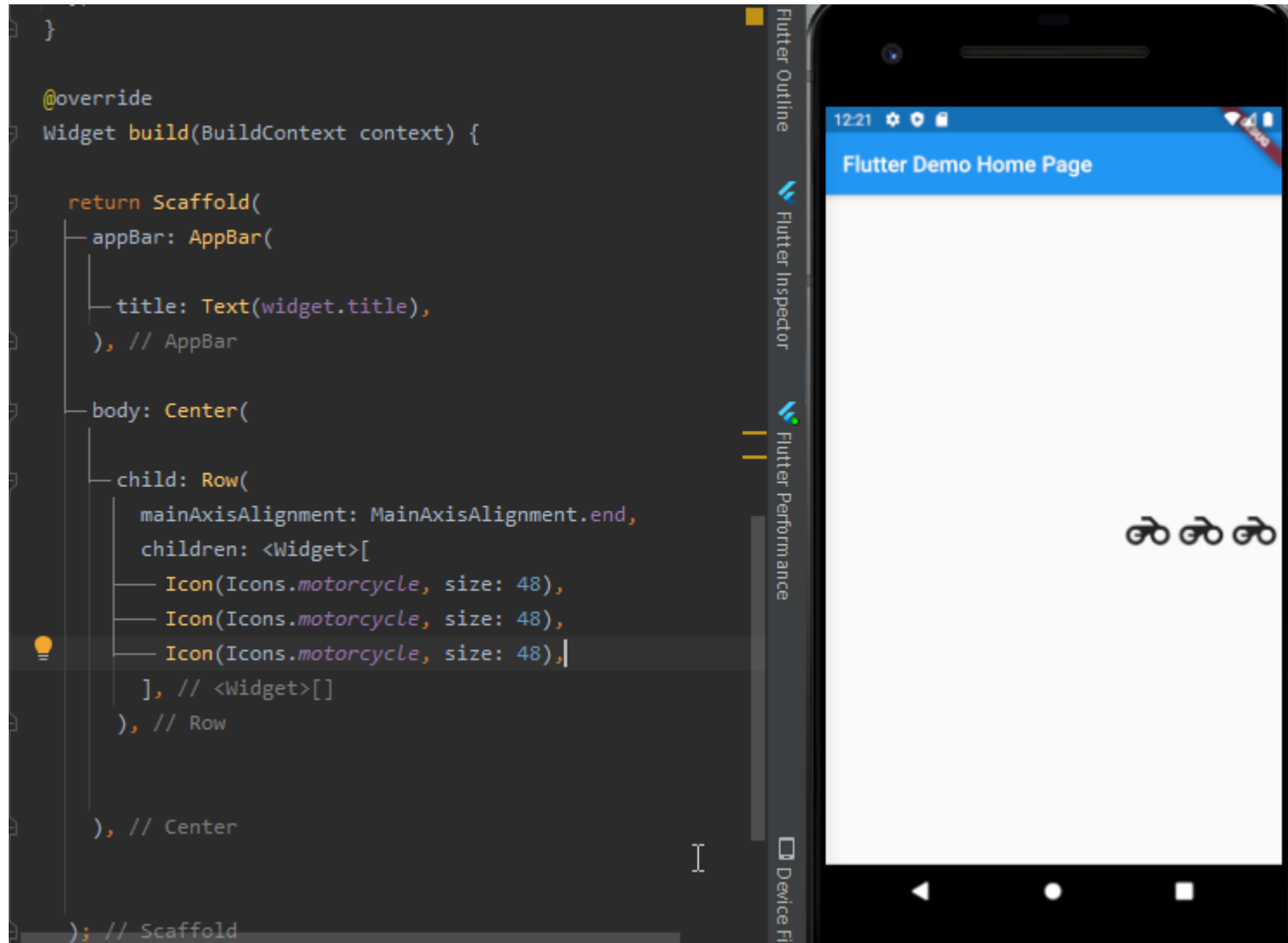
Example Row & MainAxisAlignment.center



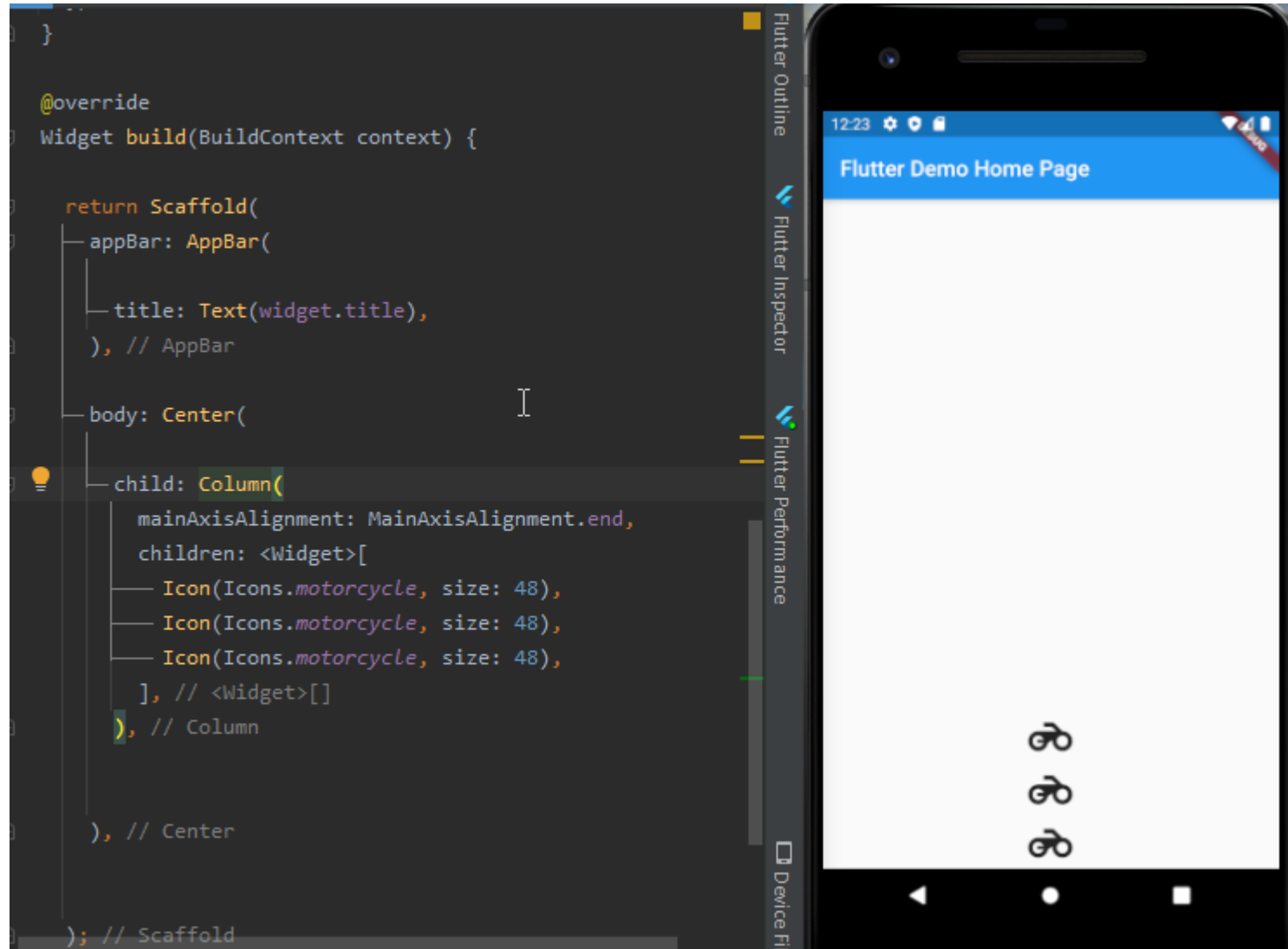
Example Column & MainAxisAlignment.center



Example Row & MainAxisAlignment.end

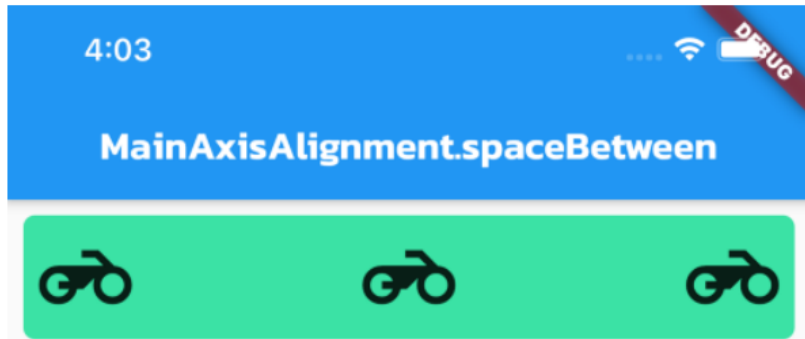


Example Column & MainAxisAlignment.end

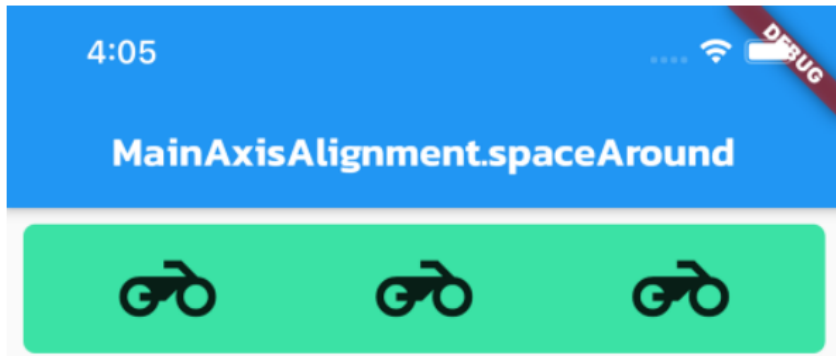


MainAxisAlignment อื่นๆ

- mainAxisAlignment: MainAxisAlignment.spaceBetween, เว้นระยะระหว่างกัน

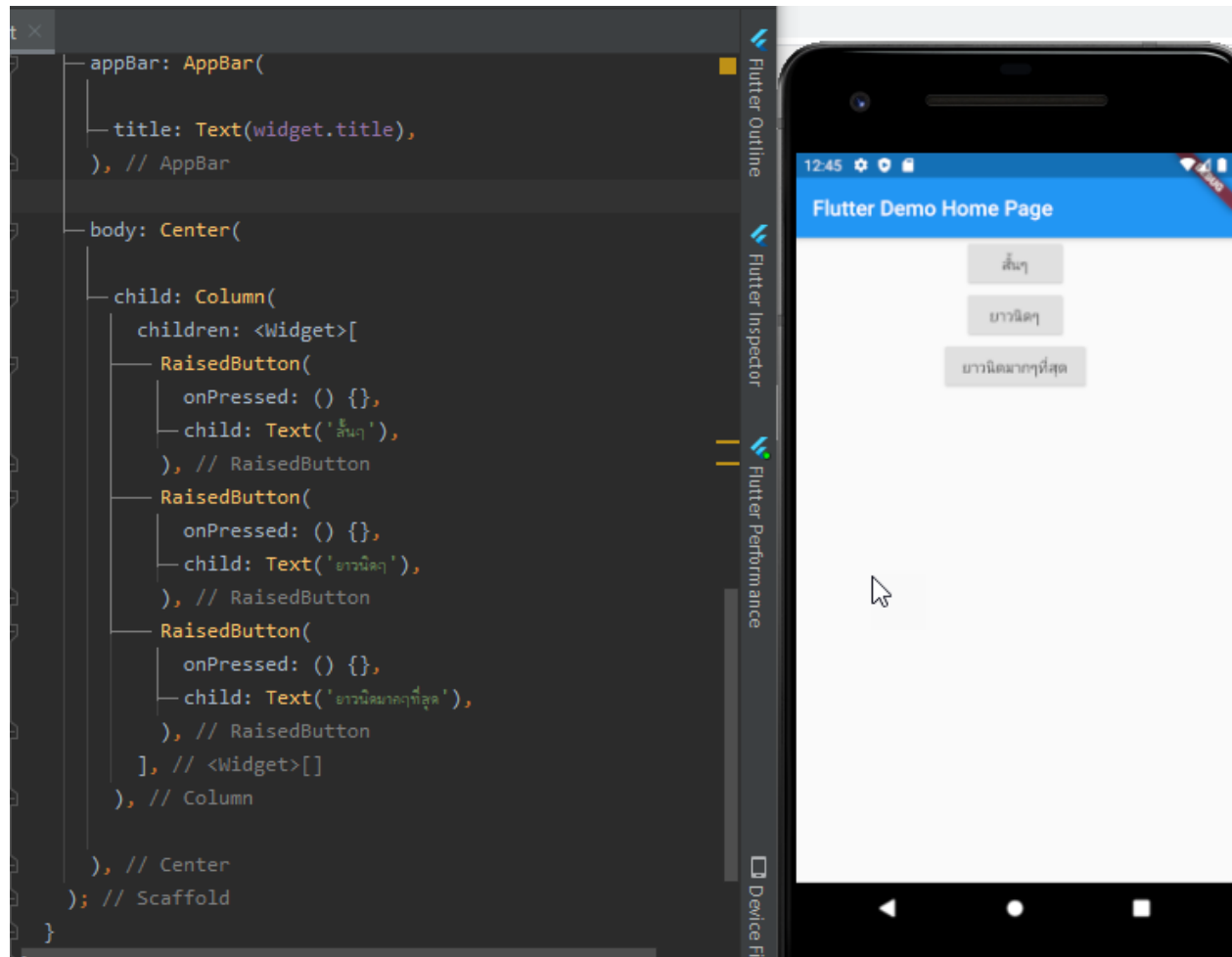


- mainAxisAlignment: MainAxisAlignment.spaceAround, เว้นระยะระหว่างกันโดยมี space รอบๆ เท่าๆ กัน



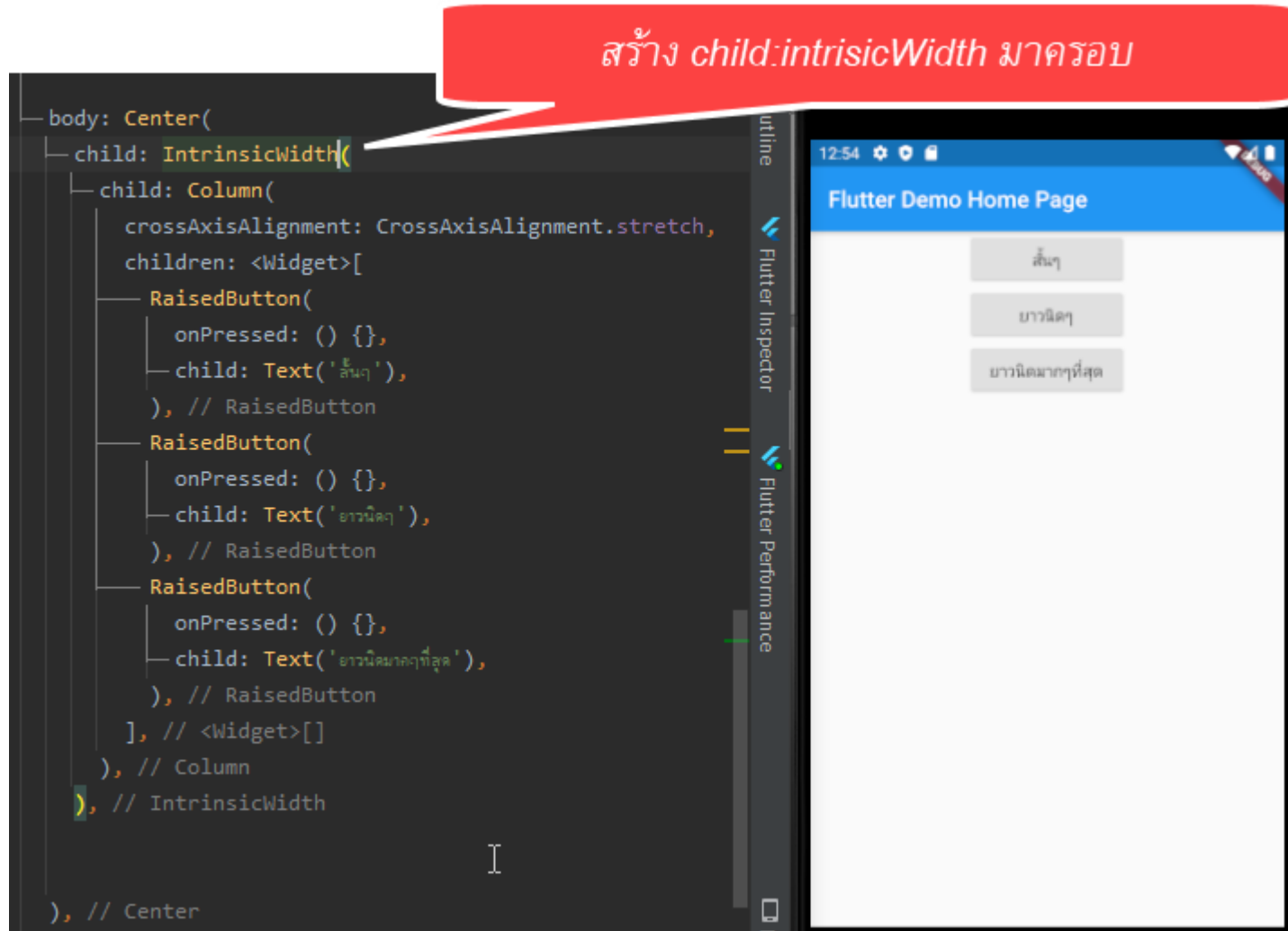
IntrinsicWidth and IntrinsicHeight

- Intrinsic แปลว่า แท้จริง, ที่อยู่ภายใน ในกรณีที่ต้องการให้ความกว้าง หรือความสูง นั้น กว้างเท่ากับ widget ที่กว้างที่สุดใน Column หรือสูงเท่ากับ widget ที่สูงที่สุดใน Row เราไม่ต้องยุ่งยากไปใช้วิธีอื่นๆ โดยทั่วไปถ้าเราไม่ใช่ Intrinsic จะเป็นดังภาพด้านล่าง

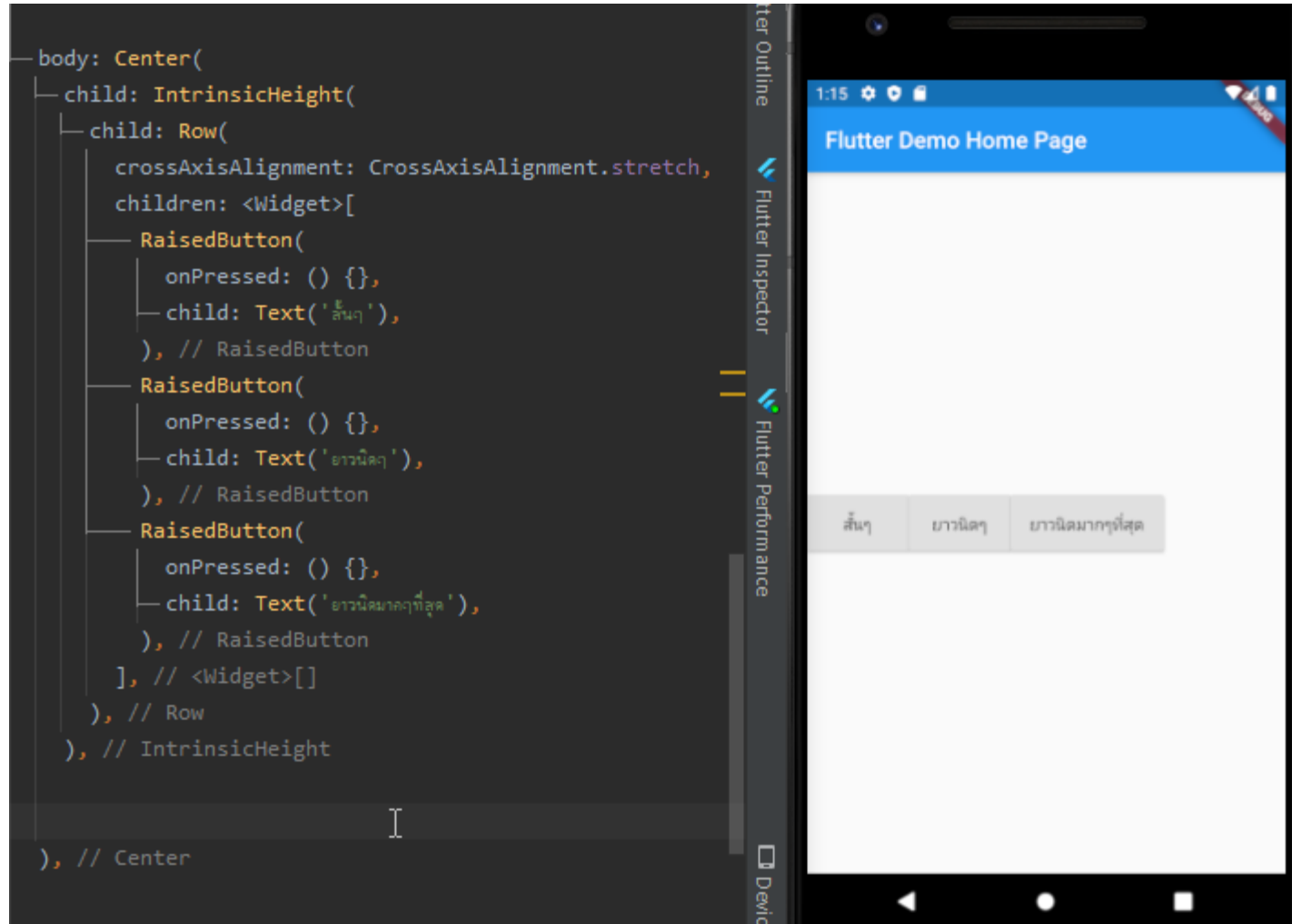


IntrinsicWidth and IntrinsicHeight

- พอใช้ IntrinsicWidth จะได้แบบภาพด้านล่าง

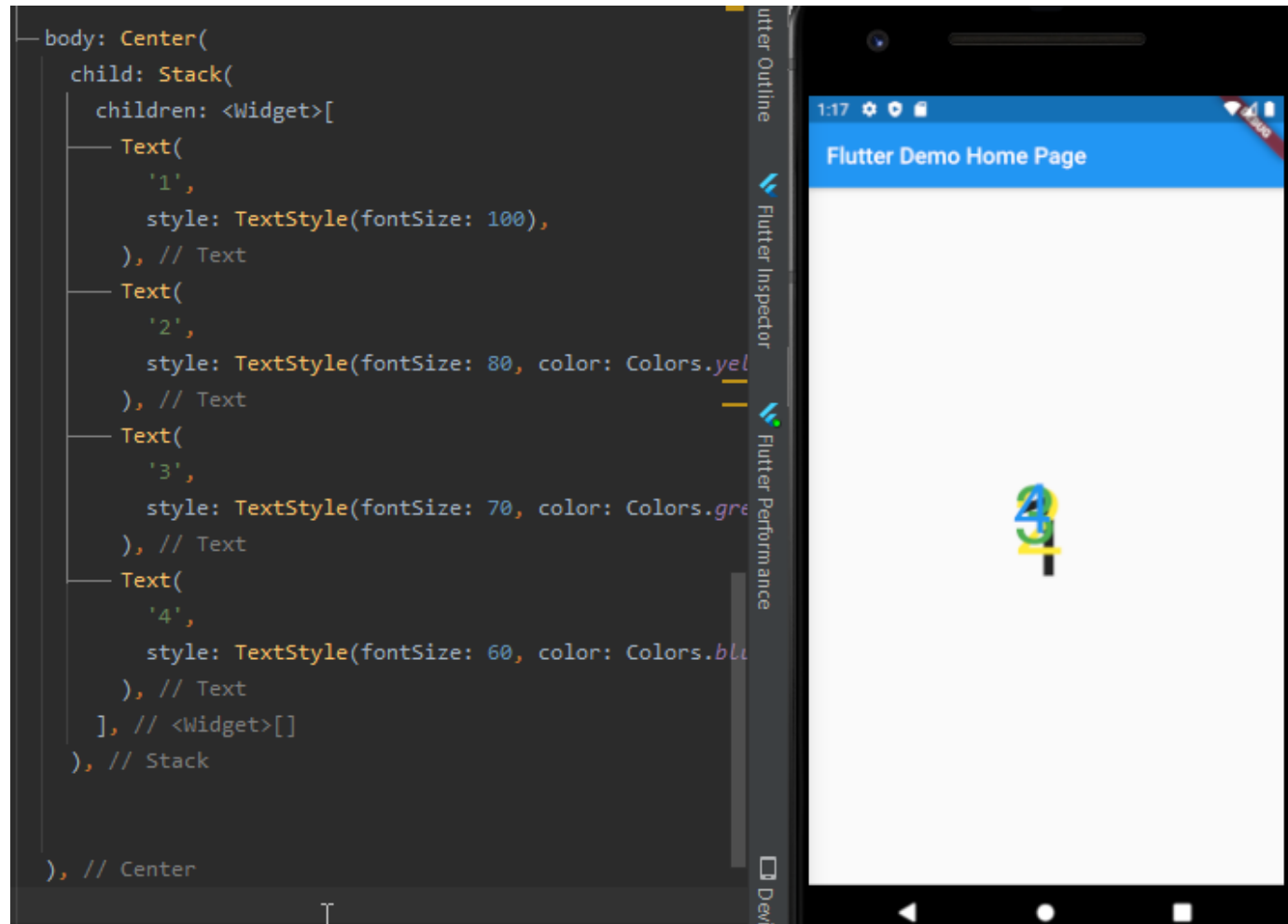


IntrinsicHeight

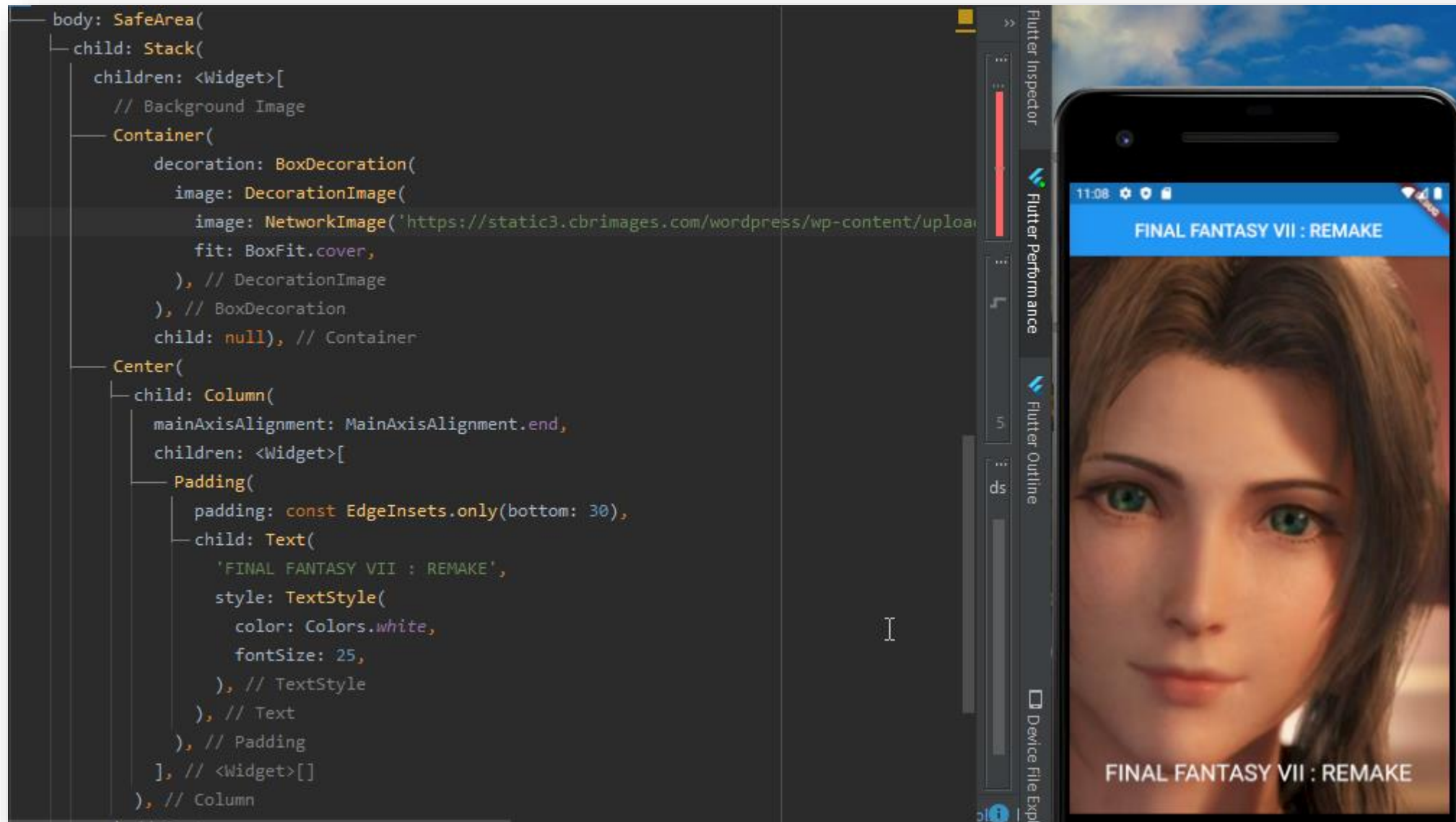


Stack

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

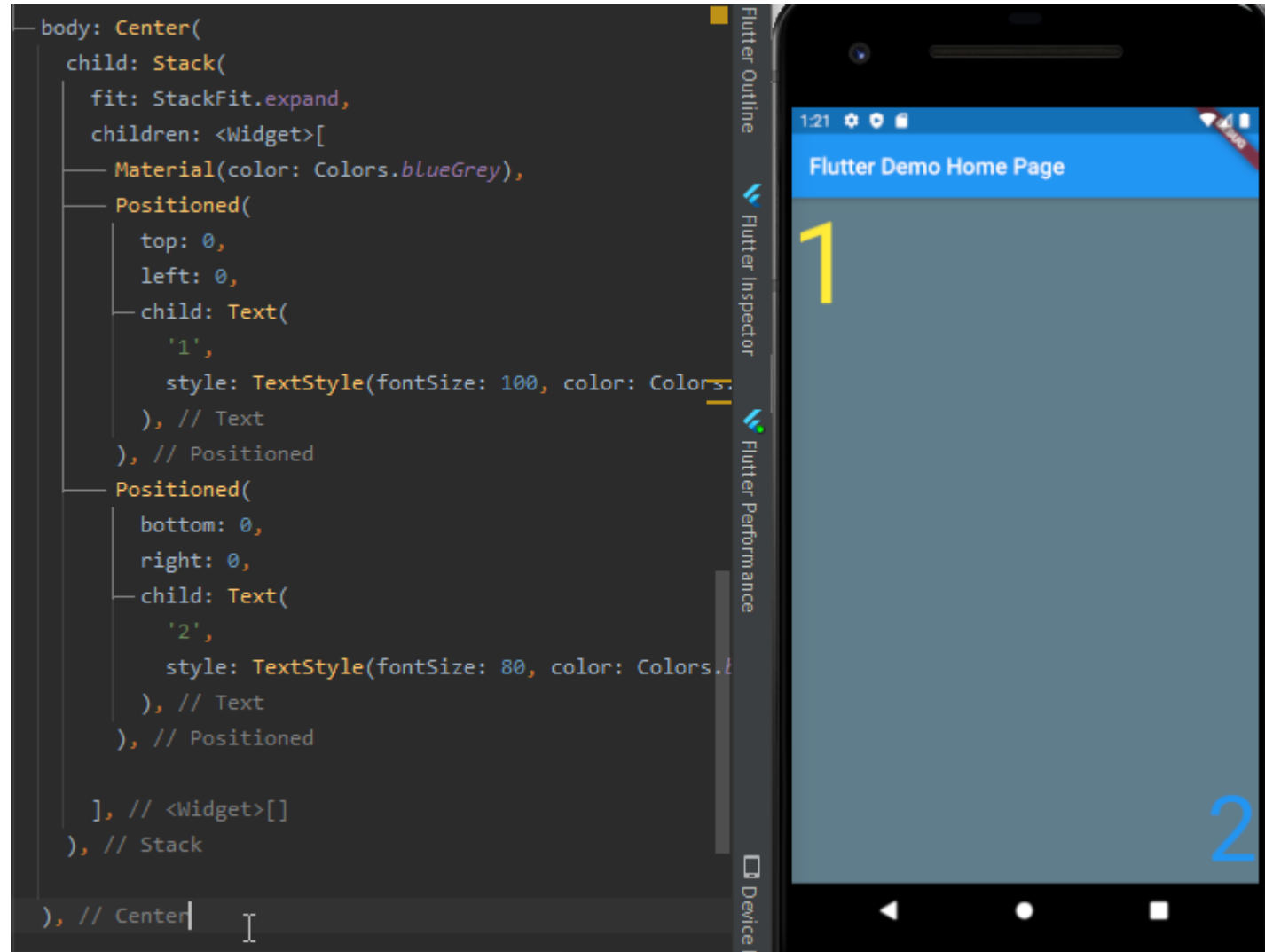


Stack เอาจุฬารูปภาพ ซ่อน Image ได้



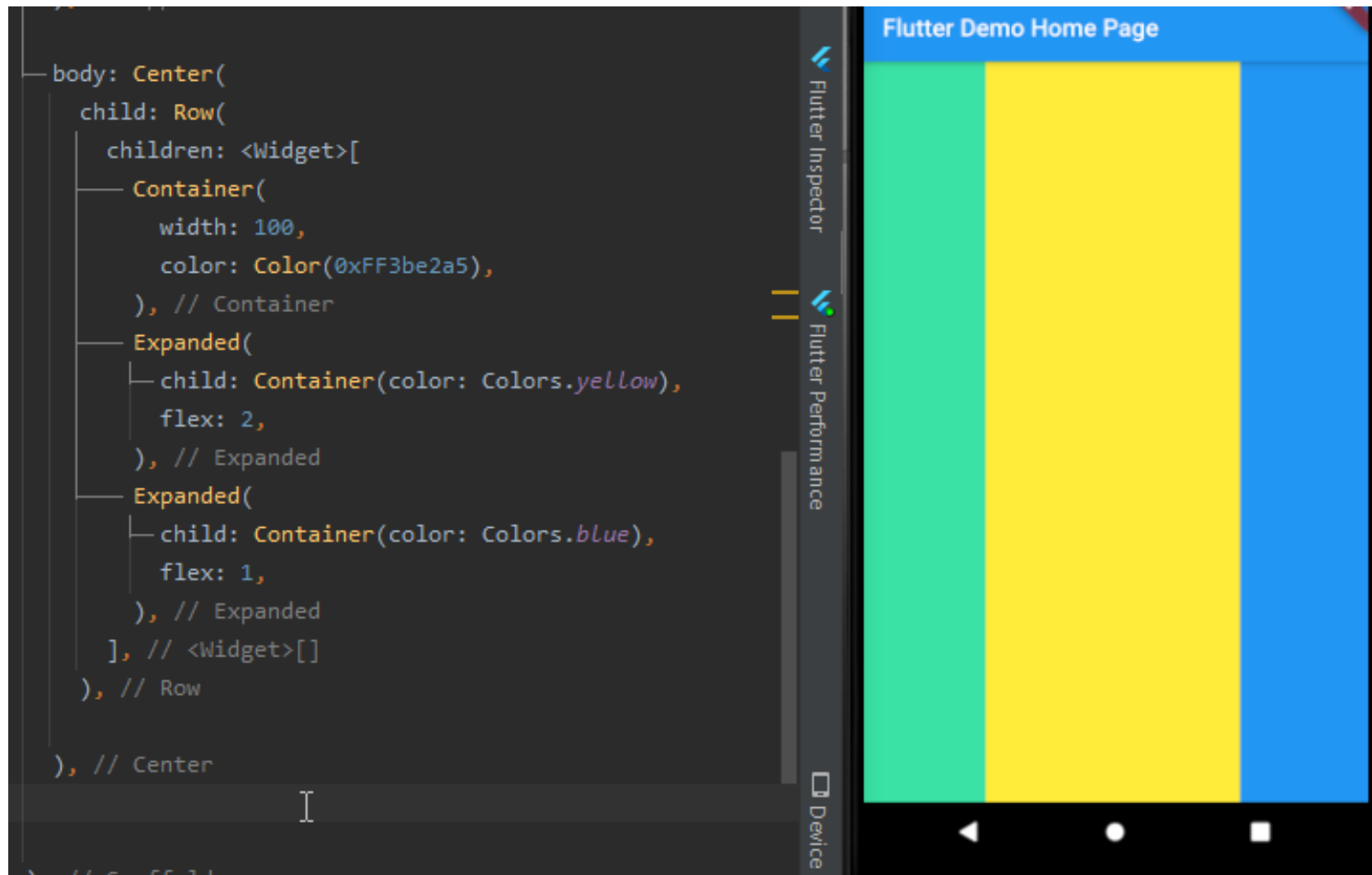
Stack

- สำหรับกรณีที่ต้องการจัดตำแหน่งสามารถใช้ Positioned ในการวางตำแหน่งตามที่ต้องการได้เลย โดยพื้นที่วางนั้นขึ้นอยู่กับขนาดของ widget แม่



Expanded

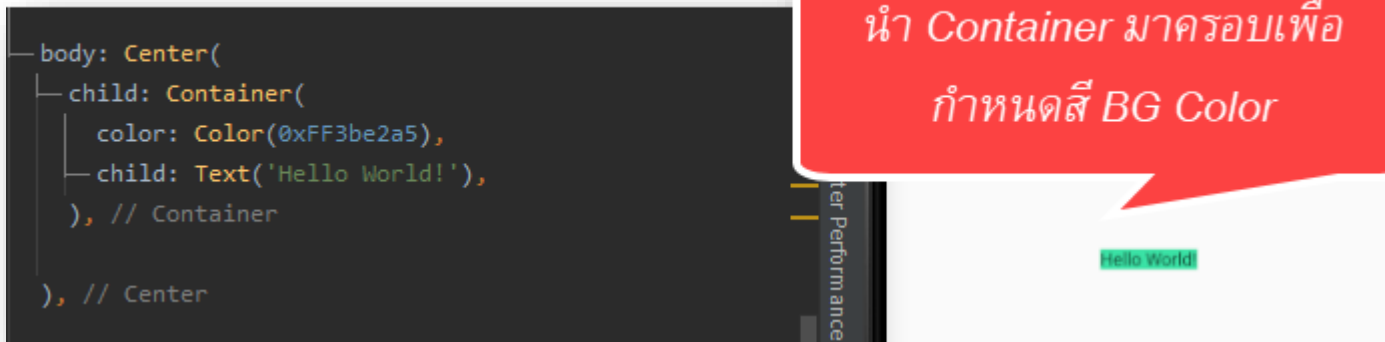
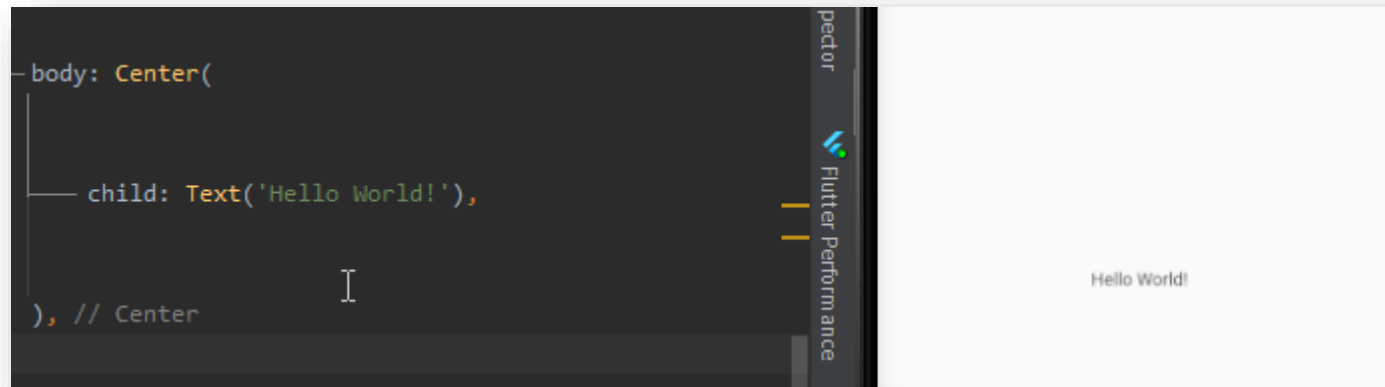
- Expanded จะใช้สำหรับกรณีที่ต้องการที่จะใช้พื้นที่ที่เหลือทั้งหมดของ widget แม่ หาก widget แม่ นั้น มี Expanded มากกว่า 1 ตัวจะต้องทำการใส่สัดส่วนให้มันผ่านค่า flex เพิ่มเติมเข้าไปด้วย



Container

Container ช่วยในการวาง layout ได้ง่ายขึ้น

กรณีที่ไม่มีการกำหนดขนาดความกว้างและความสูง ขนาดของ Container จะขึ้นอยู่กับ widget ลูกที่ใส่มา

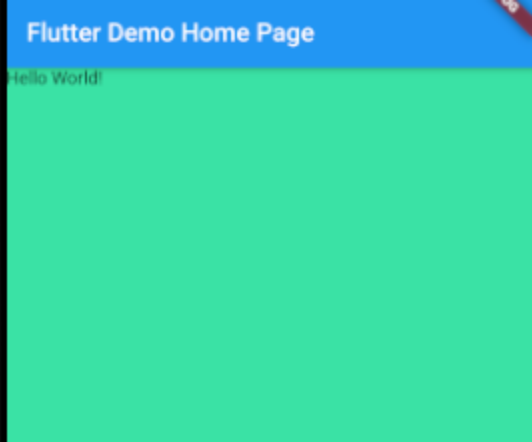


นำ Container มาครอบเพื่อ
กำหนดสี BG Color

double.infinity

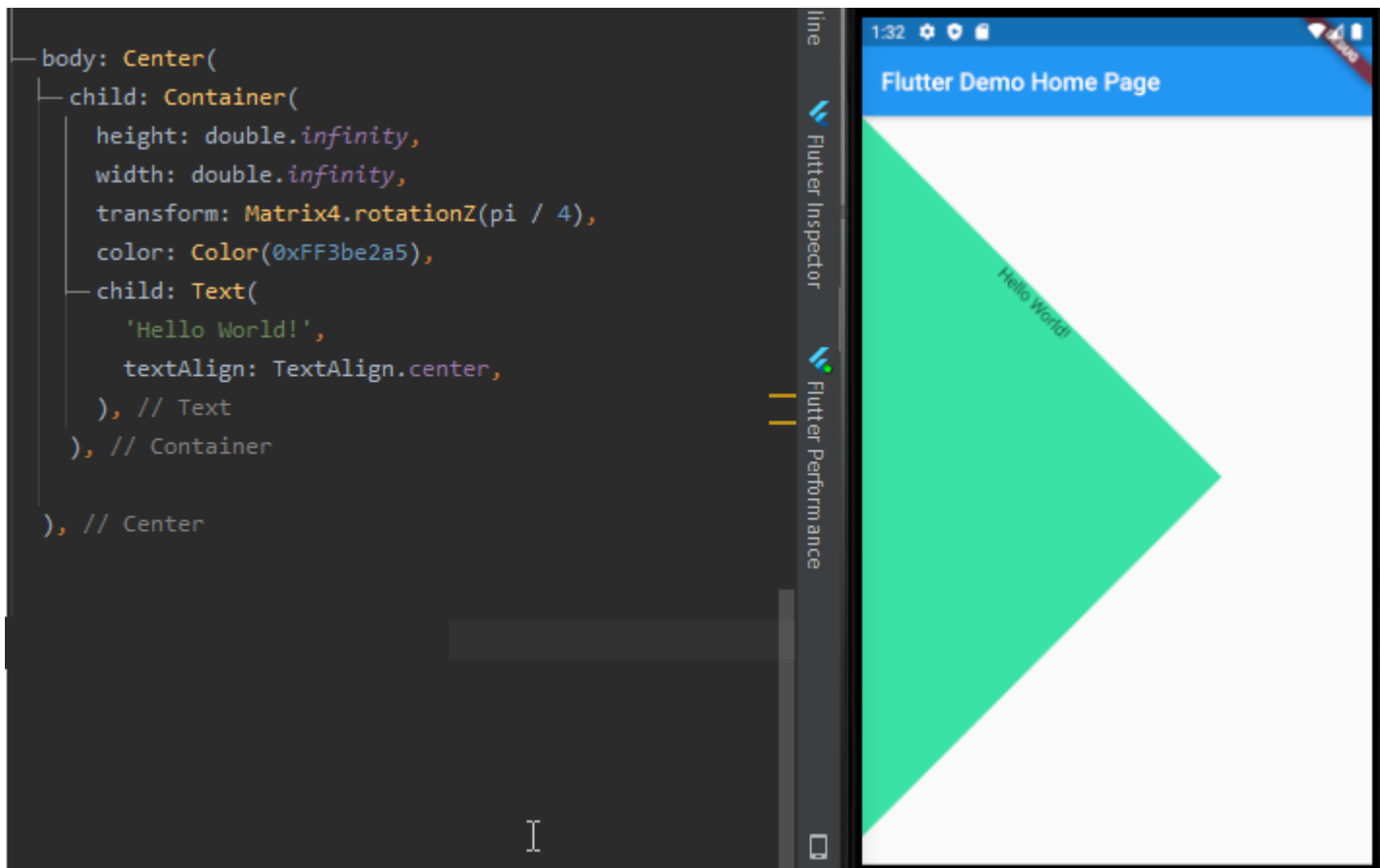
- ในกรณีที่ต้องการให้ขนาด Container ยืดเต็ม widget แม่ สามารถกำหนด double.infinity ให้กับ height หรือ width ได้เลย (จากภาพด้านล่างสี BG Color จะขยายเต็มจอ)

```
body: Center(  
  child: Container(  
    height: double.infinity,  
    width: double.infinity,  
    color: Color(0xFF3be2a5),  
    child: Text('Hello World!'),  
  ), // Container  
) // Center
```

A screenshot of the Flutter Demo Home Page. The page has a blue header with the text "Flutter Demo Home Page" and a green body with the text "Hello World!". The code on the left shows the widget tree for this page, where a Container widget is used to fill the entire screen with a green background and a centered text widget.

Container สำหรับใช้ในการ Transform widget

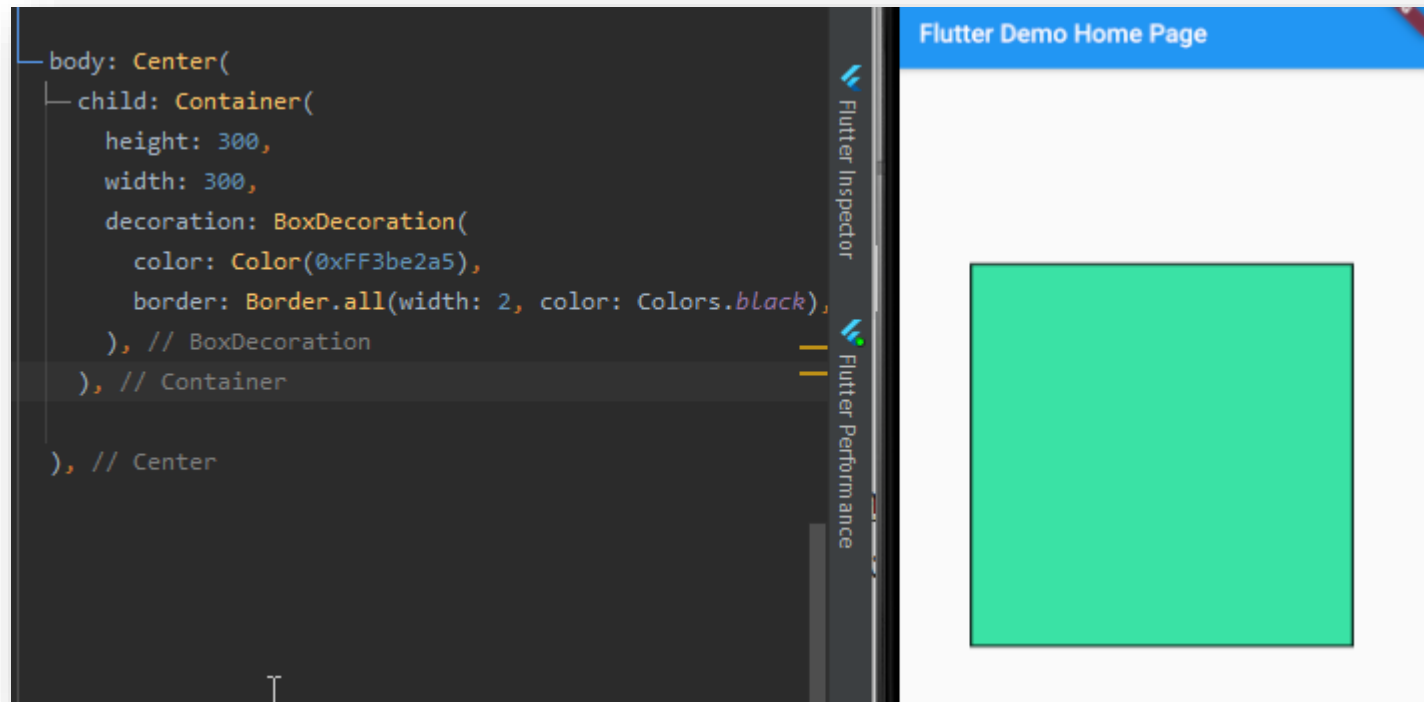
- กรณีที่ต้องการเปลี่ยนแปลง หรือ transform widget ไม่ว่าจะเป็นการหมุน เลื่อน เปลี่ยนรูปร่าง ในแนวนอน แนวตั้ง แนวทแยง



Pi จะแสดง error ให้กดที่ pi แล้ว import 'dart:math';

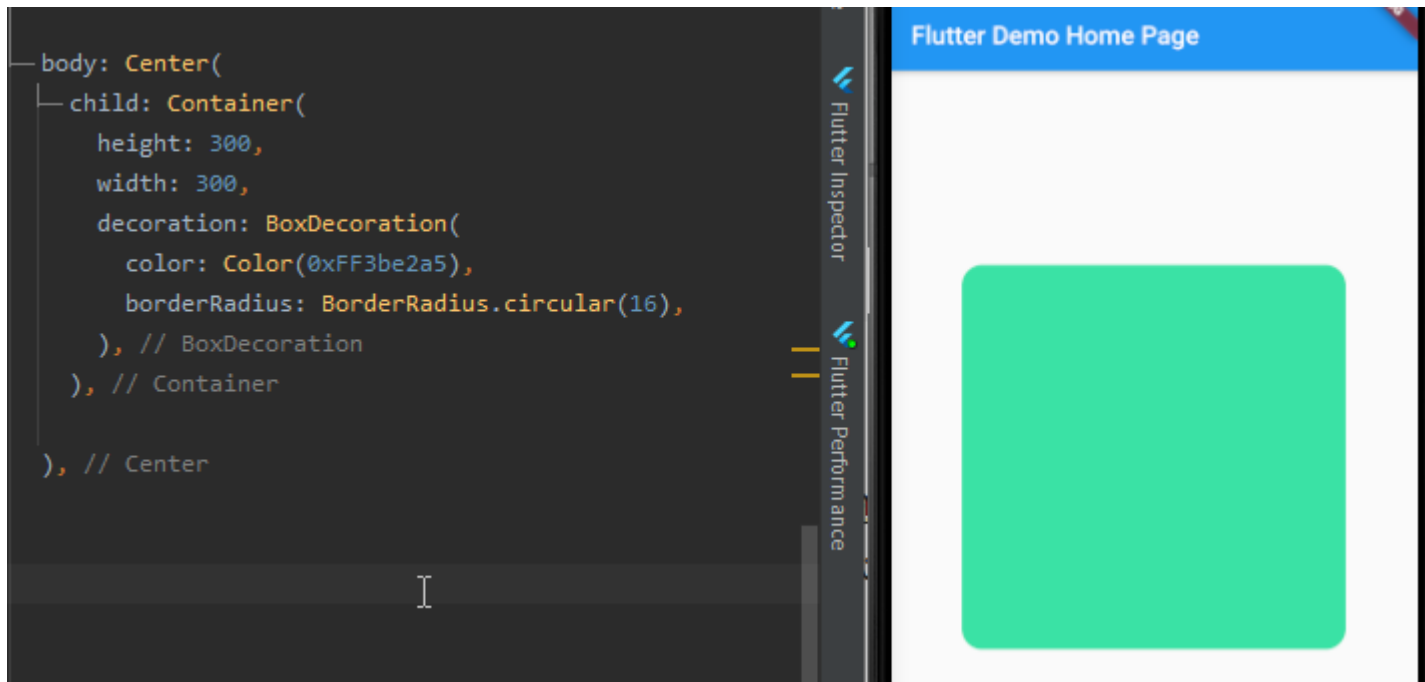
BoxDecoration **border: Border**

- เมื่อใช้ Container บ่อยแล้วสิ่งที่ต้องมาคู่กันคือ BoxDecoration สำหรับ Container ใช้กำหนด layout แล้ว ส่วน BoxDecoration ใช้สำหรับตกแต่งความสวยงามให้กับ Container อีกที เช่นการกำหนด **border: Border**



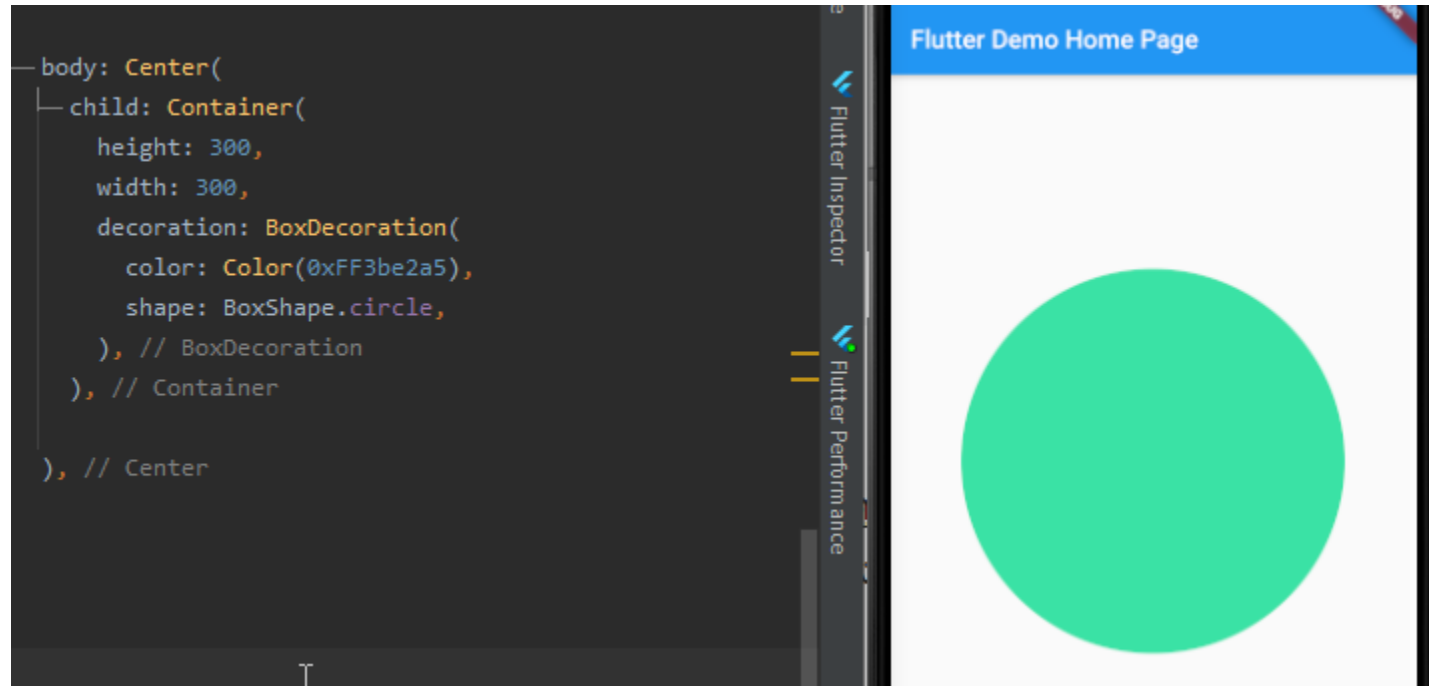
BoxDecoration borderRadius: BorderRadius

- borderRadius สำหรับกำหนดมุมมีความโค้ง ตามแบบฉบับ UI สมัยใหม่



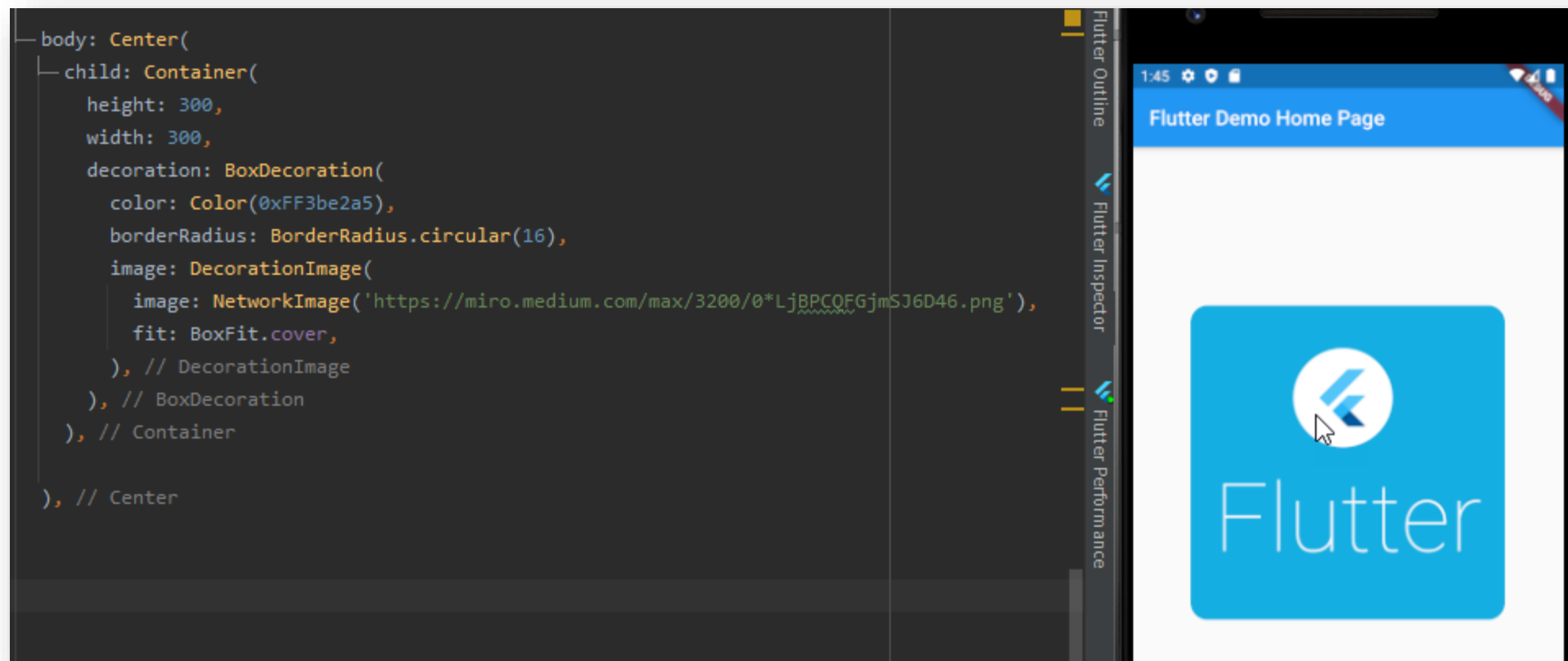
BoxDecoration shape: BoxShape

- shape ใช้สำหรับกำหนดรูปร่างของ Container สามารถกำหนดได้เป็นสี่เหลี่ยมหรือวงกลมก็ได้



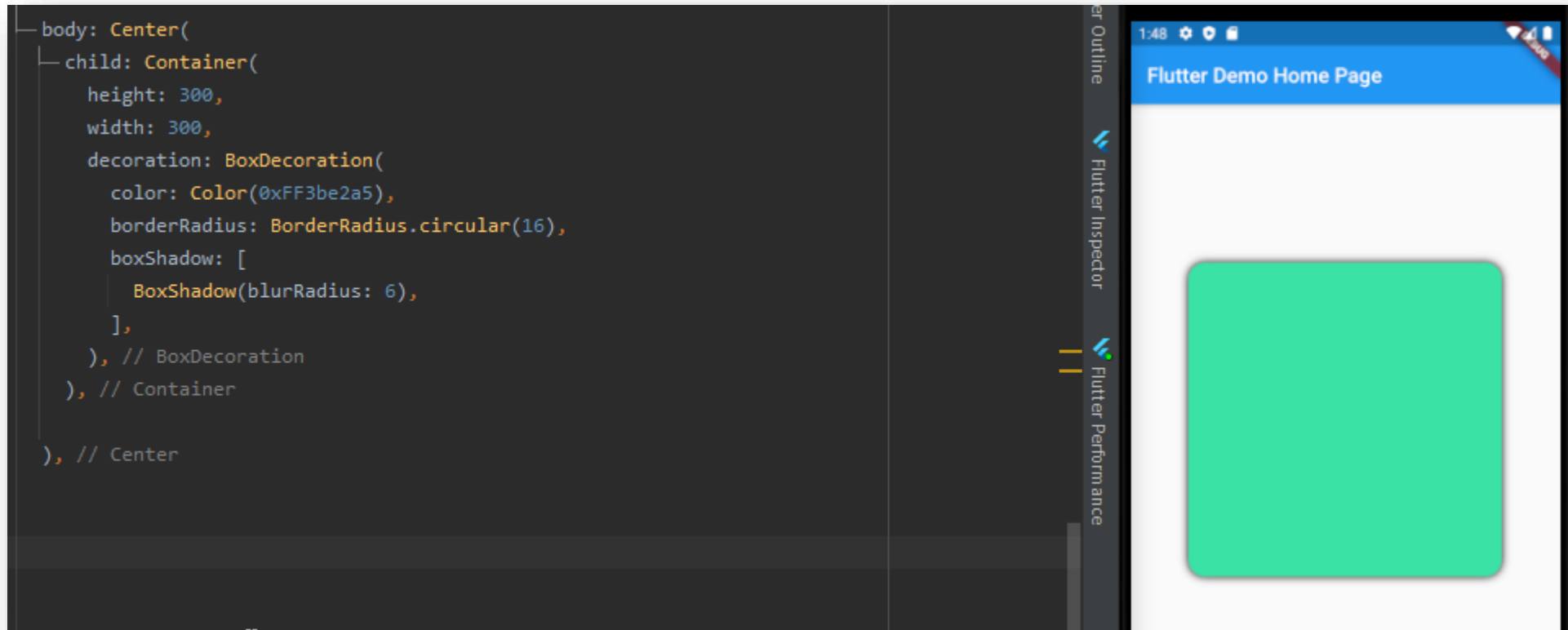
BoxDecoration image: DecorationImage

- ใช้กำหนดรูปพื้นหลัง โดยรูปจะมีรูปร่างตาม Container เช่น ต้องการให้รูปมีมุมโค้งสวยๆ



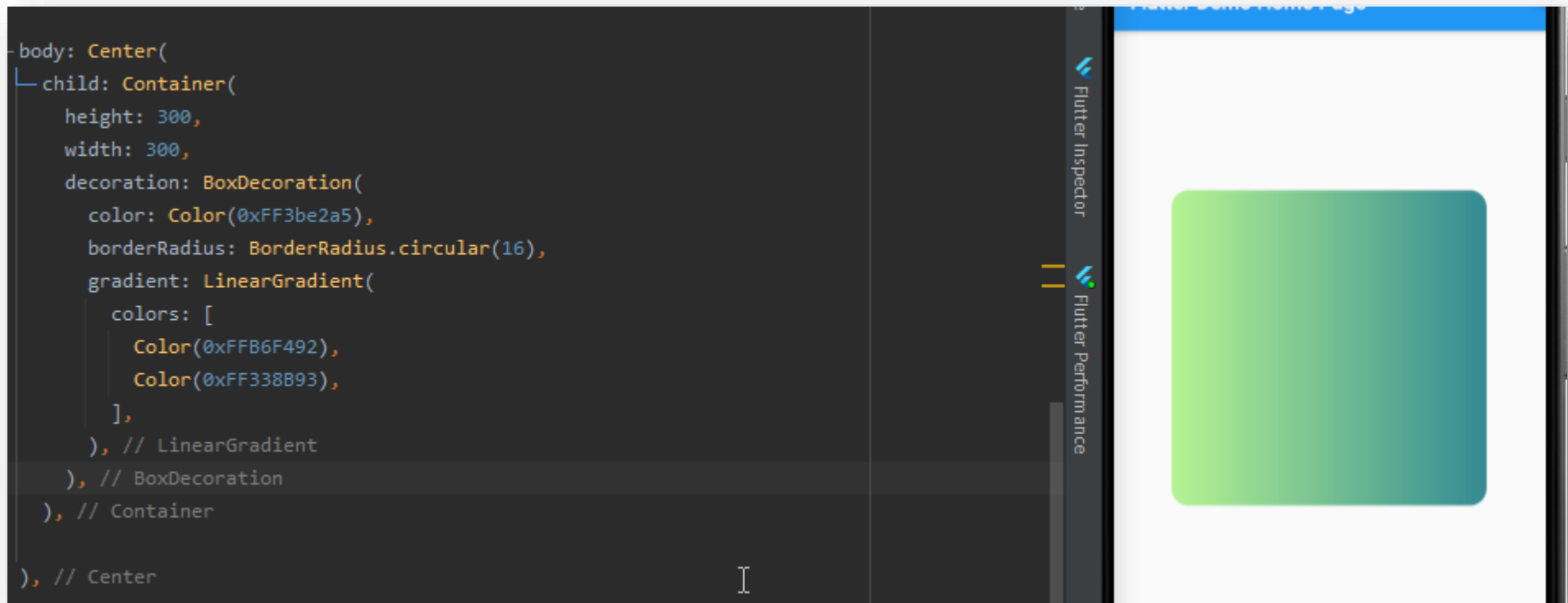
BoxDecoration boxShadow: List<BoxShadow>

- boxShadow ใช้สำหรับกำหนดเงาให้กับ Container



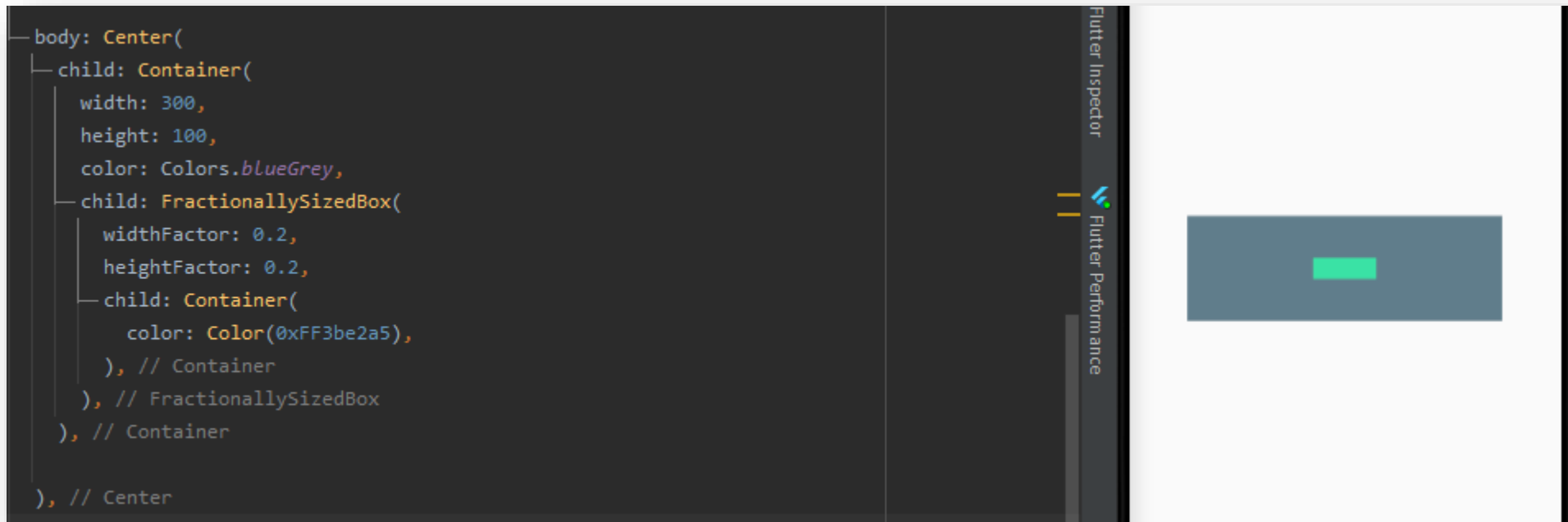
BoxDecoration gradient

- ใช้ตกแต่งลูกเล่นการไล่เฉดสี



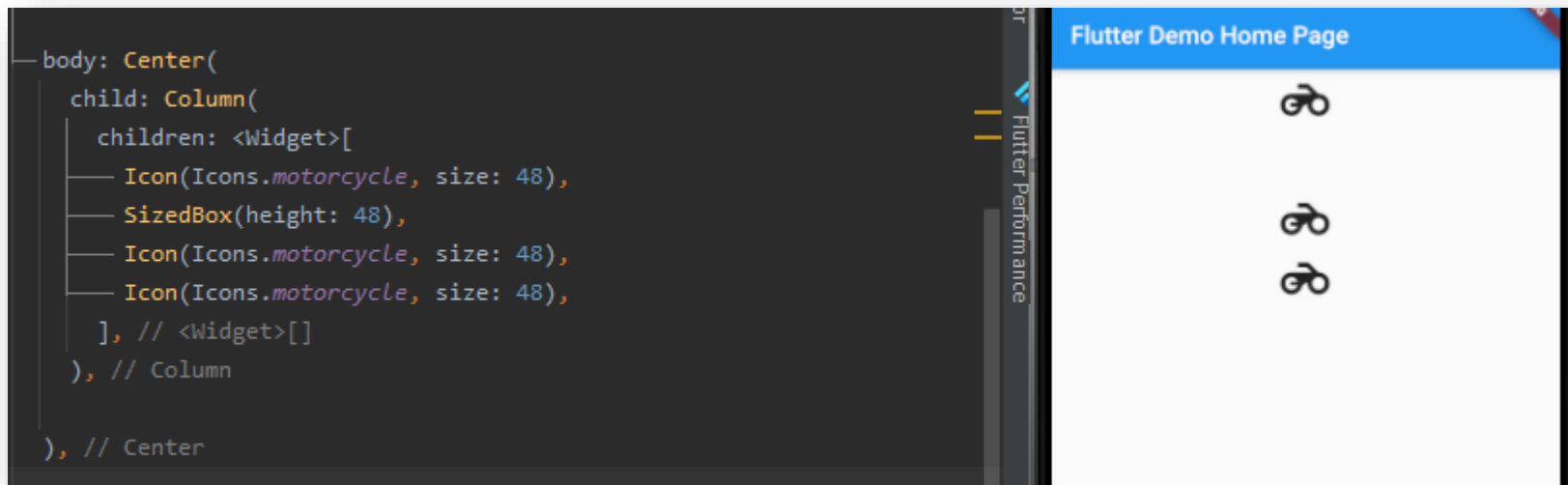
BoxDecoration FractionallySizedBox

- หากต้องการกำหนดขนาดของ widget ลูก สัมพันธ์เป็น % กับขนาดของ widget แม่ สามารถใช้ FractionallySizedBox เพื่อกำหนดความสัมพันธ์ขนาดได้ผ่าน widthFactor และ heightFactor เช่น widthFactor: 0.2 คือกำหนดขนาดให้เป็น 20% ของ widget แม่นั่นเอง



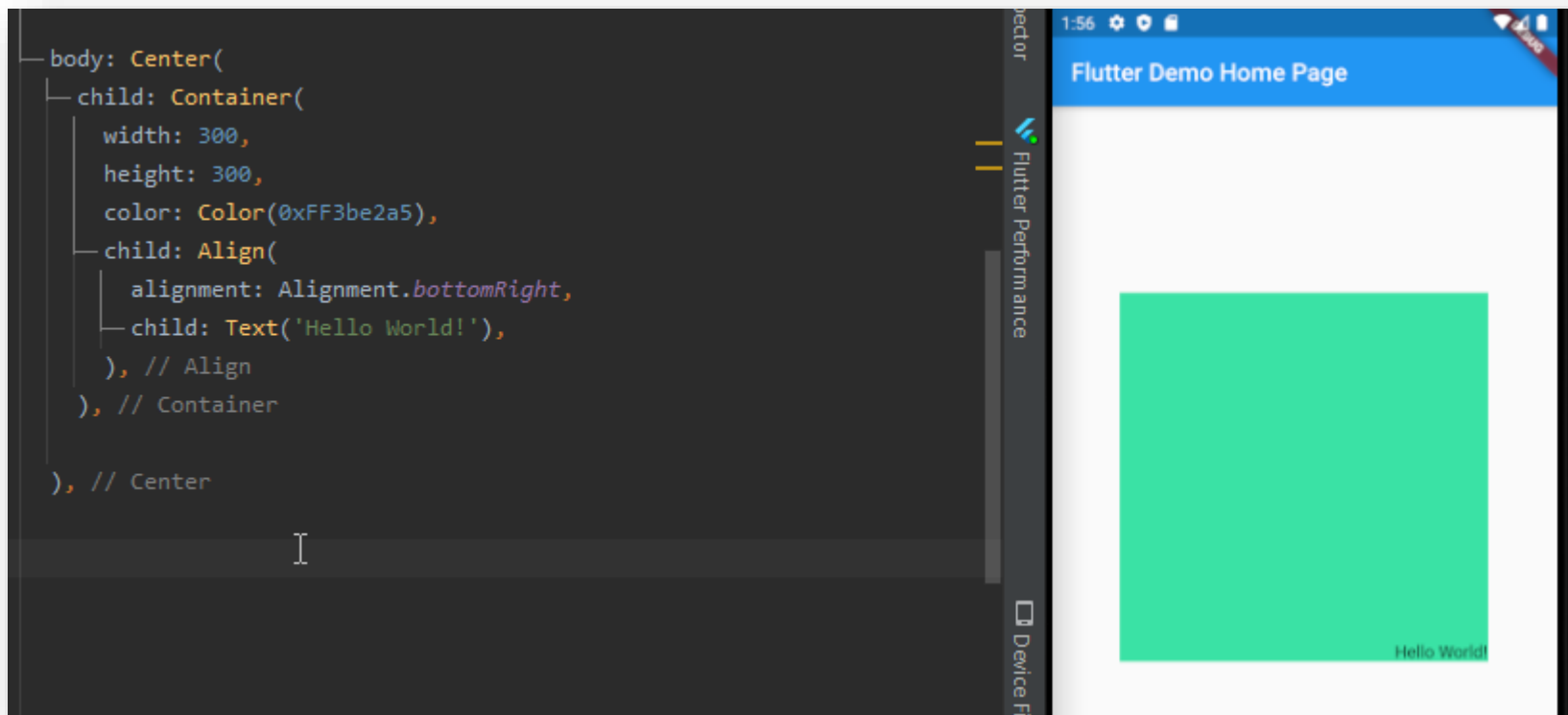
SizedBox

- ใช้ SizedBox เป็น Padding



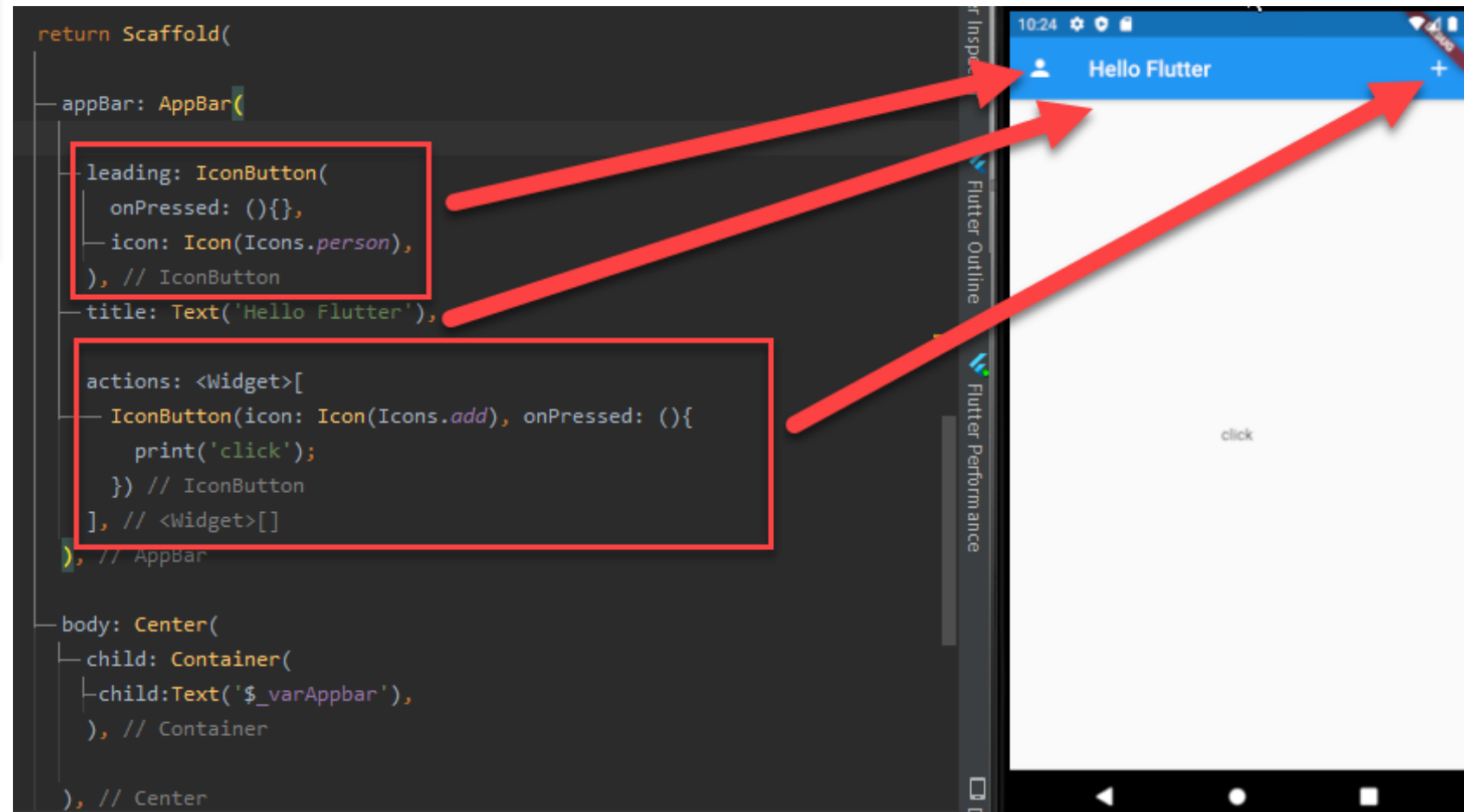
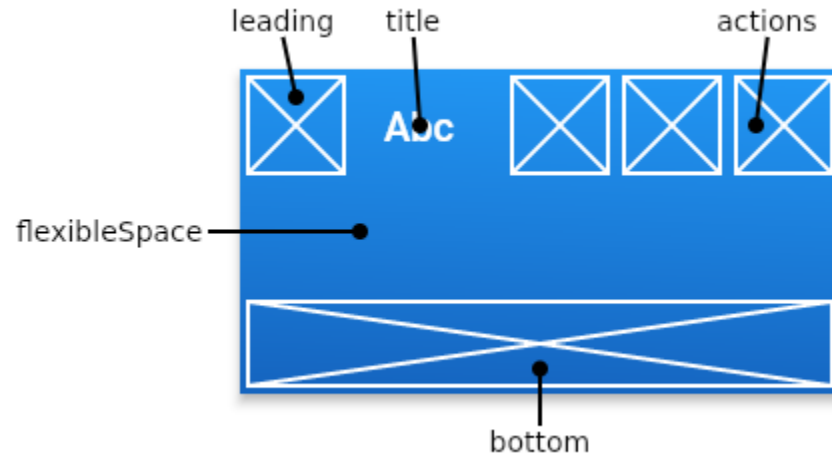
Align

- กรณีที่ต้องจัดการตำแหน่งภายใน Container แบบง่ายๆและไม่ต้องยุ่งยากใช้ Stack สามารถใช้ Align ในการจัดการตำแหน่งได้ง่ายๆ โดยมีทั้งหมด 8 ทิศให้กำหนด คือ
- centerLeft
- topLeft
- topCenter
- topRight
- centerRight
- bottomRight
- bottomCenter
- bottomLeft



AppBar

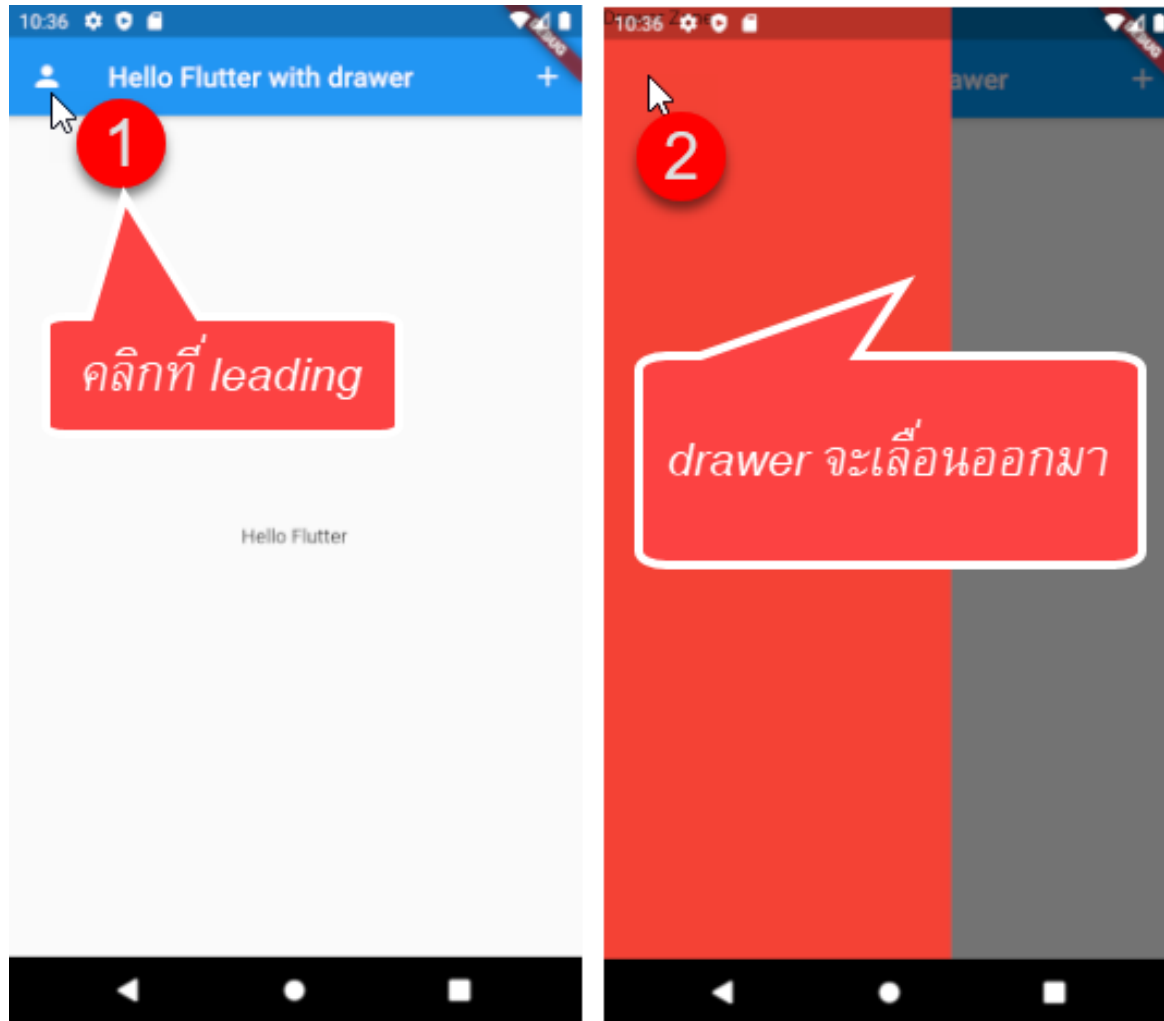
- <https://api.flutter.dev/flutter/material/AppBar-class.html>



AppBar การใช้งาน Drawer

โดยในตัวอย่างจะใช้รูปคน ที่เป็น leading เปิดใช้งาน drawer เป็นการ slide drawer ออกมา (slide widget จากด้านข้าง)

หลักการทำงาน



AppBar การใช้งาน Drawer

1. ประกาศตัวแปร

```
final GlobalKey<ScaffoldState> _scaffoldKey = new GlobalKey<ScaffoldState>();
```

2. ทำการเพิ่ม Key ในส่วนของ Scaffold

```
key: _scaffoldKey
```

```
class MyHomePageState extends State<MyHomePage> {  
  final GlobalKey<ScaffoldState> _scaffoldKey = new GlobalKey<ScaffoldState>();  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      key: _scaffoldKey,  
      appBar: AppBar(  
        leading: IconButton(  

```

3. ทำการเรียกใช้งานเปิด drawer

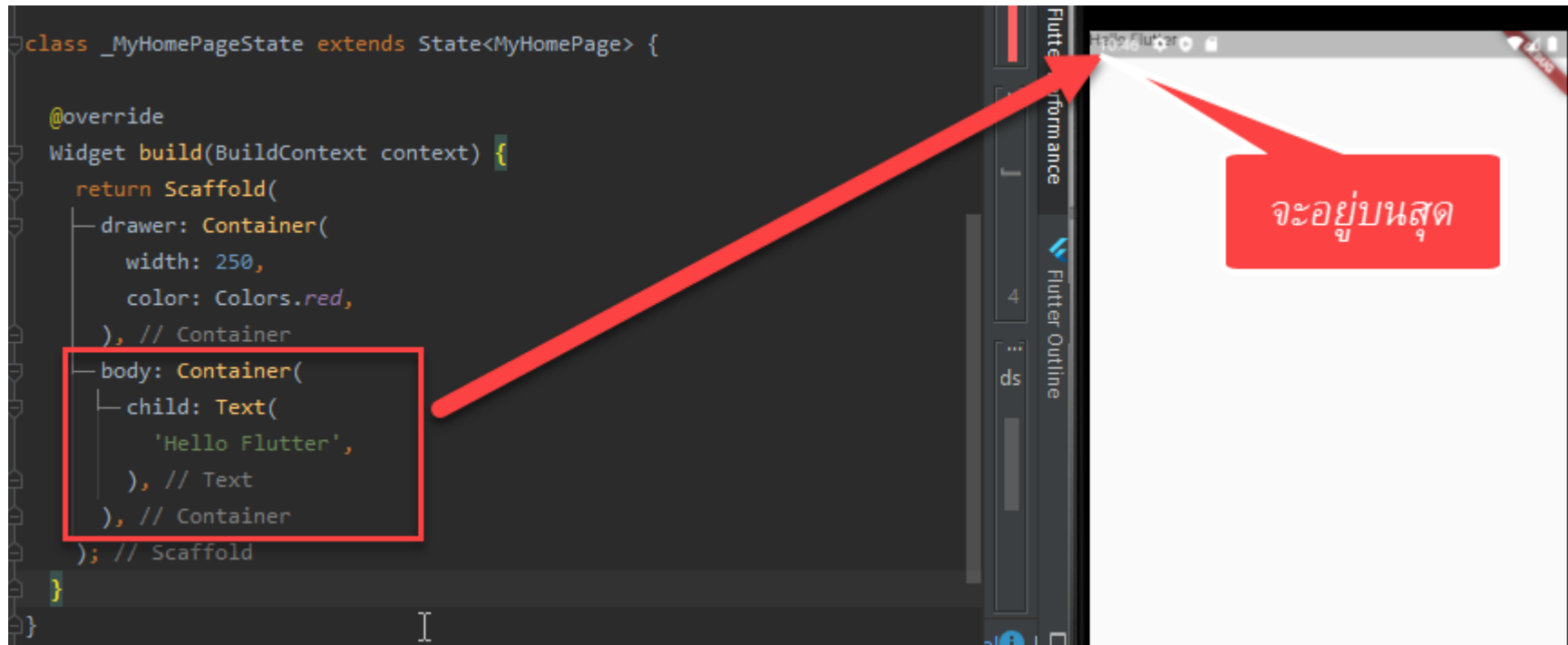
```
_scaffoldKey.currentState.openDrawer();
```

```
        appBar: AppBar(  
          leading: IconButton(  
            onPressed: () {  
              _scaffoldKey.currentState.openDrawer();  
            },  
            icon: Icon(Icons.person),  
          ), // IconButton  
          title: Text('Hello Flutter with drawer'),  
          actions: <Widget>[  
            IconButton(  
              onPressed: () {  
                print('Add');  
              },  
              icon: Icon(Icons.add),  
            ), // IconButton  
          ], // <Widget>[]  
        ), // AppBar  
        drawer: Container(  
          height: double.infinity,  
          width: 250,  
          color: Colors.red,  
          child: Text('Drawer Zone'),  
        ), // Container  

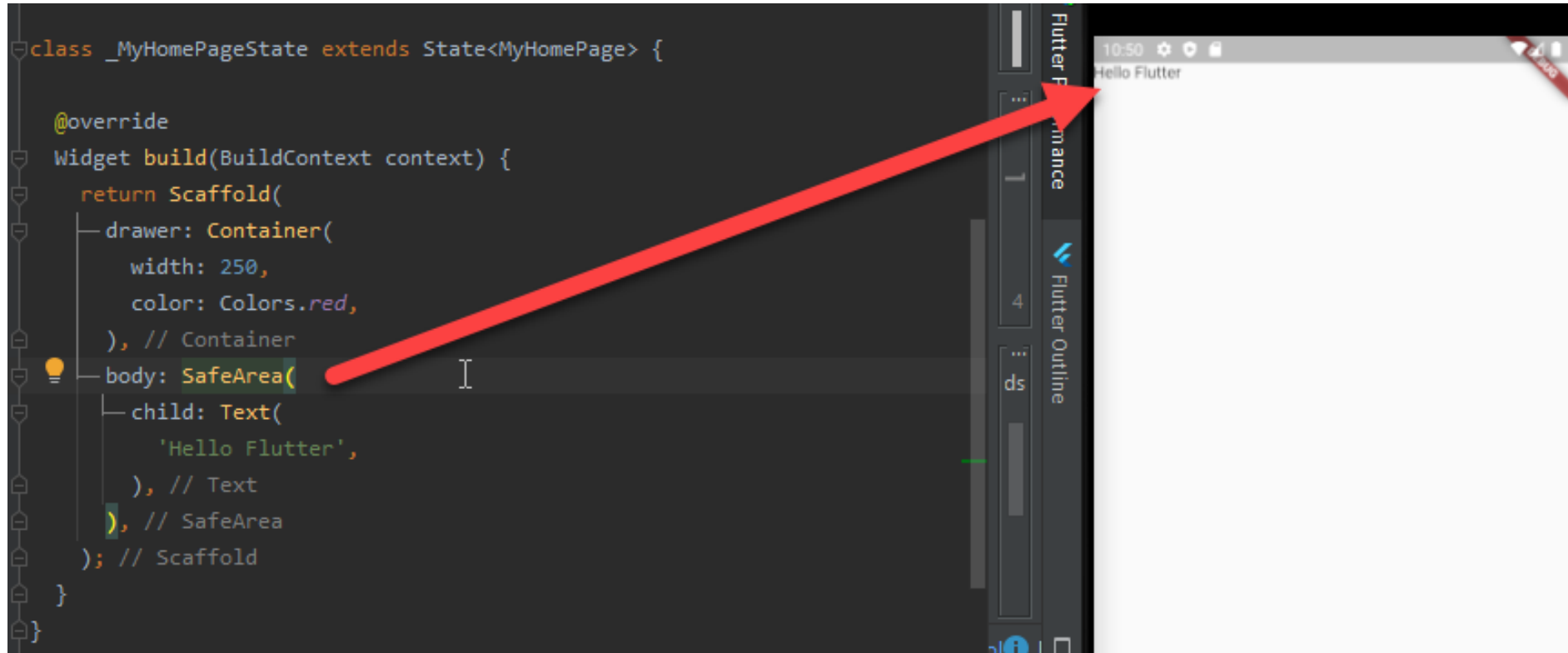
```

SafeArea

ถ้าในส่วน body เราไม่กำหนด SafeArea แอปเราจะแสดงผลทั้งจอ โดยไม่สน Layout ของแอปอื่นๆ ในจอ หรือของระบบดังภาพด้านล่างที่ คำว่า Hello Flutter จะไปอยู่ซ้อนกับ icon โทรศัพท์ วิธีแก้คือจะต้องกำหนด SafeArea ขึ้นมา



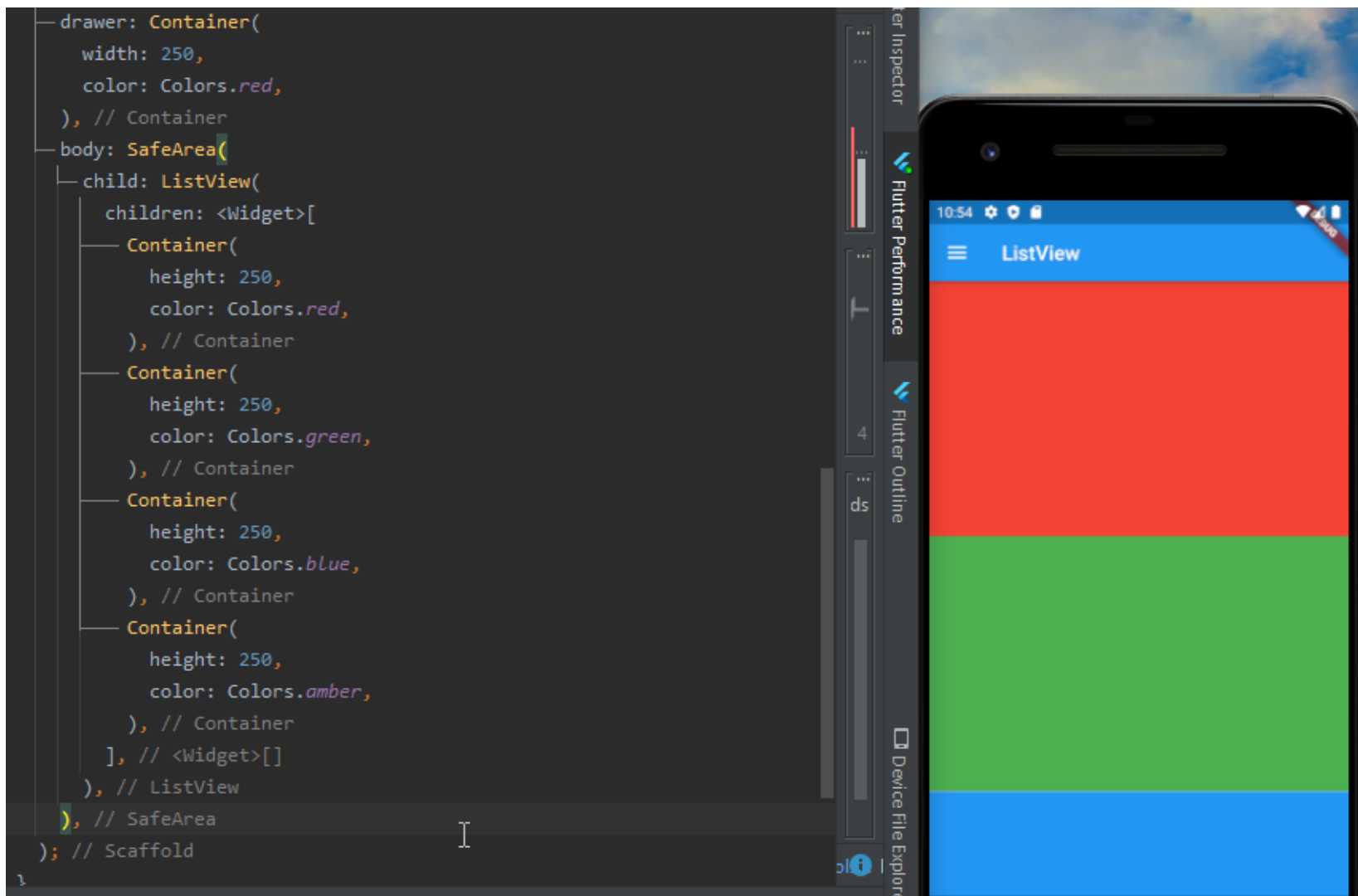
SafeArea



- จากตอนที่ body แสดงผลแบบ Container เปลี่ยนมาเป็น SafeArea ก็จะทำให้คำว่า Hello Flutter ไม่ไปซ้อนกับ icon ด้านบน

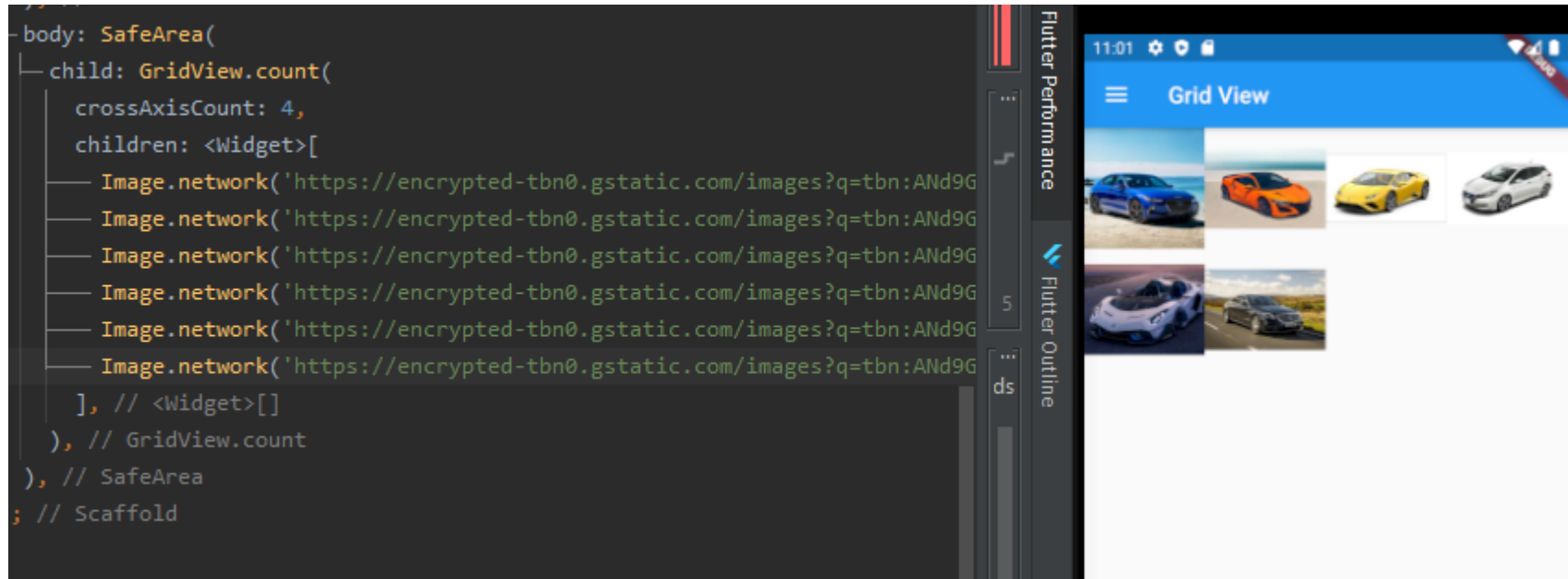
ListView

- ListView ใช้ในกรณีที่ Content ของเราเยอะ จนยาวกว่าหน้าจอจนจะครบ มันจะทำให้เราสามารถ Scroll ขึ้น ลง ได้



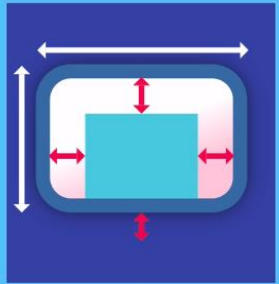
GridView

- GridView รูปแบบจะเหมือนกับ Gallery

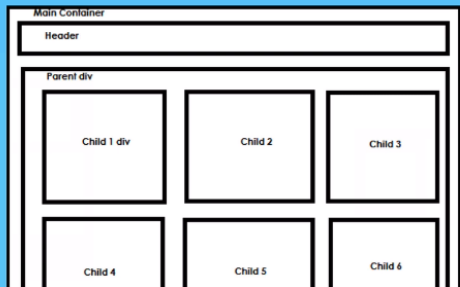


การวาง Container Layout

Container()



Div



Layout widgets - Flutter

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

Flutter

Docs Showcase Community

Single-child layout widgets

Container

A convenience widget that combines common painting,

Padding

A widget that insets its child by the given padding.

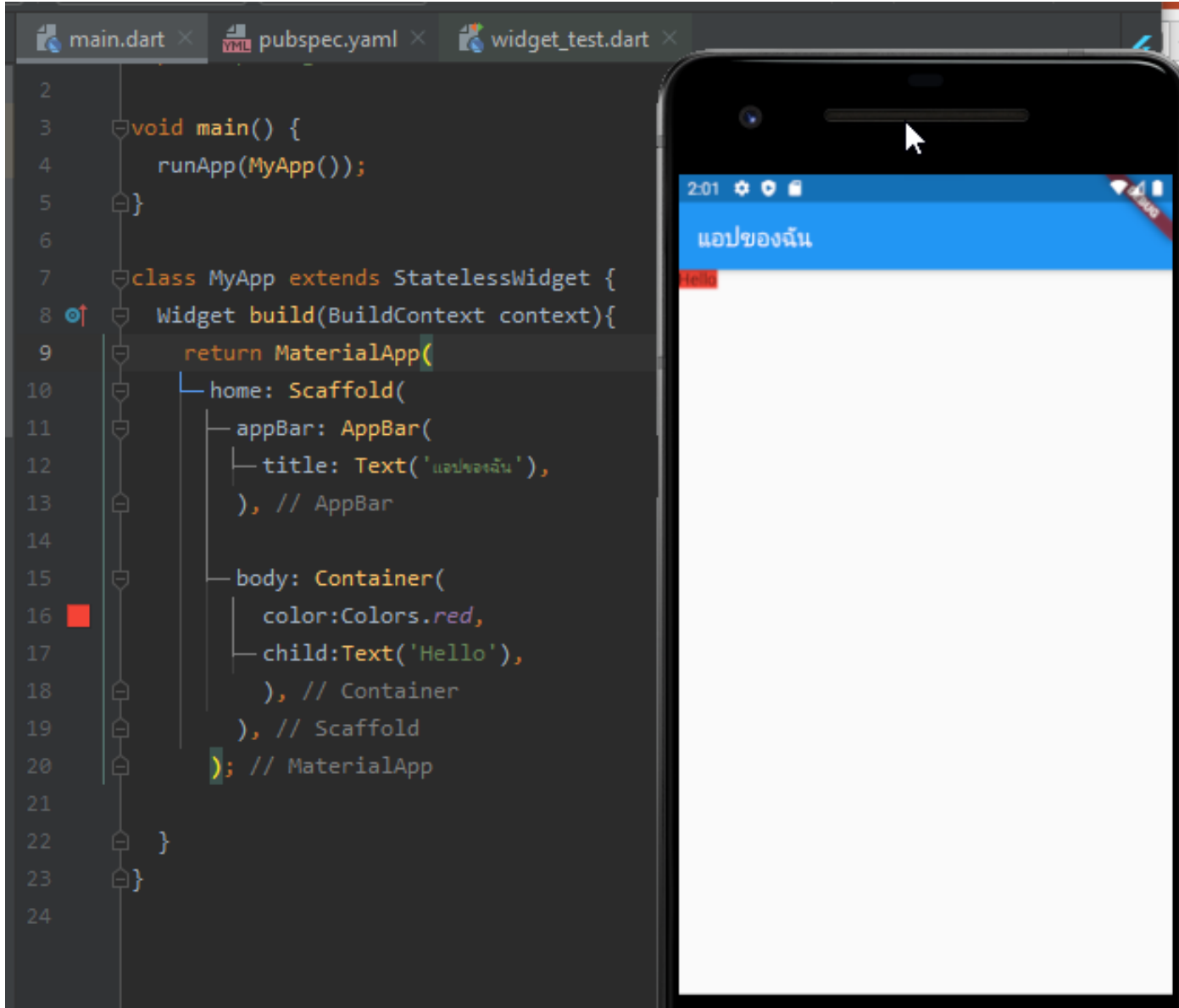
Center

A widget that centers its child within itself.

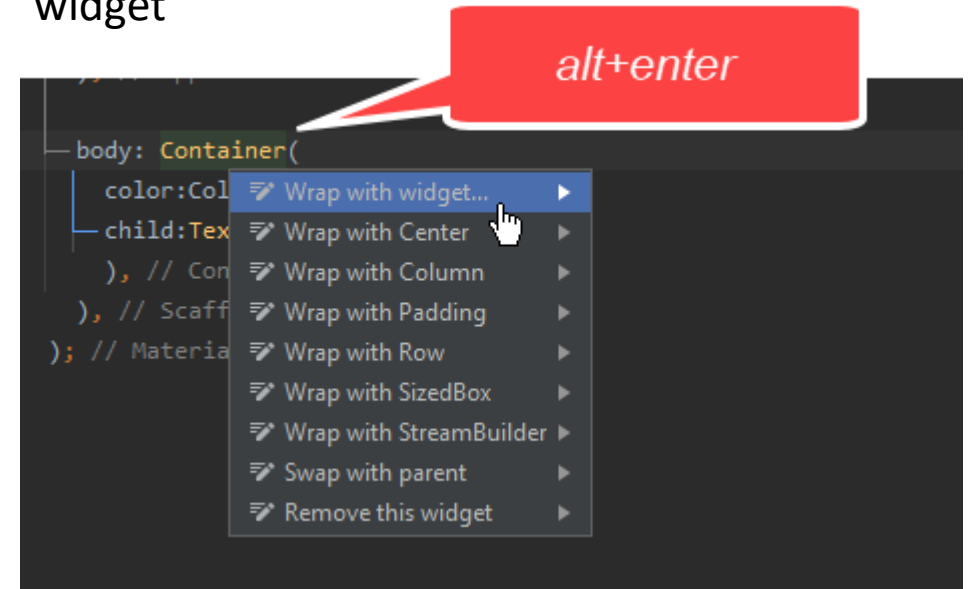
The diagram illustrates the visual structure of a widget layout. It shows a central cyan rectangle representing the child widget. This child is surrounded by a white area representing padding. The padding is further enclosed by a blue border. The entire structure is contained within a dark blue container. Red double-headed arrows indicate the spacing between the child and the padding, and between the padding and the border. The diagram is part of a documentation page for Flutter layout widgets, specifically focusing on single-child layout widgets. The page includes a navigation menu on the left with links to various Flutter documentation sections. The top of the page features the Flutter logo and navigation links for Docs, Showcase, and Community. The main heading is 'Single-child layout widgets', and the three widgets shown are Container, Padding, and Center.

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

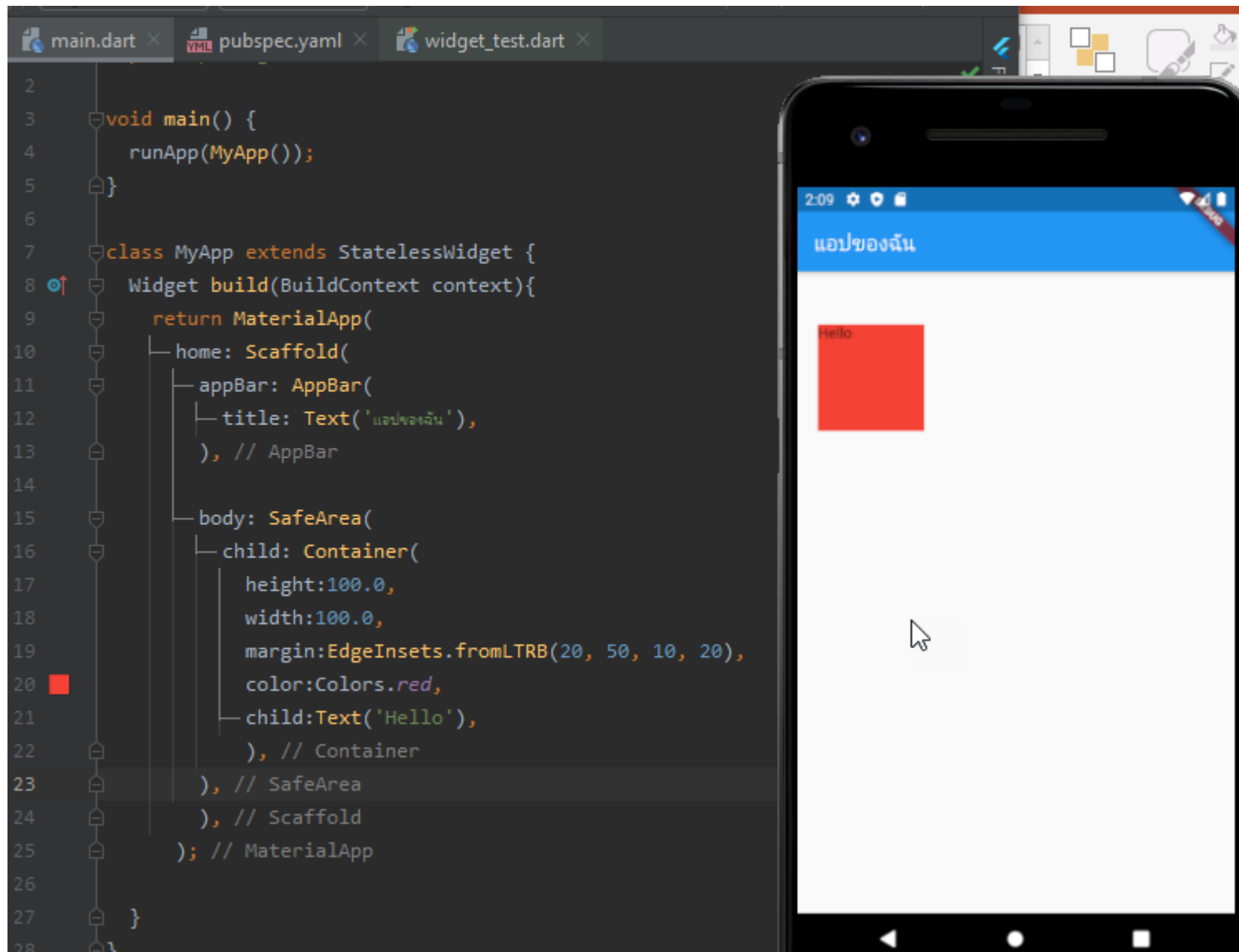
Home Layout



ถ้าใส่โค้ดตามภาพจะสังเกตว่า ข้อความ Hello จะติดกับ appBar เราสามารถกำหนด Layout ได้ โดยในส่วน ของ `body:Container` ให้กด `alt+enter` หรือ `option+enter` เลือกอันแรก wrap width widget



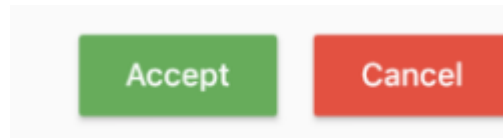
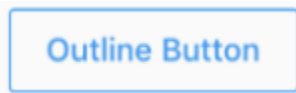
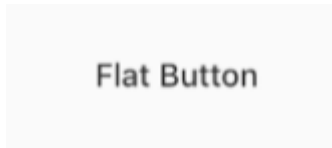
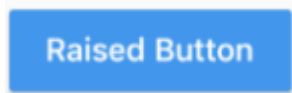
Container Layout



หลังจากนั้นให้ widget เป็น `SafeArea` เพื่อให้มัน
Fix Layout แล้วกำหนดขนาด ความกว้าง ความสูง และ
margin ของ element ได้ตามโค้ด
ในโค้ดสามารถกำหนด `margin:EdgeInsets`

Button

- Button คือปุ่มเอาไว้ให้ user ทำการกด เพื่อสั่งให้แอปทำงานอย่างใดอย่างหนึ่ง โดยใน Flutter จะมี ปุ่มกดหลากหลายรูปแบบให้ใช้งานดังนี้



Button Bar

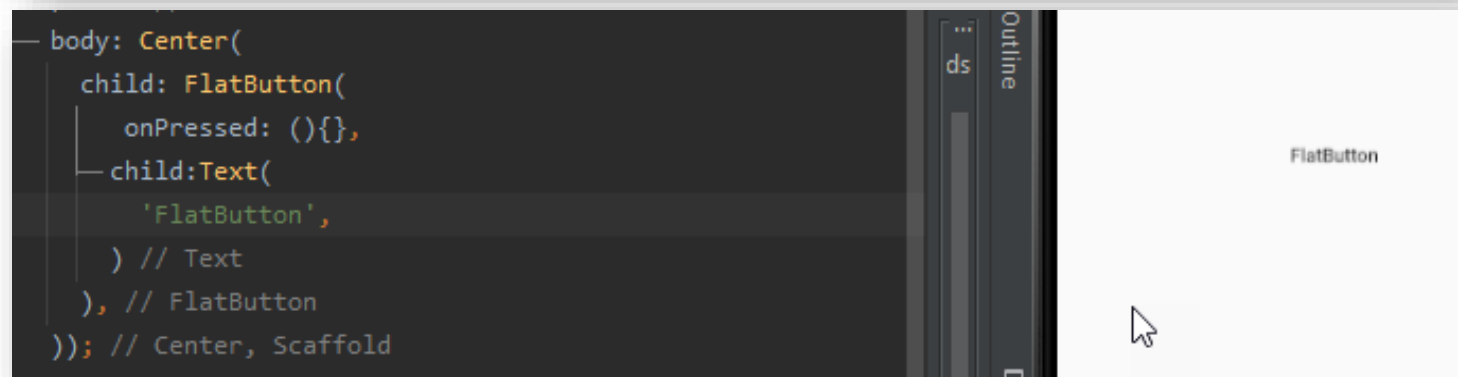
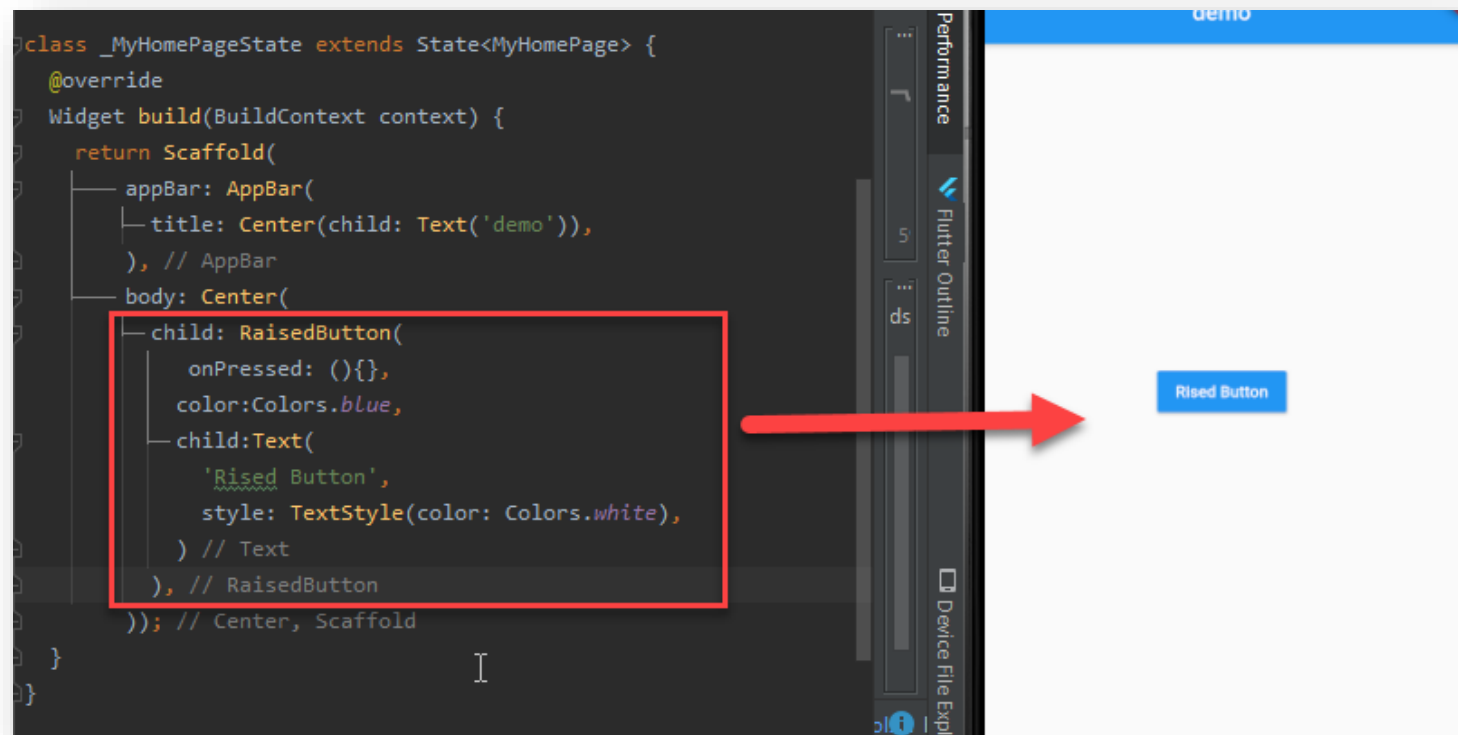


FloatingActionButton

โดยปุ่ม Button จะมี Properties ดังนี้

- onPressed: /* ส่วนของการ handle callback */
- shape: /* ส่วนของการกำหนดลักษณะ รูปร่าง */
- color: /* ส่วนของการกำหนดสีของพื้นหลัง */
- child: /* ส่วนของ widget ลูก */

Button



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

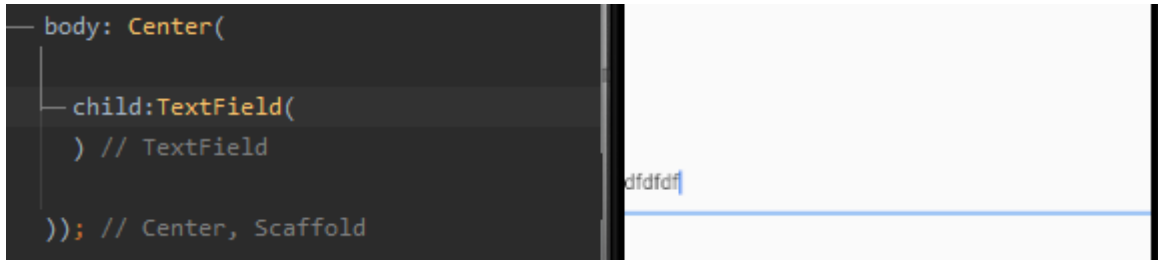
```
body: Center(  
  child: ButtonBar(  
    alignment: MainAxisAlignment.center,  
    children: <Widget>[  
      RaisedButton(  
        onPressed: () => {},  
        color: Colors.green,  
        child: Text('Accept', style: TextStyle(color: Colors.white)),  
      ), // RaisedButton  
      RaisedButton(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

Accept

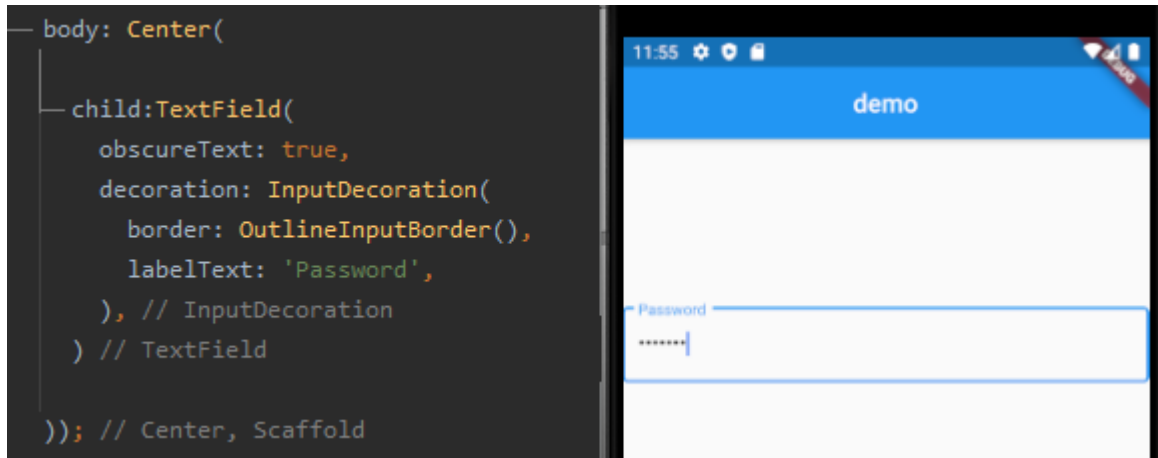
Cancel

TextField

TextField คือ Widget Input ที่จะรับค่าจากการกด key ตัวอักษรลงไปในระบบ

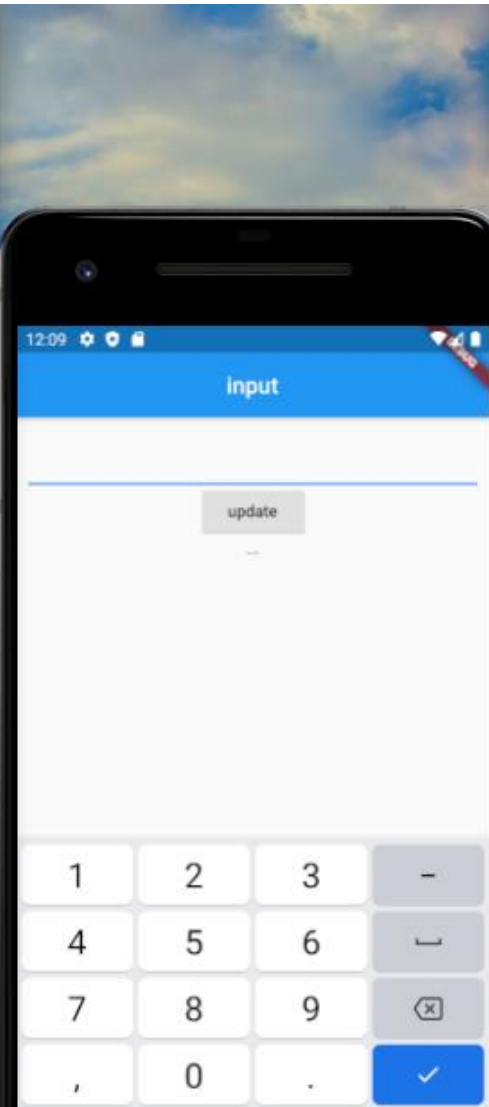


สามารถกำหนดรูปแบบเหมือนกรอก password ได้

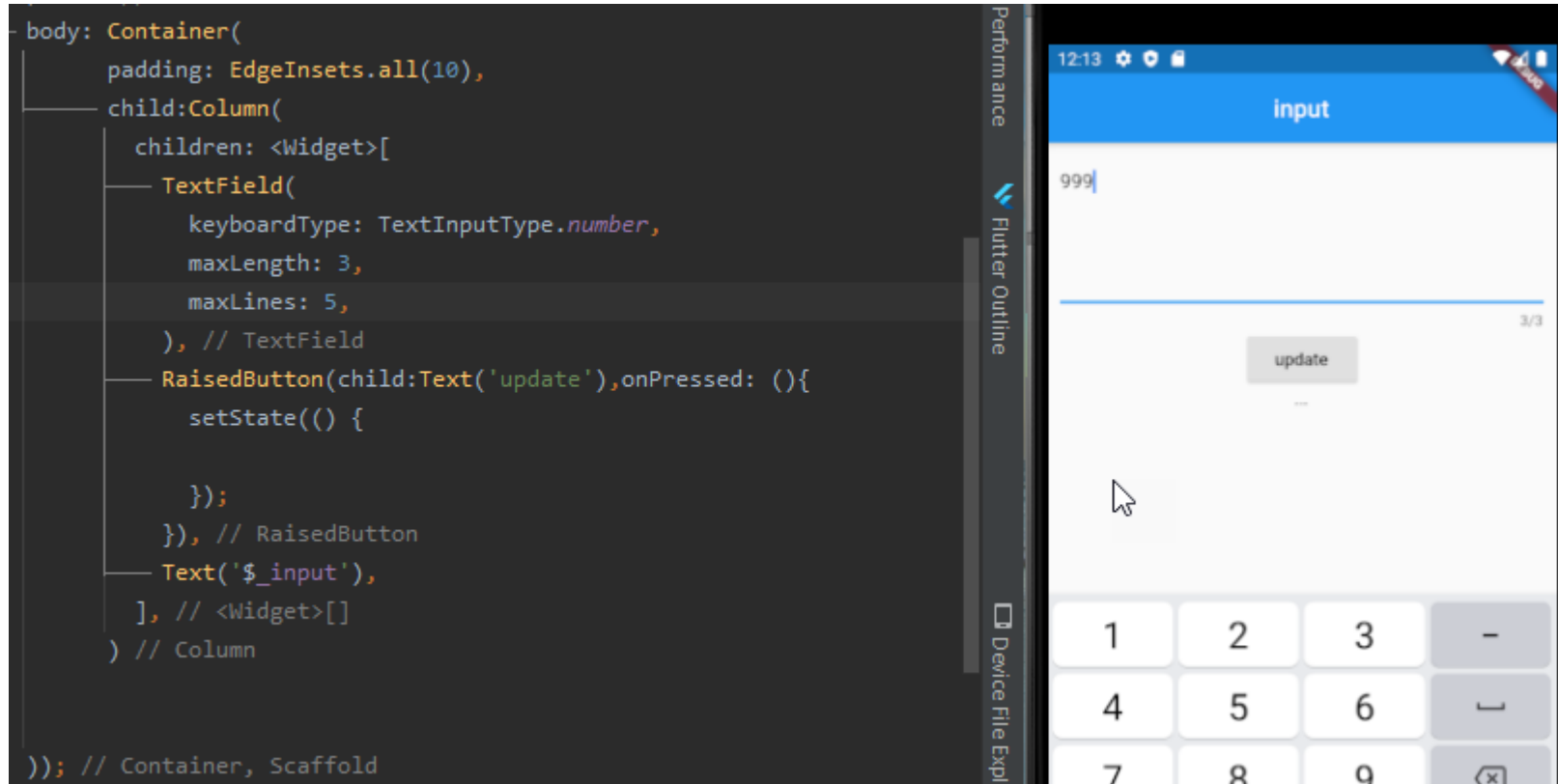


ตัวอย่างการกรอกข้อมูลใน TextField แล้วกดปุ่มให้แสดงค่าที่กดได้

```
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 = '123';  
              });  
            }, // RaisedButton  
            Text('${_input}'),  
          ], // <Widget>[]  
        ) // Column  
      )); // Container, Scaffold  
    }  
}
```

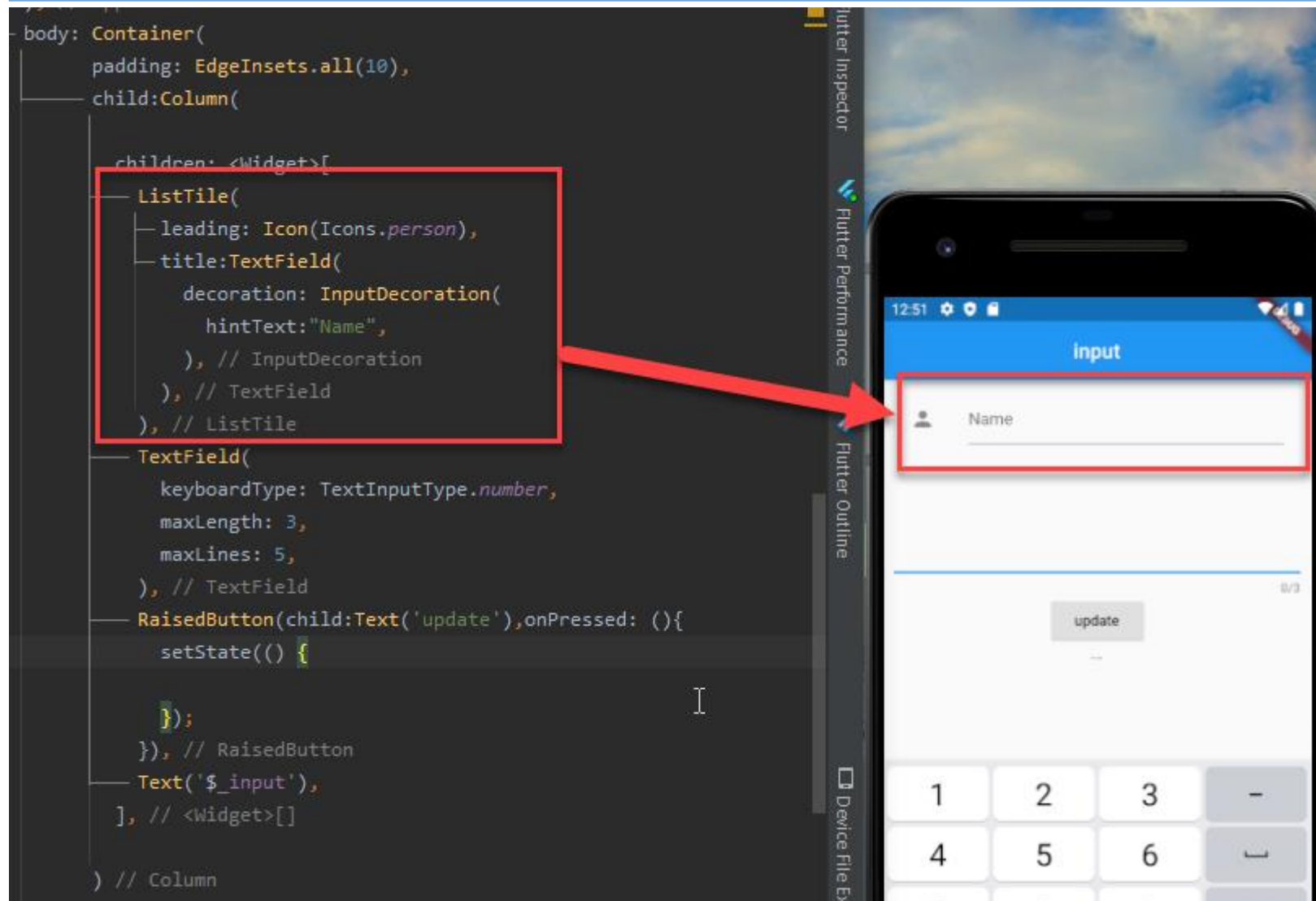


TextField มี Properties ที่น่าสนใจดังนี้

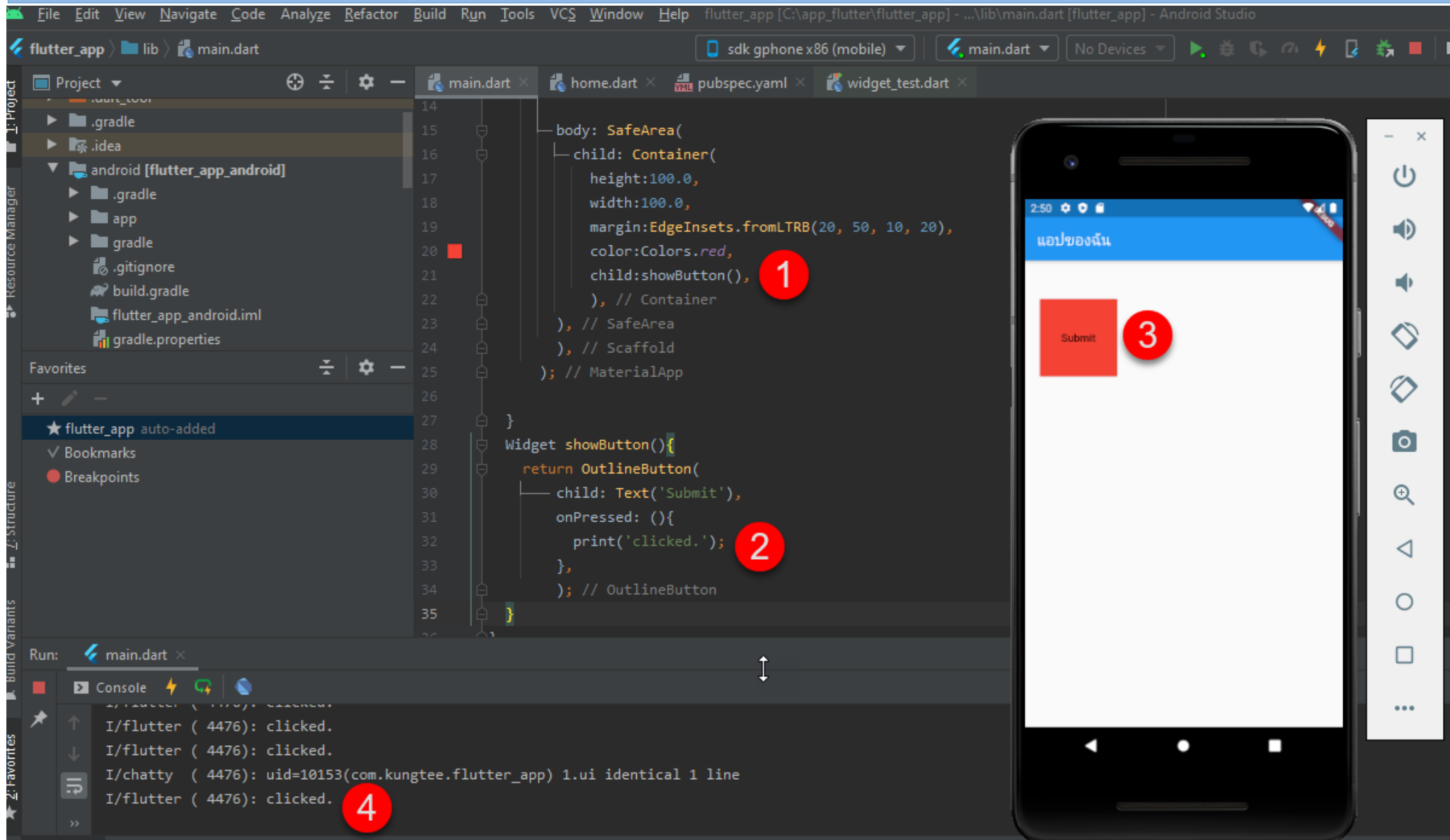


- keyboardType กำหนดรูปแบบการกรอกข้อมูล เช่นกรอกแต่ตัวเลข กรอกอีเมล
- maxLength กำหนดขนาดความยาวของตัวอักษร
- maxLines กำหนดจำนวนไลน์ที่สามารถกรอกได้ (จำนวนบรรทัดในการกรอก)
- TextInputAction สามารถกำหนดให้มีรูปแหวนขยายมาใน keyboard ได้ (TextInputAction=TextInputAction.search)

ListTile widget ที่สามารถใส่ leading และ textfield ได้



การสร้างปุ่ม button และให้กด click



Widget คืออะไร

ใน Flutter จะมองทุกอย่างเกือบทั้งหมดเป็น widget

Widget คือ ส่วนที่ถูกใช้สร้างเป็นหน้าตาของ App หรือที่เรียกว่า user interface (UI) โดยนำมาประกอบเรียงกันเป็นลำดับขั้นขึ้นเป็นโครงสร้าง แต่ละ widget จะถูกวางซ้อนอยู่ภายใน Parent widget และได้รับการส่งต่อสืบทอดคุณสมบัติต่างๆ จาก Parent อีกที แม้กระทั่ง 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 และกำหนดมุมมองเลเอาท์ (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 ก็คือสถานะของสิ่งนั้นๆ stateless จึงหมายถึง widget ที่ไม่มี state หรือไม่มีสถานะการเปลี่ยนแปลง หรือไม่จำเป็นต้องใช้งานการเปลี่ยนแปลง จึงใช้งาน widget นี้ ส่วน stateful หมายถึง widget ที่มี state หรือมีสถานะการเปลี่ยนแปลง ไปตามข้อมูลที่ได้รับหรือจากการกำหนดจากผู้ใช้อื่นๆที่แตกต่างกันที่สำคัญของทั้งสองส่วนนี้คือ stateful widget จะมี State object ที่ใช้ในการเก็บข้อมูล state และ ทำการส่งต่อสำหรับใช้งานในกระบวนการสร้าง widget ใหม่เมื่อมีการเปลี่ยนแปลง ทำให้ค่า state ไม่ได้หายไปไหน

การใช้งาน StatelessWidget

Stateless widget ใน Flutter เป็น widget ที่ไม่จำเป็นต้องมีการเปลี่ยนแปลง state เกิดขึ้น โดยเราจะใช้ stateless widget สำหรับสร้าง widget แบบคงที่ เหมาะสำหรับการใช้ในการสร้าง และกำหนดส่วนของ UI ซึ่งจะปรับแต่งเฉพาะค่าข้อมูลของ ตัว widget เท่านั้นเช่น Text widget ก็ถือเป็น stateless widget ที่เป็น subclass ของ StatelessWidget

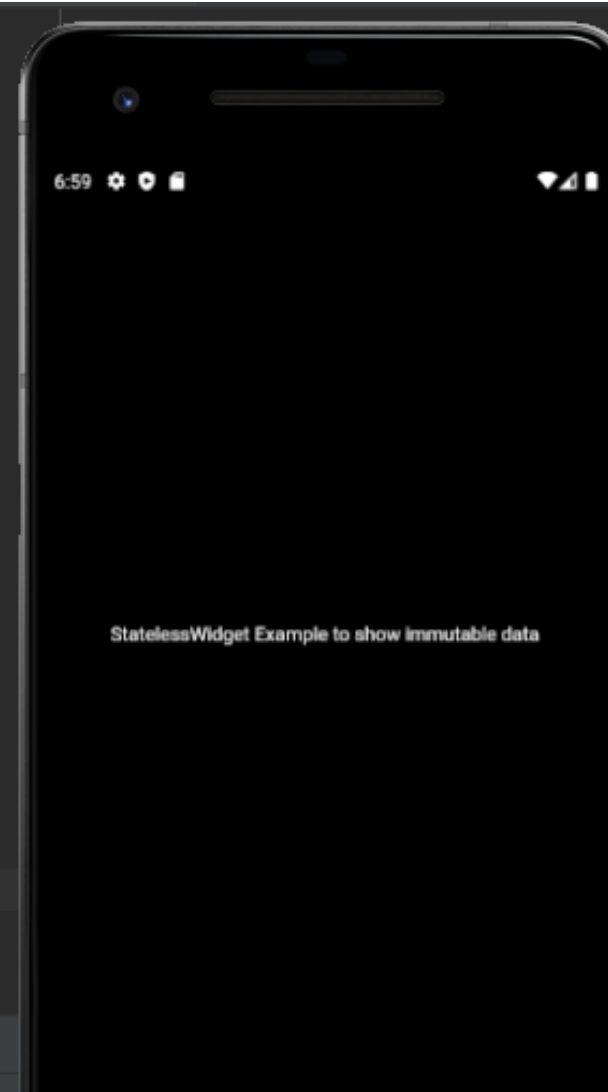
StatelessWidget ตามโค้ดด้านล่าง

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

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

class MyStatelessWidget extends StatelessWidget {
  final String text;
  // constructor
  MyStatelessWidget({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: 'ทดสอบการใช้ Route',
16       theme: ThemeData(
17         primarySwatch: Colors.green,
18       ), // ThemeData
19       initialRoute: '/',
20       routes: {
21         // เมื่อ ให้ "/" route ให้ล้ร้าหน้าแรก
22         '/': (context) => Page1(title: "หน้าที่ 1",),
23         // เมื่อ ให้ "/page2" route ให้ล้ร้าหน้าที่สอง
24         '/page2': (context) => Page2(title: "หน้าที่ 2",),
25       },
26     ); // MaterialApp
27   }
28 }
```

การทำ route เพื่อเปิดอีกหน้า (ต่อ)

- Page1.dart

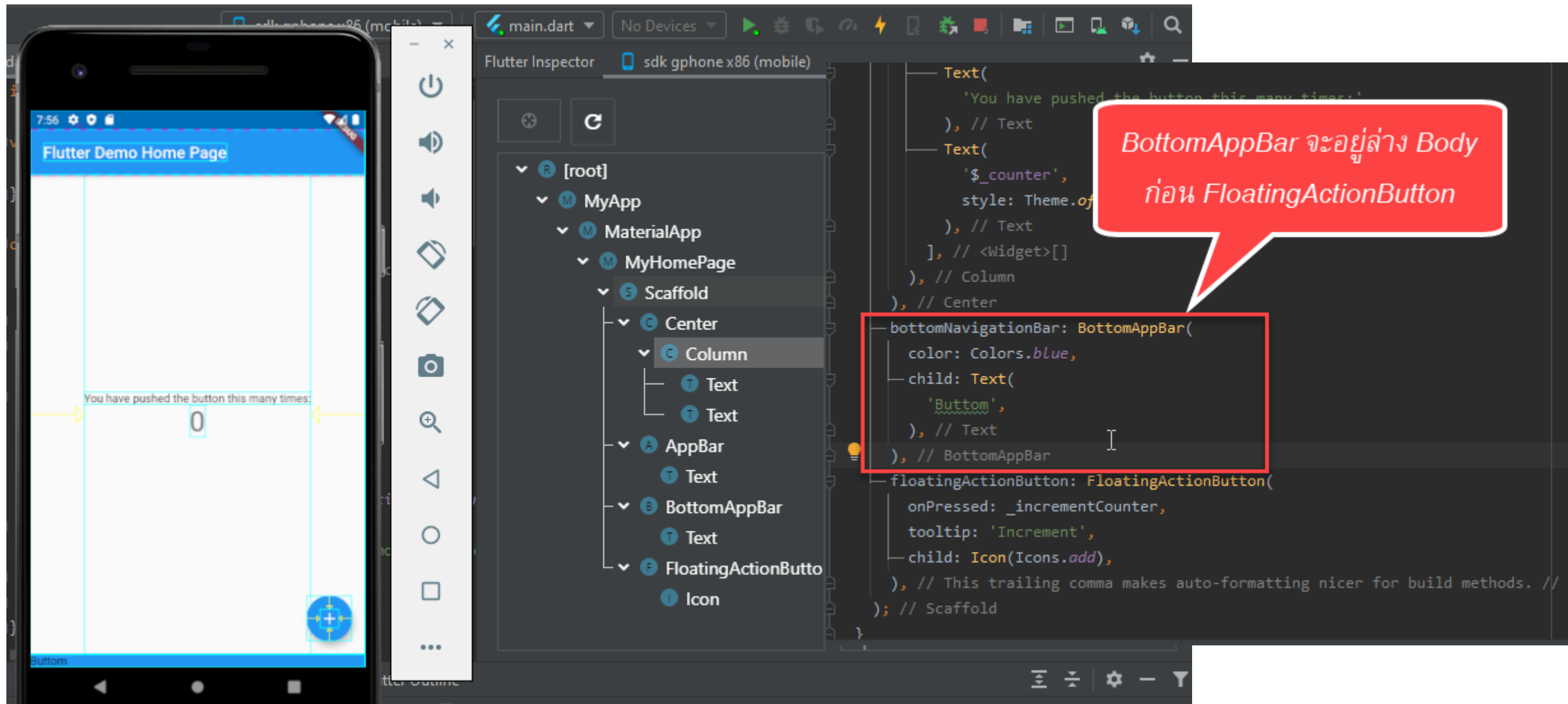
```
main.dart × Page2.dart × Page1.dart × register.dart × pubspec.yaml × widget_test.dart ×
1  import 'package:flutter/material.dart';
2  class Page1 extends StatefulWidget {
3    Page1({Key key, this.title}) : super(key: key);
4
5    final String title;
6
7    @override
8    _Page1State createState() => _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('เปิดหน้าที่สอง'),
22            onPressed: () {
23              // เปิดหน้าที่สองเมื่อมีการกดปุ่ม
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)

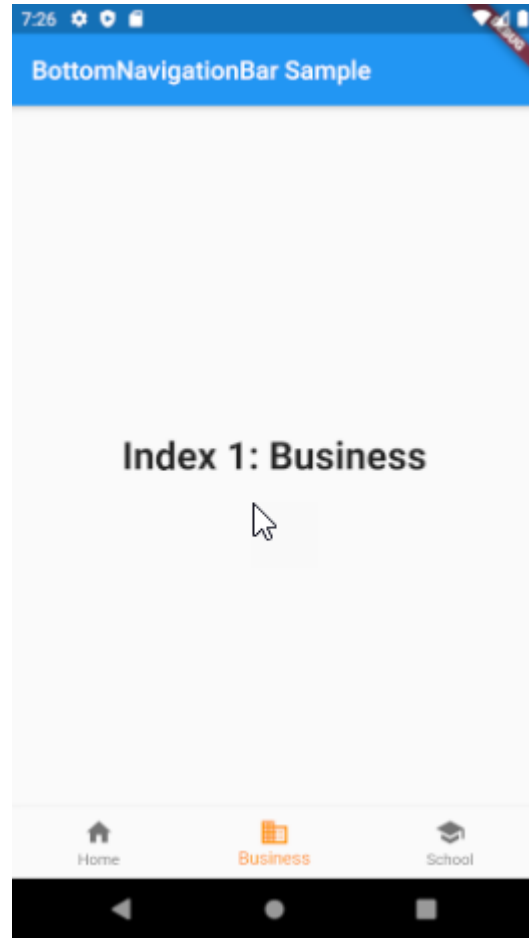
- Page2.dart

```
main.dart x Page2.dart x Page1.dart x register.dart x pubspec.yaml x widget_test.dart x
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('เปิดหน้าทีหนึ่ง'),
22              onPressed: () {
23                // เปิดหน้าที่สองเมื่อมีการกดปุ่ม
24                Navigator.pushNamed(context, '/');
25              },
26            ), // RaisedButton
27          ), // This trailing comma makes auto-formatting nicer for build methods. // Center
28        ); // Scaffold
29      }
30    }
```

BottomAppBar



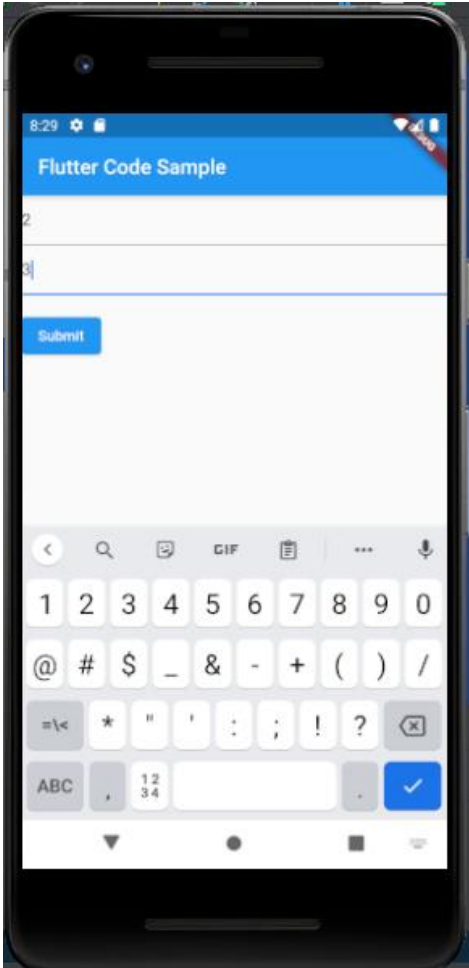
BottomNavigationBar



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

Input Text

- การตั้งค่า input ของ text



สร้างจะใช้ TextFormField และใช้ Controller เป็น id

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

และจะสร้างตัวแปล controller มารับค่า input ที่สร้าง

```
/// This is the private State class that goes with MyStatefulWidget.
class _MyStatefulWidgetState extends State<MyStatefulWidget> {
  final _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,
```

Lab Week 2

- ให้ออกแบบหน้าจอให้แสดงผลดังนี้
- และให้เขียน Tree (Flutter Anatomy) ในการสร้าง

