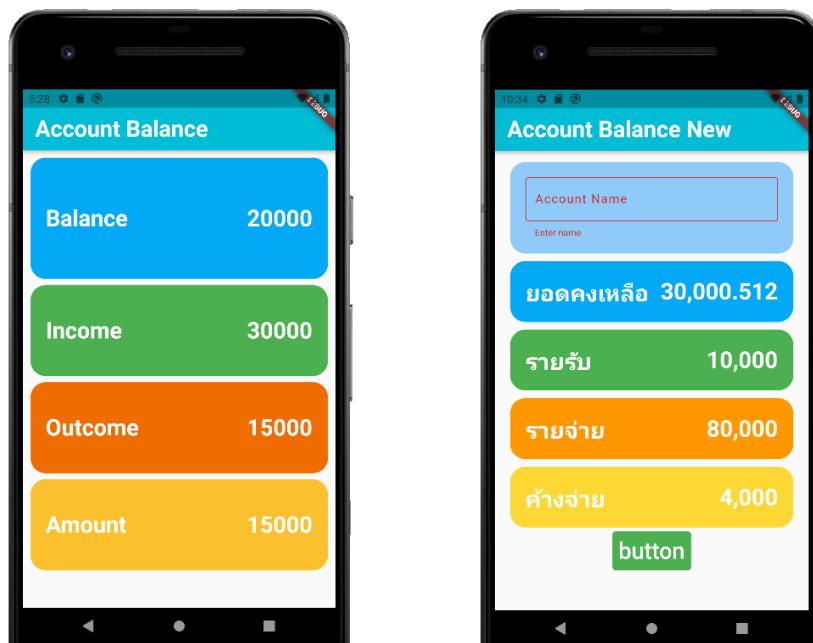


Basic Flutter Mobile – Container Widget



Container Widget (กล่องใส่ของ)

คือ Widget ที่ใช้กำหนดพื้นที่ สามารถกำหนดขนาด รูปทรง หรือจัดวางตำแหน่ง รวมไปถึงการกำหนดลวดลายและตกแต่งด้วยสีให้สวยงามได้ ปกติตัว Container จะไม่แสดงผลขึ้นมาให้เห็น เราจะต้องกำหนดความสูงลงไปและขนาดของ Container ก็จะขยายเต็มพื้นที่ ใช้ในการจัดวาง UI ให้แสดงผลเหมาะสมกับหน้าจอ



BoxDecoration

สำหรับกำหนดรูปทรงของ Container เช่น สี ลักษณะรูปทรงของ Container เป็นต้น

Padding

คือ การกำหนดระยะห่างของ Widget ออกจากขอบของ Layout

Columns Widget

คือ Widget ที่ใช้แสดง Widget ย่อย (Child Widget) ในแนวตั้ง

Row Widget

คือ Widget ที่ใช้แสดง Widget ย่อย (Child Widget) ในแนวนอน มีการกำหนดคุณสมบัติคล้ายๆกับ Columns Widget แต่ต่างกันที่รูปแบบจัดเรียง

Expanded Widget

Widget ที่ใช้ขยายความสูงของ Widget ย่อย (Child Widget) ที่ต้องการ โดยทำการจัดสรรพื้นที่ของ Widget ย่อยให้เต็มพื้นที่โดยดูจากพื้นที่ว่างที่เหลือ ในหน้าจอ (คล้ายๆกับ Flexbox ใน css) และมีการกำหนดพื้นที่ผ่าน Flex Properties

ประเภทของ Widget

Visible Widget คือ สามารถมองเห็นได้บนหน้าจอ เช่น Text , Image , Radio , ฯลฯ

Invisible Widget คือ ไม่สามารถมองเห็นได้บนหน้าจอ แต่ใช้เพื่อควบคุมโครงสร้างของ Widget อีน ๆ เช่น Row , Column , Stack , ListView ฯลฯ

ซึ่ง Container Widget จะเป็นได้ทั้ง Visible และ Invisible เพราะโดยปกติ Container จะไม่แสดงผลให้เห็นบนหน้าจอ แต่สามารถกำหนด Properties เช่น สีพื้นหลัง , ระยะห่าง , ความกว้าง , สูง ให้กับ Container เพื่อแสดงผลบนหน้าจอได้

การจัดตำแหน่ง Widget ประเภท Layout

Widget ที่ทำหน้าที่จัดตำแหน่งให้กับ Widget ตัวอื่น ๆ เรียกว่า Layout Widget แบ่งได้เป็น 2 ประเภท

1. Single-child Layout widget คือ สามารถมี Widget ย่อยได้ 1 Widget เช่น Container , Padding , Center คือ การส่งพารามิเตอร์เพื่อสร้าง widget นี้ จะกำหนด widget ย่อยไปให้กับ child :

`Container (`

`child: Text('Single Child Widget'),`

`);`

2. Multi-child Layout widget คือ Widget ที่ทำหน้าที่จัดตำแหน่ง Widget อีน ๆ โดยภายในจะประกอบไปด้วย Widget ต่าง ๆ เช่น Row , Column , Stack เป็นต้น โดยเมื่อทำการส่งพารามิเตอร์เพื่อกำหนด Widget ย่อย จะต้องทำเป็น List โดยส่งพารามิเตอร์ไปยัง children

`Container (`

`children: [`

`Text('Hello child1'),`

`Text('Hello child2'),`

`Text('Hello child3'),`

`],`

`);`

การใช้งาน Container

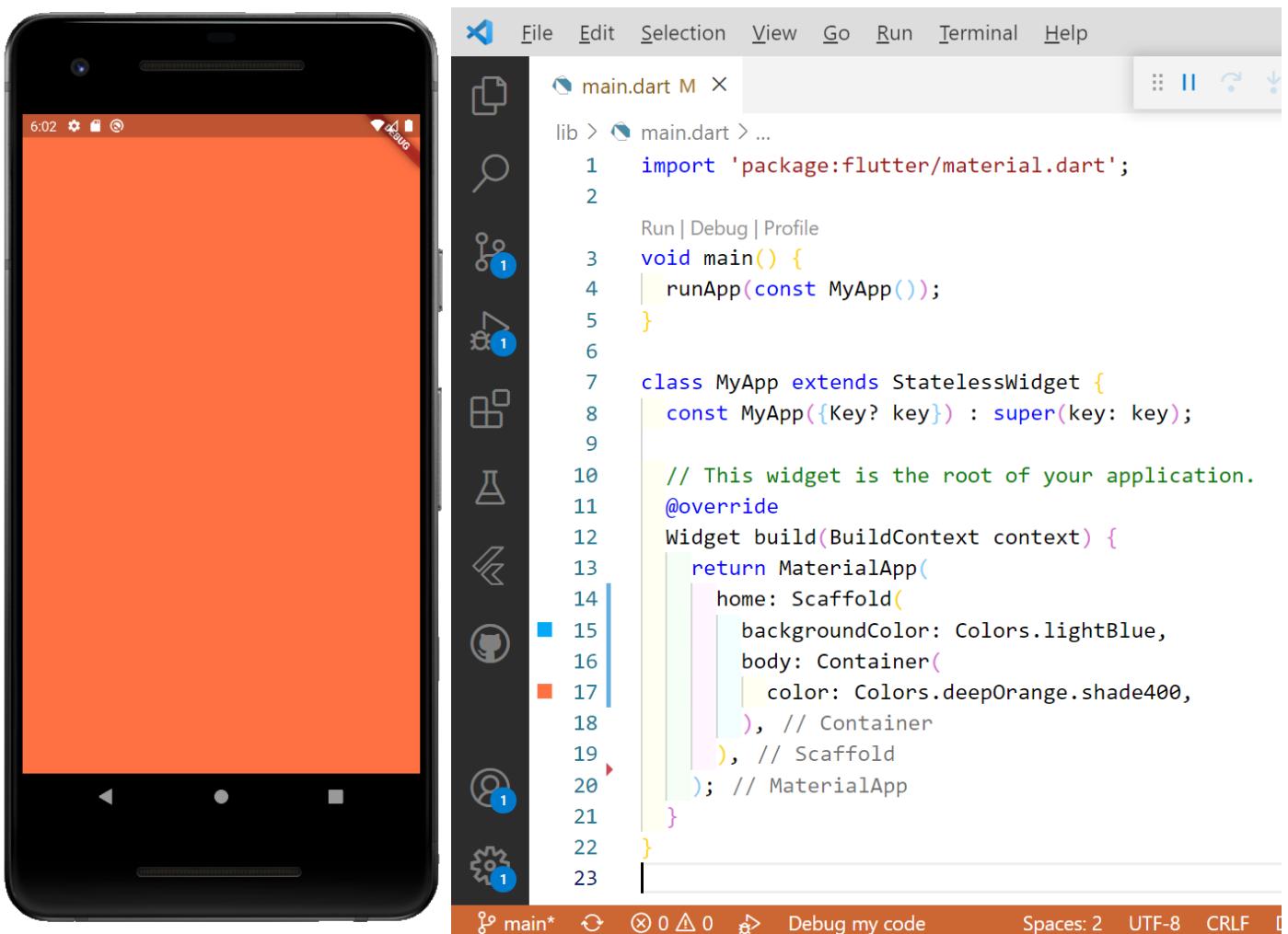
ใช้ในการตกแต่งหรือกำหนดขนาดให้กับ Widget ที่อยู่ภายใน Container เป็นประเภท Single-child

ทำให้สามารถกำหนด Widget ลูกได้เพียง Widget เดียวเท่านั้น โดยมี Properties หลักที่ใช้งานกับ Container ดังนี้

`width , height , margin , padding , color , child`

โดยการใช้งาน Container มีกฏพื้นฐาน 2 ประการ คือ

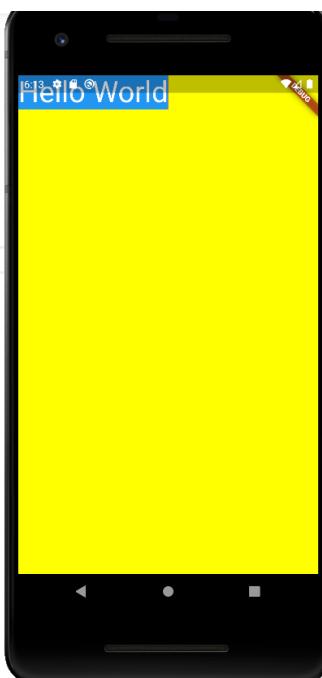
1.Container จะขยายขนาดใหญ่ที่สุดเท่าที่จะเป็นไปได้



ผลลัพธ์จะเห็นได้ว่า มีการแสดงผลของสีส้ม ซึ่งเป็นสีของ container เท่านั้น หมายความว่า Container ถูกขยายเต็มหน้าจอ

2.ขนาดของ Child Widget ที่อยู่ภายใต้ Container จะมีขนาดเล็กที่สุด เพื่อให้สามารถแสดงข้อมูล Widget นั้นได้

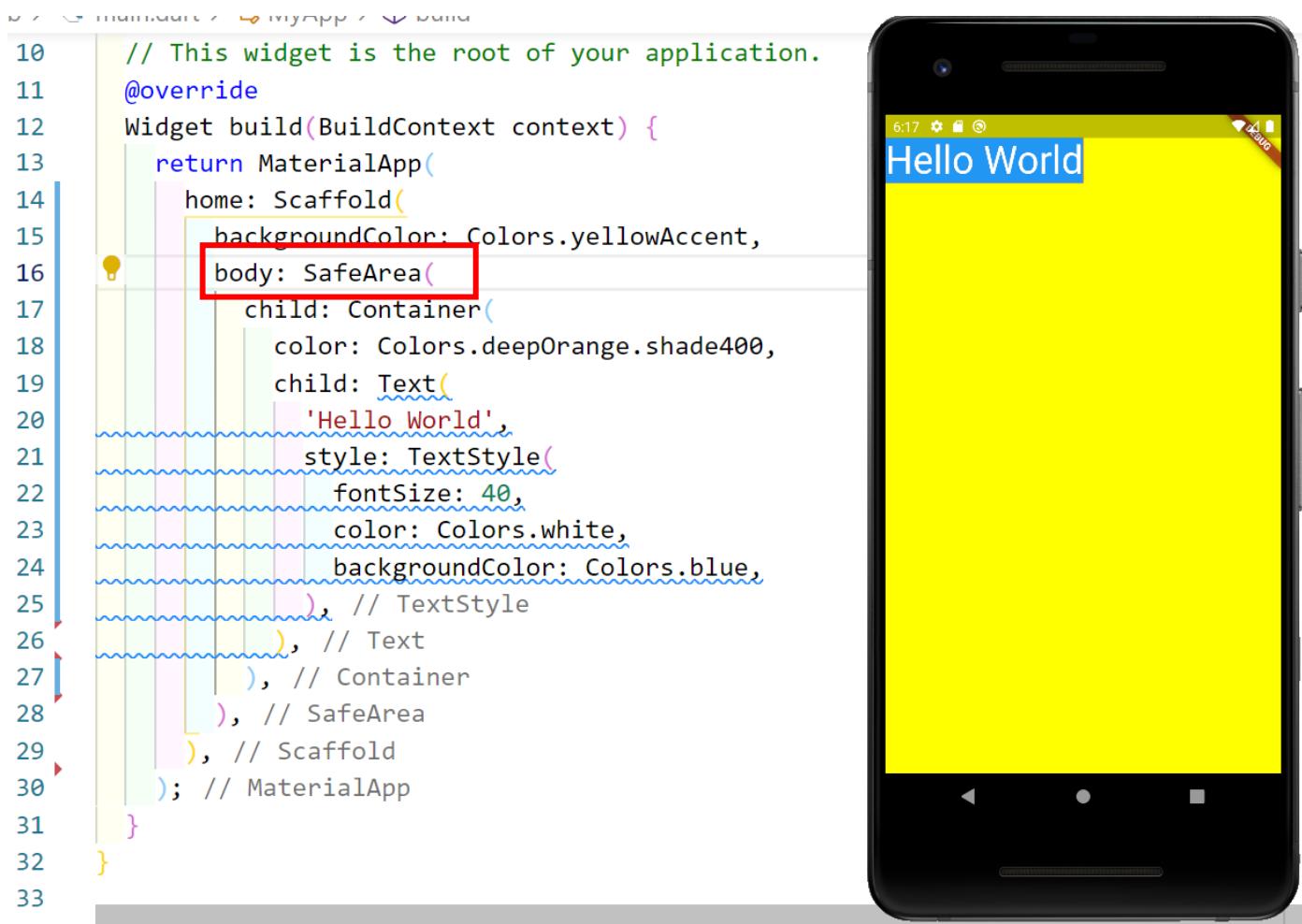
```
7 class MyApp extends StatelessWidget {
8     const MyApp({Key? key}) : super(key: key);
9
10    // This widget is the root of your application.
11    @override
12    Widget build(BuildContext context) {
13        return MaterialApp(
14            home: Scaffold(
15                backgroundColor: Colors.yellowAccent,
16                body: Container(
17                    color: Colors.deepOrange.shade400,
18                    child: Text(
19                        'Hello World',
20                        style: TextStyle(
21                            fontSize: 40,
22                            color: Colors.white,
23                            backgroundColor: Colors.blue,
24                        ), // TextStyle
25                    ), // Text
26                ), // Container
27            ), // Scaffold
28        ); // MaterialApp
29    }
30 }
```



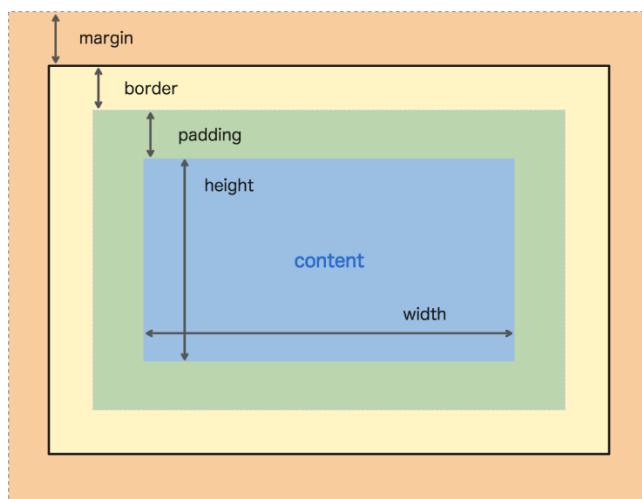
การใช้งาน Widget SafeArea

คือ การแสดงผลข้อมูล เช่น เครื่องข่าย เวลา แบตเตอรี่ ความแรงสัญญาณ จะอยู่ที่ด้านบนของ หน้าจอ ดังนั้น ไม่ควรจะให้มีการแสดงผลของ Widget อื่น ๆ ปรากฏ

การใช้งาน SafeArea ก็เพื่อกำหนดไม่ให้ Widget ต่าง ๆ ไปแสดงผลในบริเวณที่จะแสดงข้อมูลของ SmartPhone



Container และ Layout Widget ต่าง ๆ จะใช้การจัดตำแหน่ง ในแบบ Box Object Model เมื่อเทียบกับ Webpage



Container(
color: ,
padding: ,
margin: ,
child:
border: ,
)

สามารถกำหนด properties ต่าง ๆ ตามความต้องการ

Object ที่ทำการกำหนด Properties Margin และ Padding เป็น Object ที่สร้างมาจาก Class EdgeInsetsGeometry ที่ใช้สำหรับการจัดตำแหน่งใน Container Widget

โดย Class EdgeInsetsGeometry มี Constructor หลายรูปแบบให้เลือกใช้งานได้ดังนี้

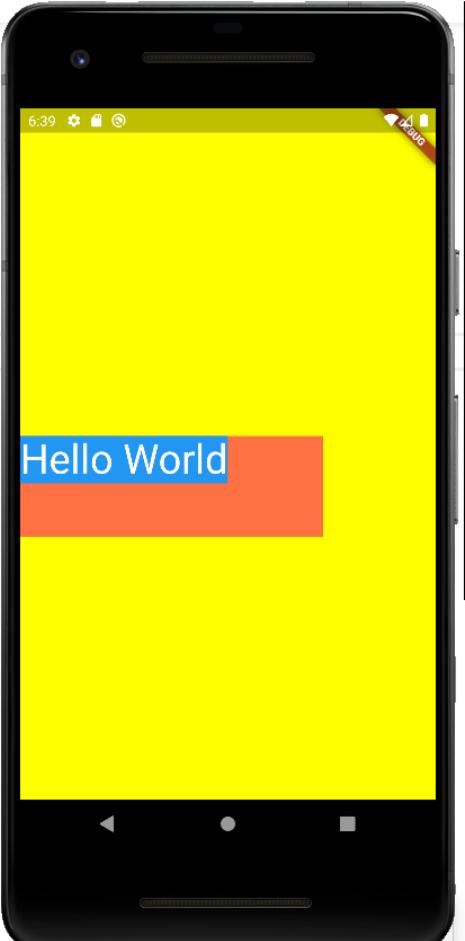
- EdgeInsets.all() : กำหนดให้ระยะห่างทั้ง 4 ด้าน เท่ากัน
: EdgeInsets.all(35.0)
- EdgeInsets.symmetric() : กำหนดให้ระยะห่างแนวอน ด้านบน เท่ากับ ด้านล่าง และ ด้านซ้าย เท่ากับ ด้านขวา
: EdgeInsets.symmetric(vertical:10.0 , horizontal:30.0)
- EdgeInsets.fromLTRB() : กำหนดให้ระยะห่าง ซ้าย บน ขวา ล่าง ตามลำดับ
: EdgeInsets.fromLTRB(2.0,3.0,4.0,5.0)
- EdgeInsets.only() : กำหนดให้ระยะห่าง เฉพาะด้านที่กำหนดเท่านั้น
: EdgeInsets.only(left:4.0)



The screenshot shows an Android emulator displaying a simple application. The background is yellow. In the center, there is a blue rectangular container holding the text "Hello World". The code in the editor corresponds to this setup.

```
10 // This widget is the root of your application.
11
12 @override
13 Widget build(BuildContext context) {
14   return MaterialApp(
15     home: Scaffold(
16       backgroundColor: Colors.yellowAccent,
17       body: SafeArea(
18         child: Container(
19           color: Colors.deepOrange.shade400,
20           margin: EdgeInsets.only(top: 300),
21           child: Text(
22             'Hello World',
23             style: TextStyle(
24               fontSize: 40,
25               color: Colors.white,
26               backgroundColor: Colors.blue,
27             ), // TextStyle
28             ), // Text
29             ), // Container
30             ), // SafeArea
31             ), // Scaffold
32           ); // MaterialApp
33 }
```

การกำหนดขนาด Container ด้วย width และ height



The screenshot shows an Android emulator running an application. The background is yellow. A red rectangular box highlights a red bar at the top of the screen. On this bar, the text "Hello World" is displayed in white. The phone's status bar at the top shows the time as 6:39 and various icons.

```
10 // This widget is the root of your application.
11
12 @override
13 Widget build(BuildContext context) {
14     return MaterialApp(
15         home: Scaffold(
16             backgroundColor: Colors.yellowAccent,
17             body: SafeArea(
18                 child: Container(
19                     width: 300,
20                     height: 100,
21                     color: Colors.deepOrange.shade400,
22                     margin: EdgeInsets.only(top: 300),
23                     child: Text(
24                         'Hello World',
25                         style: TextStyle(
26                             fontSize: 40,
27                             color: Colors.white,
28                             backgroundColor: Colors.blue,
29                         ), // TextStyle
30                     ), // Container
31                 ), // SafeArea
32             ), // Scaffold
33         ); // MaterialApp
```

ทดสอบการสร้าง container ลบโค้ดส่วนที่แสดงเมนูอาหาร ใน listView ออกจากโปรแกรม สำหรับเตรียมสร้าง Container

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Editor:** The main editor window displays the file `main.dart` under the `lib` folder. The code is a `Scaffold` widget with an `AppBar` and a `ListView.builder` for displaying food items. A yellow warning icon is visible near line 58.
- Explorer:** Shows the project structure with files like `gradlew`, `local.properties`, `settings.gradle`, `testapp1_android.iml`, `assets`, `build`, `ios`, `lib` (containing `foodMenu.dart`, `main_ver1.dart`, `main.dart`, `mainFoodMenu.dart`), `test`, `widget_test.dart`, `web`, `.gitignore`, `.metadata`, `.packages`, `analysis_options.yaml`, `pubspec.lock`, `pubspec.yaml`, and `README.md`.
- Bottom Status Bar:** Shows the current line (Ln 52, Col 13), character count (903 selected), and other system information.

เพิ่ม Column Widget ในส่วนของ body

เพื่อนำ widget ที่เป็น children ให้เรียกันในแนวตั้ง สำหรับองรับการเพิ่ม widget ต่าง ๆ เข้าไป

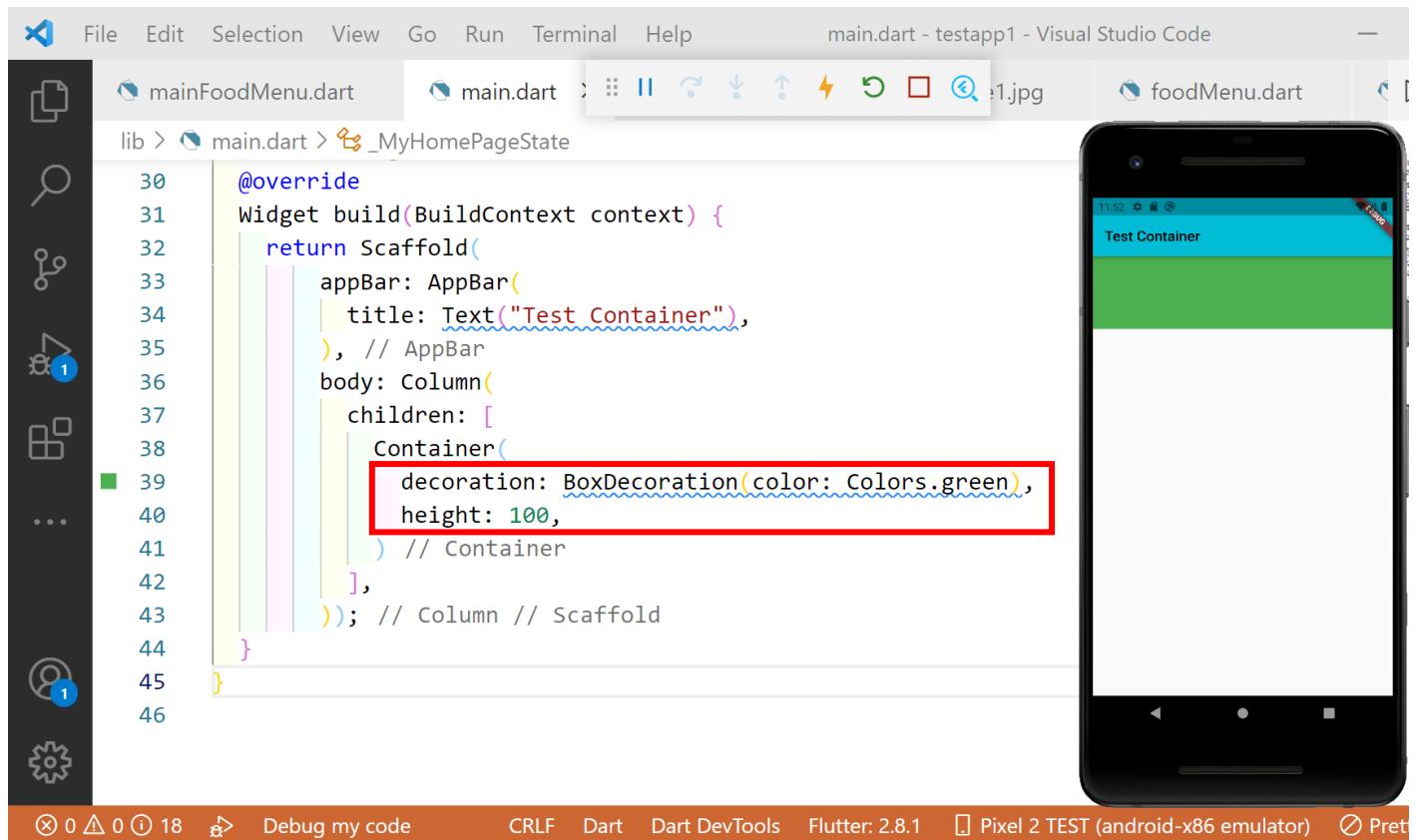
จากนั้น ให้ทดสอบ Run App ใน Emulator

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Editor:** The main editor window displays the file `main.dart` under the `lib` folder. The code defines a `_MyHomePageState` class with a `build` method that returns a `Scaffold` with an `AppBar` and a `body` section containing a `Column` widget. The `children` property of the `Column` is highlighted with a red box. The code also includes a `Test Container` in the emulator preview.
- Explorer:** Shows the project structure with files like `gradlew`, `local.properties`, `settings.gradle`, `testapp1_android.iml`, `assets`, `build`, `ios`, `lib` (containing `foodMenu.dart`, `main_ver1.dart`, `main.dart`, `mainFoodMenu.dart`), `test`, `widget_test.dart`, `web`, `.gitignore`, `.metadata`, `.packages`, `analysis_options.yaml`, `pubspec.lock`, `pubspec.yaml`, and `README.md`.
- Bottom Status Bar:** Shows the current line (Ln 41, Col 13), character count (18), and other system information.

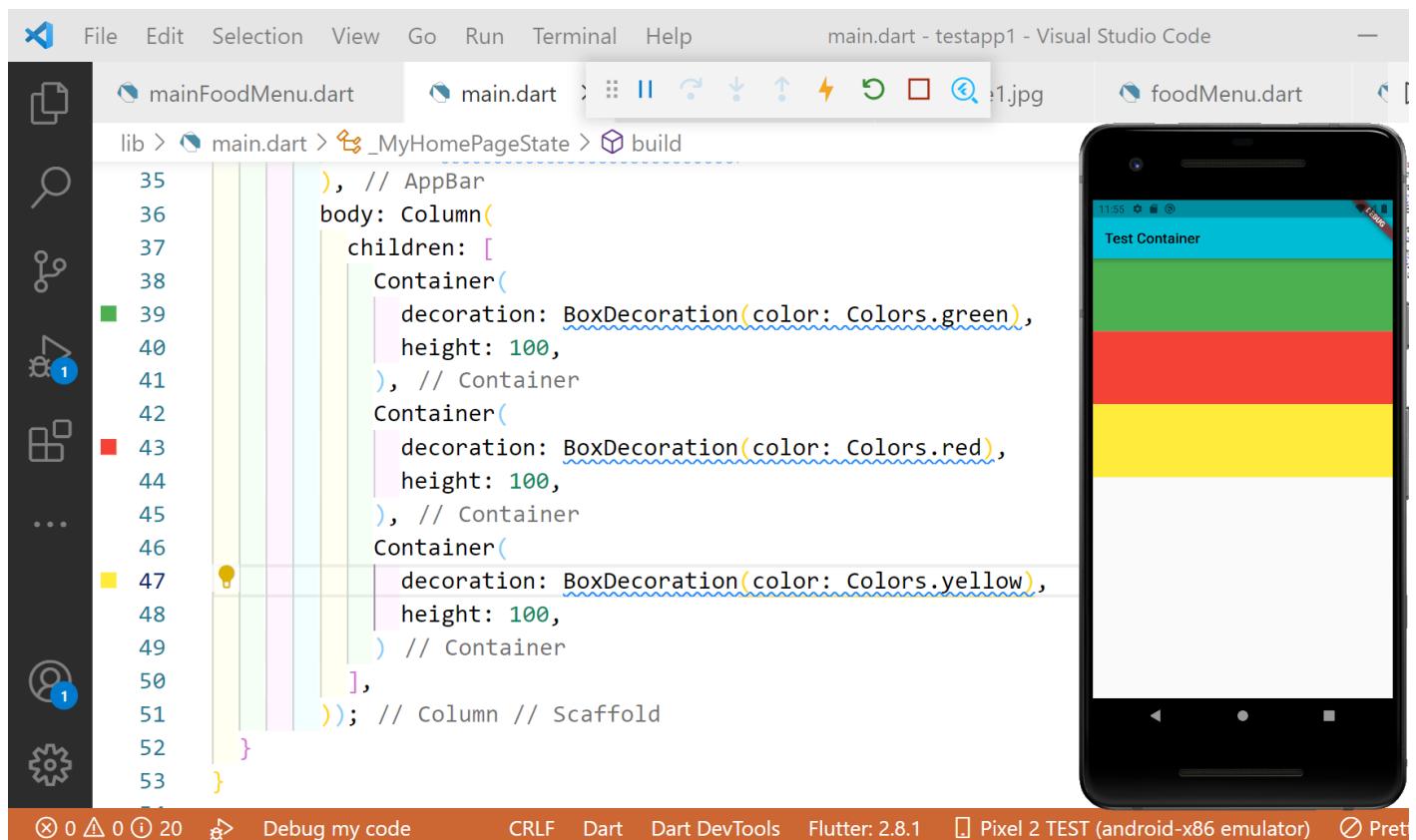
เพิ่ม Container Widget ใน children

โดยเพิ่มการกำหนดรูปแบบหรือลักษณะของ container ด้วย Attribute ใน box decoration และกำหนดความสูงของ Container ด้วย height แล้วทดสอบการ run app



```
30 @override
31 Widget build(BuildContext context) {
32   return Scaffold(
33     appBar: AppBar(
34       title: Text("Test Container"),
35     ), // AppBar
36     body: Column(
37       children: [
38         Container(
39           decoration: BoxDecoration(color: Colors.green),
40           height: 100,
41         ), // Container
42         ],
43       ); // Column // Scaffold
44     }
45   }
46 }
```

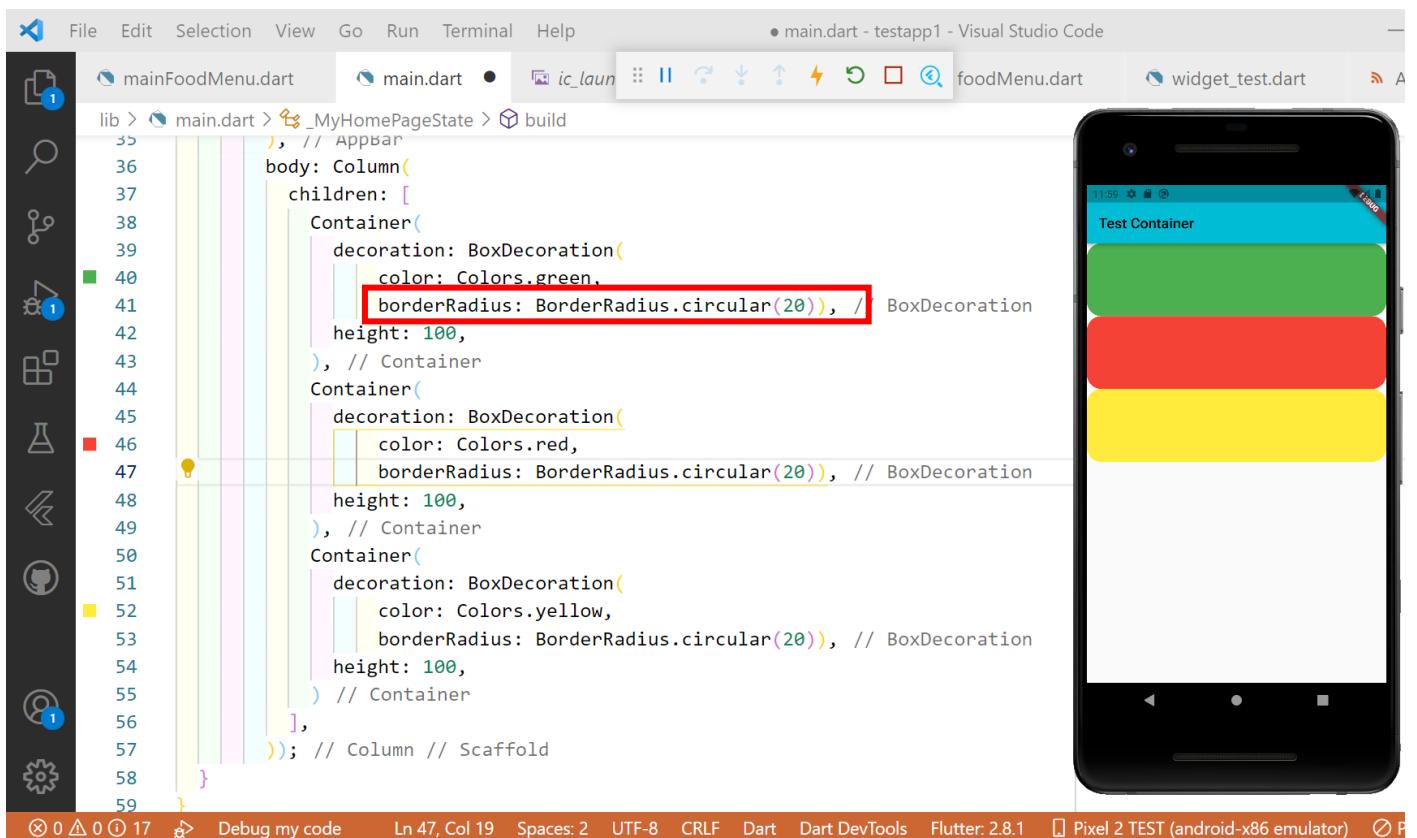
ทดสอบการเพิ่ม container



```
35   ), // AppBar
36   body: Column(
37     children: [
38       Container(
39         decoration: BoxDecoration(color: Colors.green),
40         height: 100,
41       ), // Container
42       Container(
43         decoration: BoxDecoration(color: Colors.red),
44         height: 100,
45       ), // Container
46       Container(
47         decoration: BoxDecoration(color: Colors.yellow),
48         height: 100,
49       ), // Container
50     ],
51   ); // Column // Scaffold
52 }
53 }
```

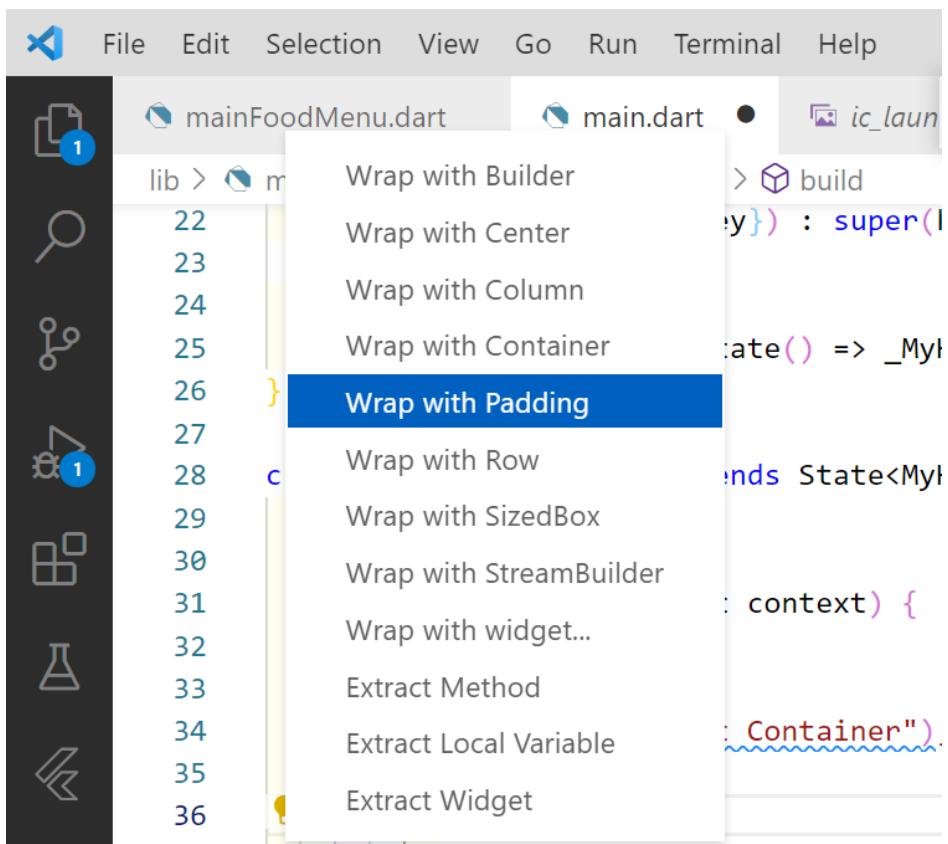
ทำให้ Container มีมุมที่โค้งมน

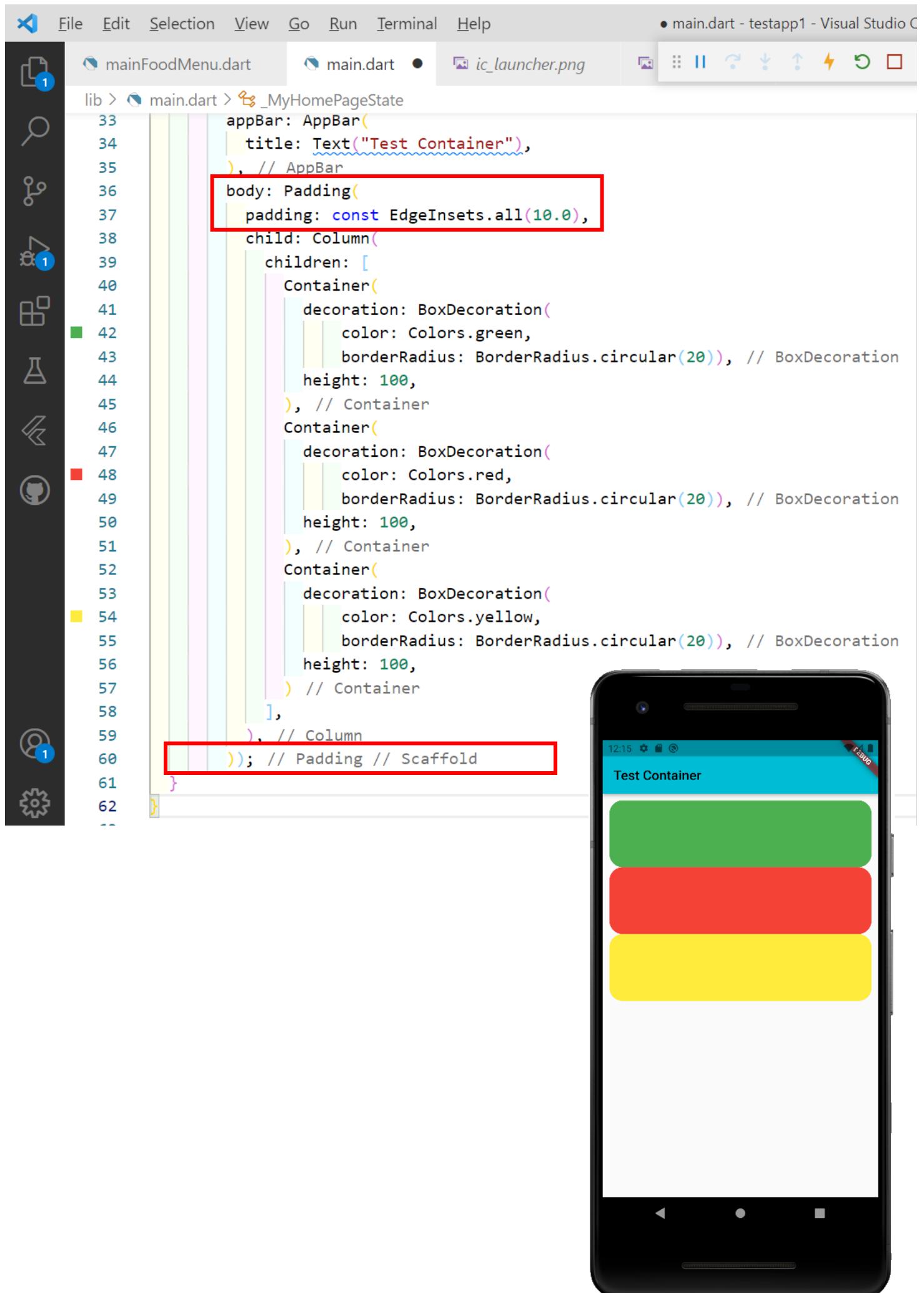
โดยกำหนด Attribute คือ borderRadius เพื่อให้ขอบมุม มีความโค้งมน



การใช้ Padding

เพื่อการจัดวางตำแหน่งโดยกำหนดระยะห่างจากขอบของหน้าจอ โดยกำหนดให้คลุมในส่วนของ column





File Edit Selection View Go Run Terminal Help

• main.dart - testapp1 - Visual Studio C

mainFoodMenu.dart main.dart ic_launcher.png

```
lib > main.dart > _MyHomePageState
  33   appBar: AppBar(
  34     title: Text("Test Container"),
  35   ), // AppBar
  36   body: Padding(
  37     padding: const EdgeInsets.all(10.0),
  38     child: Column(
  39       children: [
  40         Container(
  41           decoration: BoxDecoration(
  42             color: Colors.green,
  43             borderRadius: BorderRadius.circular(20)), // BoxDecoration
  44             height: 100,
  45           ), // Container
  46           Container(
  47             decoration: BoxDecoration(
  48               color: Colors.red,
  49               borderRadius: BorderRadius.circular(20)), // BoxDecoration
  50               height: 100,
  51             ), // Container
  52             Container(
  53               decoration: BoxDecoration(
  54                 color: Colors.yellow,
  55                 borderRadius: BorderRadius.circular(20)), // BoxDecoration
  56                 height: 100,
  57               ), // Container
  58             ],
  59           ), // Column
  60         ); // Padding // Scaffold
  61     }
  62 --
```

การใช้ Row Widget

ในกรณีจะให้แสดง widget ใน Container แต่ละตัว ในแนวนอน

เพิ่มข้อความสำหรับการแสดงผล ในจัดเรียงในแนวนอน ใน container แต่ละตัว

The screenshot shows the Visual Studio Code interface with the file `mainFoodMenu.dart` open. The code is as follows:

```
38     child: Column(
39       children: [
40         Container(
41           decoration: BoxDecoration(
42             color: Colors.green,
43             borderRadius: BorderRadius.circular(10),
44             height: 100,
45             child: Row(
46               children: [
47                 Text("Balance"),
48                 Text("20000")
49               ],
50             ), // Row
51           ), // Container
52           Container(
```

A red box highlights the `Row` widget and its children (`Text` widgets). To the right, an iPhone simulator displays the app's UI with three colored containers (green, red, yellow) each containing the text "Balance" and "20000".

เพิ่ม padding เพื่อเว้นระยะห่างของข้อความ จากขอบ App

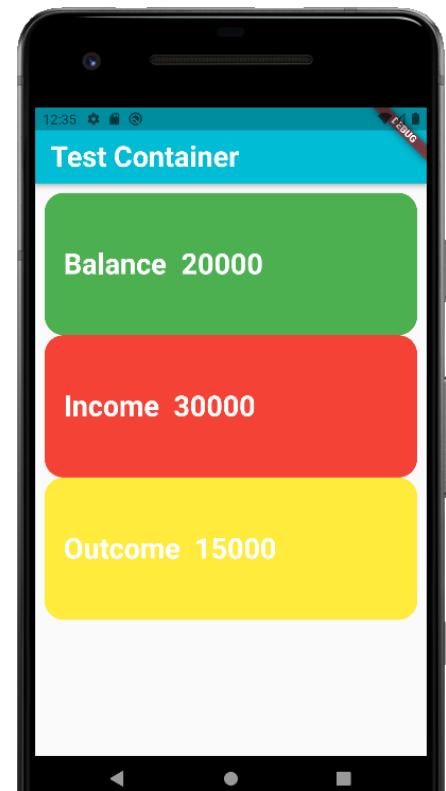
The screenshot shows the Visual Studio Code interface with the file `mainFoodMenu.dart` open. The code is as follows:

```
38     child: Column(
39       children: [
40         Container(
41           padding: const EdgeInsets.all(20.0),
42           decoration: BoxDecoration(
43             color: Colors.green,
44             borderRadius: BorderRadius.circular(10),
45             height: 100,
46             child: Row(
47               children: [
48                 Text("Balance"),
49                 Text("20000")
50               ],
51             ), // Row
52           ), // Container
```

A red box highlights the `padding: const EdgeInsets.all(20.0)` line. To the right, an iPhone simulator displays the app's UI with three colored containers (green, red, yellow) each containing the text "Balance" and "20000", with visible padding around the text.

ให้ทำการกำหนดรูปแบบของ ข้อความ ด้วย TextStyle ตามความต้องการ

```
child: Row(  
  children: [  
    Text(  
      "Balance ",  
      style: TextStyle(  
        fontSize: 30,  
        color: Colors.white,  
        fontWeight: FontWeight.bold), // TextStyle  
    ), // Text  
    Text(  
      " 20000",  
      style: TextStyle(  
        fontSize: 30,  
        color: Colors.white,  
        fontWeight: FontWeight.bold), // TextStyle  
    ) // Text
```



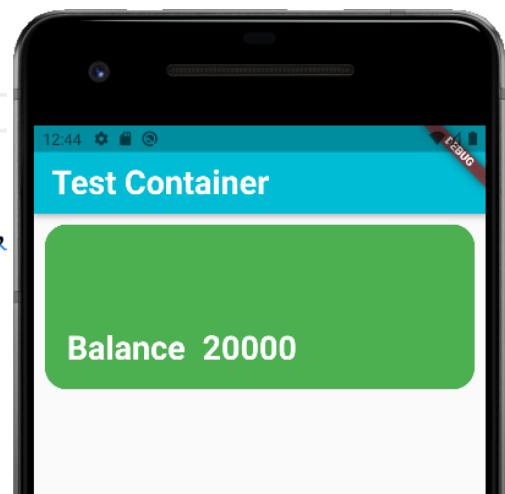
การใช้งาน Attribute CrossAxisAlignmentAlignment

เพื่อจัดวางตำแหน่งของ Widget Children ใน Container

```
44 Container(  
45   padding: const EdgeInsets.all(20.0),  
46   decoration: BoxDecoration(  
47     color: Colors.green,  
48     borderRadius: BorderRadius.circular(20)), // BoxDecoration  
49   height: 150,  
50   child: Row(  
51     crossAxisAlignment: CrossAxisAlignment.end,  
52     children: [  
53       Text(  
54         "Balance ",  
55         style: TextStyle(  
56           fontSize: 30,  
57           color: Colors.white,  
58           fontWeight: FontWeight.bold), // TextStyle  
59     ), // Text
```



```
  child: Row(  
    crossAxisAlignment: CrossAxisAlignment.start,  
    children: [  
      Text(  
        "Balance ",  
        style: TextStyle(  
          fontSize: 30,  
          color: Colors.white,  
          fontWeight: FontWeight.bold), // TextStyle  
    ) // Text
```

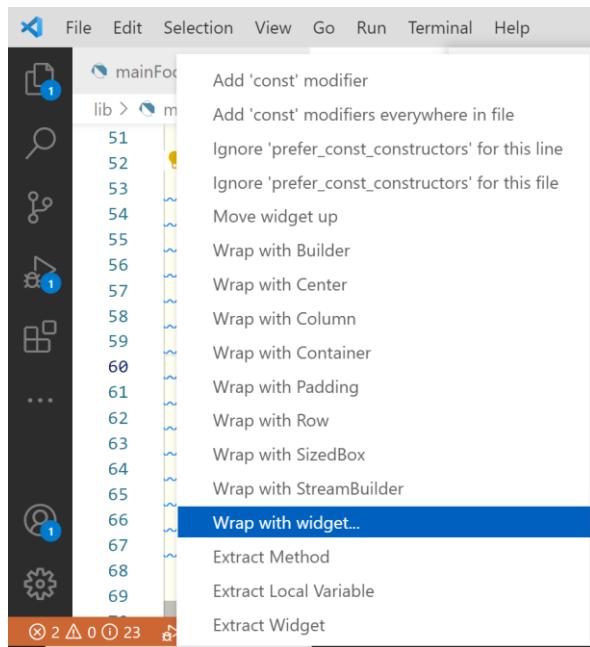


การใช้ Expanded Widget

สำหรับจัดวางให้คำแนะนำข้อความ ด้วย TextAlign

1.เลือกคำแนะนำสำหรับการใส่ Expanded

2.เลือก Wrap with widget * ในขั้นตอนนี้ flutter จะเพิ่ม child มาให้อัตโนมัติ



3.เปลี่ยนชื่อ Widget เป็น Expanded

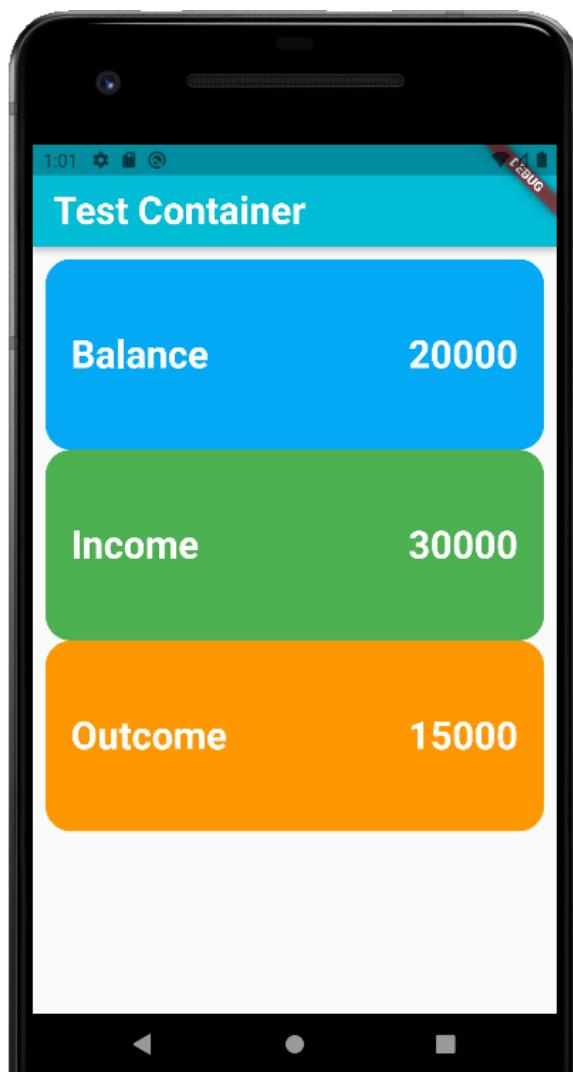
```
lib > main.dart > _MyHomePageState > build
    ...
58     fontStyle: TextStyle(fontWeight: FontWeight.bold), // TextStyle
59     ),
60     Expanded(
61         child: Text(
62             "20000",
63             style: TextStyle(
64                 fontSize: 30,
65                 color: Colors.white,
66                 fontWeight: FontWeight.bold), // TextStyle
67             ), // Text
68         ),
69     ),
70     ),
71     /*, // Row
72     ) /*, // Container
```

4. กำหนด align ให้กับ ข้อความ

The screenshot shows the Visual Studio Code interface with a Dart file named `mainFoodMenu.dart` open. The code is part of a `_MyHomePageState` class and defines a `build` method. A specific line of code, `textAlign: TextAlign.right,`, is highlighted with a red box. To the right of the editor, a mobile device icon displays a Flutter application titled "Test Container". The app shows a green button with the text "Balance 20000". The Visual Studio Code status bar at the bottom indicates "Flutter: 2.8.1" and "Pixel 2 TEST (android-x86 emulator)".

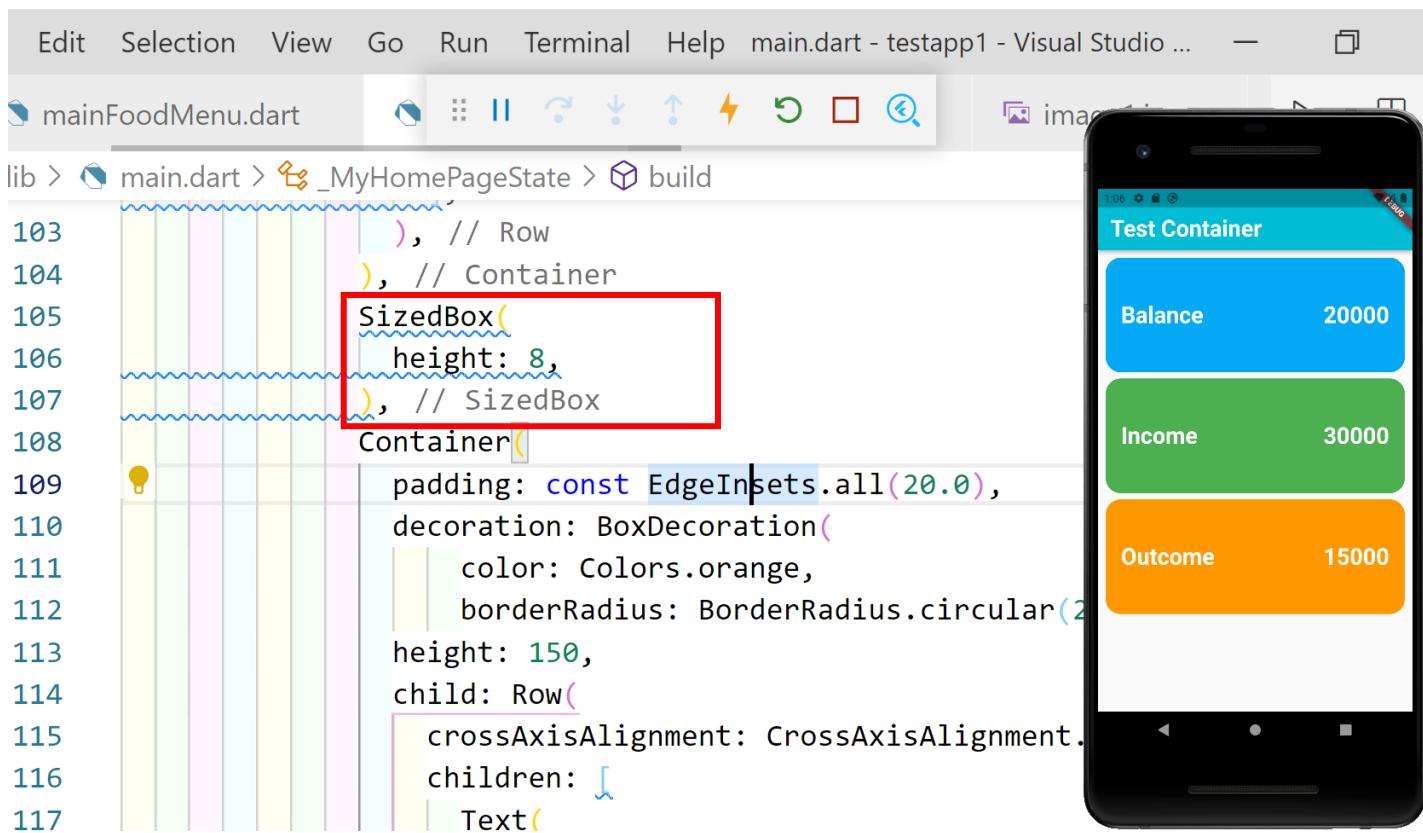
```
lib > mainFoodMenu.dart
lib > main.dart > _MyHomePageState > build
59   ),
60   Expanded(
61     child: Text(
62       " 20000",
63       style: TextStyle(
64         fontSize: 30,
65         color: Colors.white,
66         fontWeight: FontWeight.bold),
67       textAlign: TextAlign.right,
68     ),
69   ),
70   ],
71   ),
72   ),
73   */,
```

เพิ่มการจัดรูปแบบให้กับทุก container



กำหนดระยะห่างใน Container ด้วย SizedBox

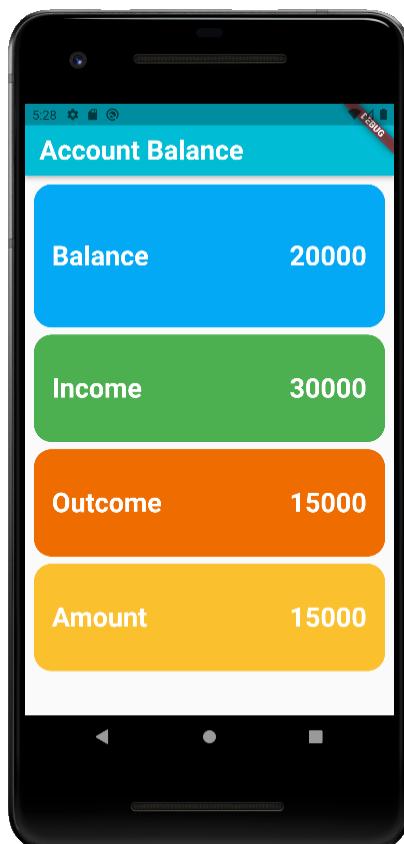
sizeBox คือ การสร้างกล่องเปล่า โดยกำหนดขนาดตามต้องการ เพื่อเว้นระยะห่างในกับ container



```
lib > main.dart > _MyHomePageState > build
103     ), // Row
104   ), // Container
105   SizedBox(
106     height: 8,
107   ), // SizedBox
108   Container(
109     padding: const EdgeInsets.all(20.0),
110     decoration: BoxDecoration(
111       color: Colors.orange,
112       borderRadius: BorderRadius.circular(20.0),
113       height: 150,
114       child: Row(
115         mainAxisAlignment: MainAxisAlignment.spaceBetween,
116         children: [
117           Text("Balance", style: TextStyle(color: Colors.white)),
```

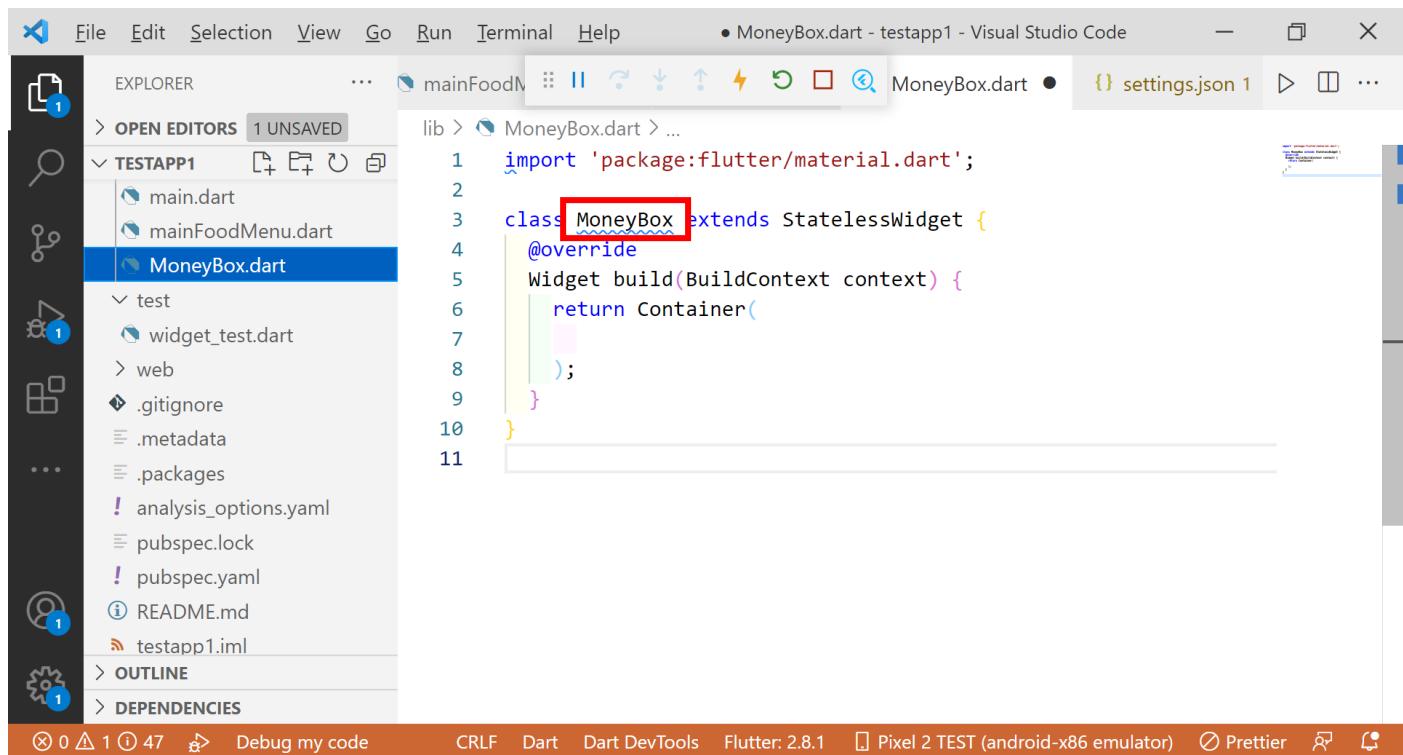
The screenshot shows the code for the `_MyHomePageState` class in a Dart file. The `SizedBox` widget at line 105 is highlighted with a red box. To its right is a preview of the app running on an iPhone, titled "Test Container". It displays four colored cards: blue (Balance 20000), green (Income 30000), orange (Outcome 15000), and yellow (Amount 15000). The cards are separated by vertical gaps created by the `SizedBox`.

สร้าง container เพิ่มอีก 1 ชุด และปรับขนาด Container ให้เหมาะสม



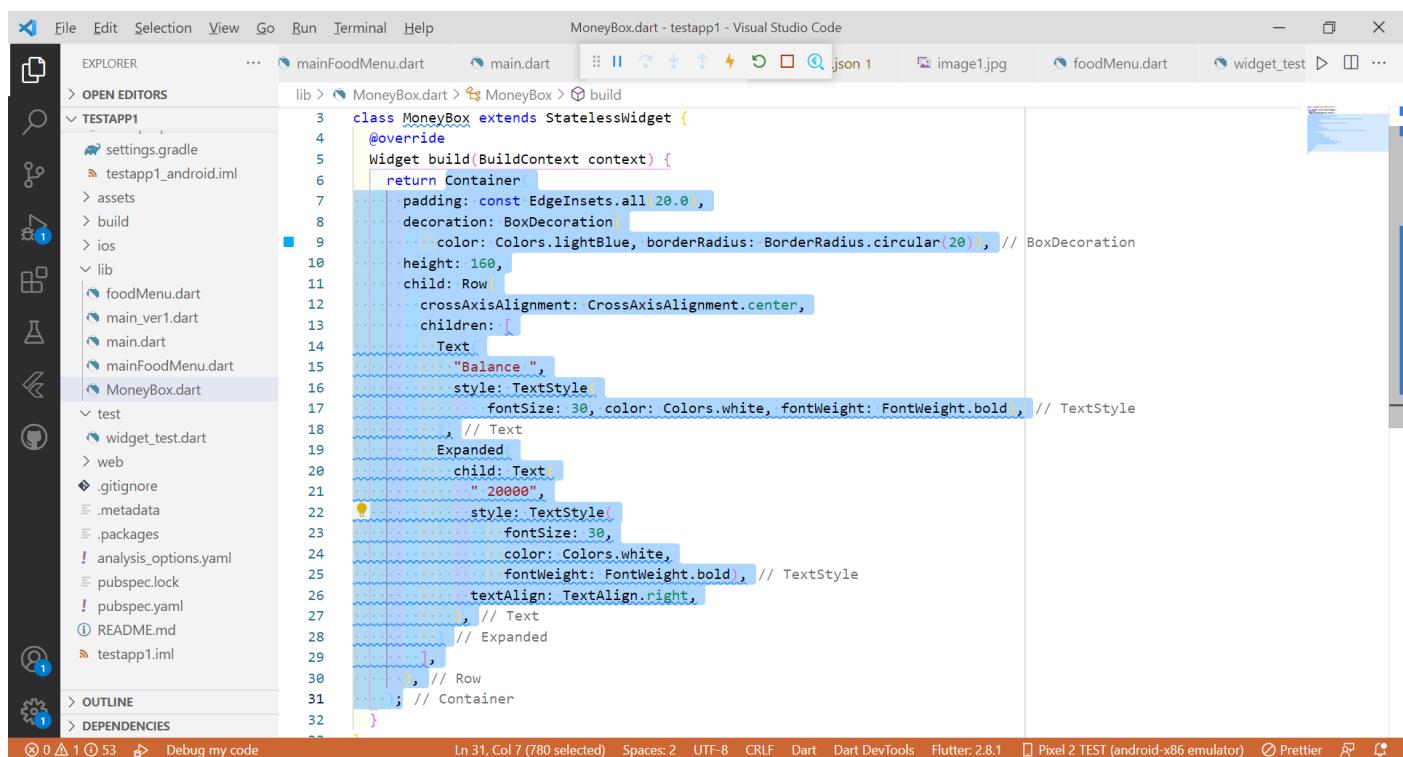
การสร้าง Widget ต้นแบบ

สร้างไฟล์ใหม่ MoneyBox.dart และสร้าง class แบบ StatelessWidget



```
lib > MoneyBox.dart > ...
1 import 'package:flutter/material.dart';
2
3 class MoneyBox extends StatelessWidget {
4   @override
5   Widget build(BuildContext context) {
6     return Container(
7       padding: const EdgeInsets.all(20.0),
8       decoration: BoxDecoration(
9         color: Colors.lightBlue, borderRadius: BorderRadius.circular(20.0)),
10      height: 160,
11      child: Row(
12        mainAxisAlignment: MainAxisAlignment.center,
13        children: [
14          Text("Balance",
15            style: TextStyle(
16              fontSize: 30, color: Colors.white, fontWeight: FontWeight.bold),
17            ),
18          Expanded(
19            child: Text("20000",
20              style: TextStyle(
21                fontSize: 30,
22                color: Colors.white,
23                fontWeight: FontWeight.bold),
24                textAlign: TextAlign.right,
25              ),
26            ),
27          ],
28        ),
29      );
30    }
31  }
32 }
```

นำ Container จาก main.dart มาใส่ใน MoneyBox เพื่อสร้างเป็นต้นแบบ

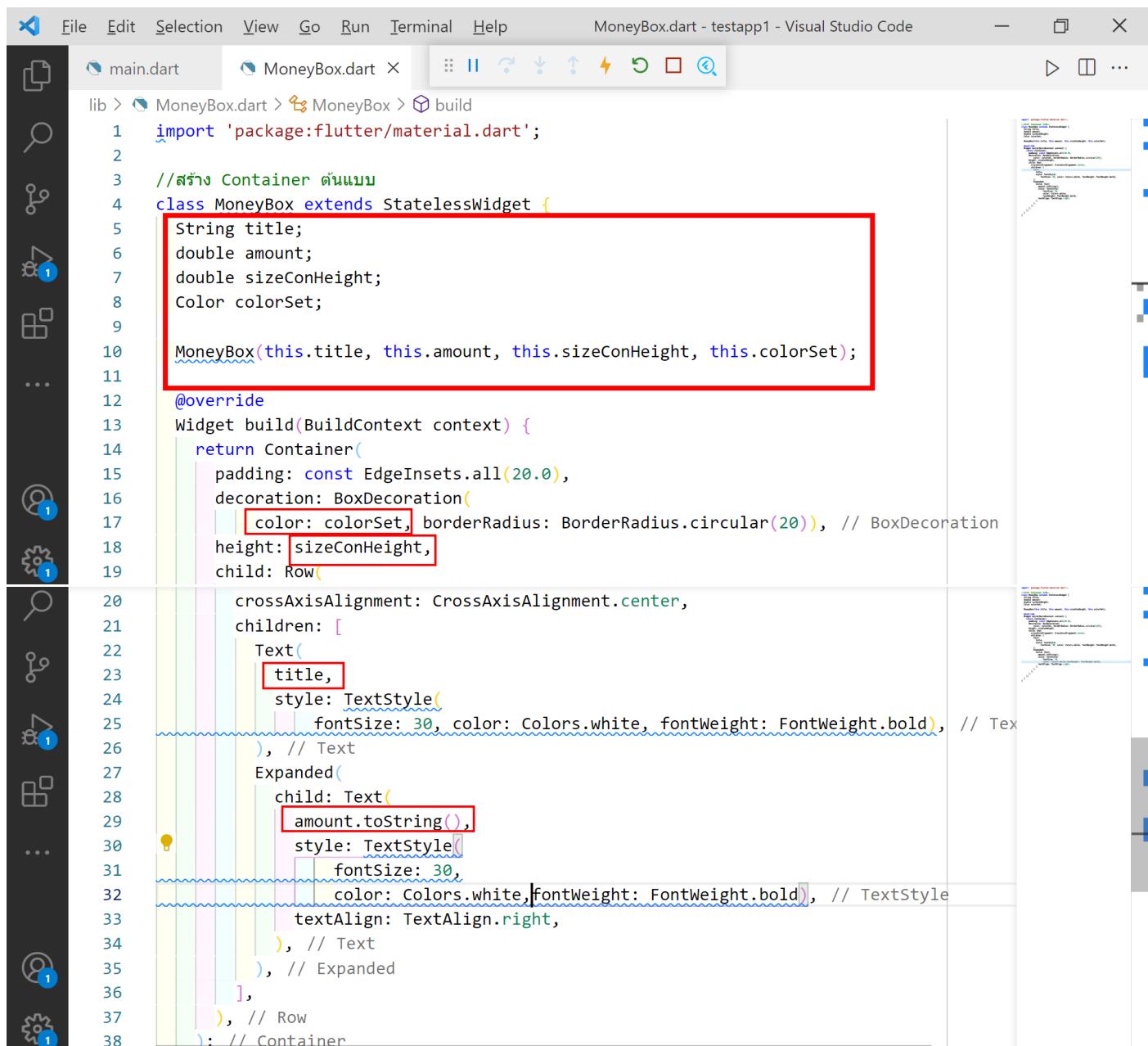


```
lib > MoneyBox.dart > MoneyBox > build
3 class MoneyBox extends StatelessWidget {
4   @override
5   Widget build(BuildContext context) {
6     return Container(
7       padding: const EdgeInsets.all(20.0),
8       decoration: BoxDecoration(
9         color: Colors.lightBlue, borderRadius: BorderRadius.circular(20.0)),
10      height: 160,
11      child: Row(
12        mainAxisAlignment: MainAxisAlignment.center,
13        children: [
14          Text("Balance",
15            style: TextStyle(
16              fontSize: 30, color: Colors.white, fontWeight: FontWeight.bold),
17            ),
18          Expanded(
19            child: Text("20000",
20              style: TextStyle(
21                fontSize: 30,
22                color: Colors.white,
23                fontWeight: FontWeight.bold),
24                textAlign: TextAlign.right,
25              ),
26            ),
27          ],
28        ),
29      );
30    }
31  }
32 }
```

เมื่อต้องการสร้าง container อีน ๆ ก็จะปรับเฉพาะข้อมูลที่ต้องการแสดงผลที่แตกต่างกัน โดยสร้างจาก container ต้นแบบที่ก็ได้กำหนดไว้ใน Class MoneyBox

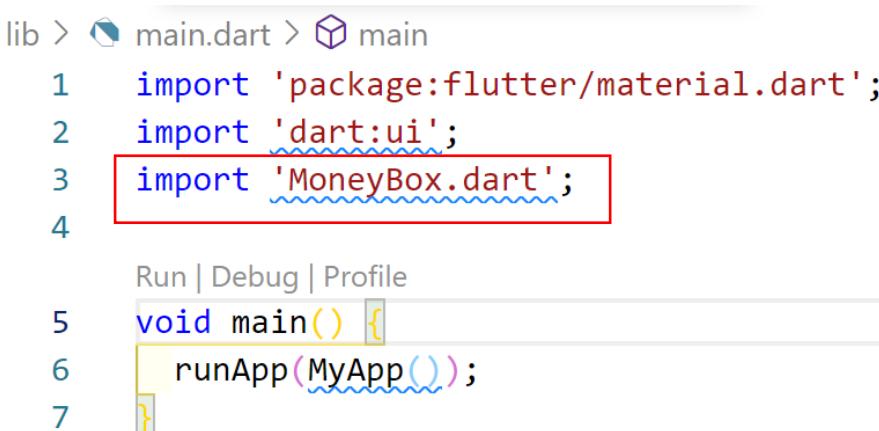
สร้าง Attribute ใน MoneyBox.dart และ constructor เพื่อกำหนดให้ container สามารถแสดงผลได้

แล้วแทนค่าตัวแปรในการสร้าง container ต้นแบบ



```
lib > MoneyBox.dart > build
1 import 'package:flutter/material.dart';
2
3 //สร้าง Container ต้นแบบ
4 class MoneyBox extends StatelessWidget {
5   String title;
6   double amount;
7   double sizeConHeight;
8   Color colorSet;
9
10 MoneyBox(this.title, this.amount, this.sizeConHeight, this.colorSet);
11
12 @override
13 Widget build(BuildContext context) {
14   return Container(
15     padding: const EdgeInsets.all(20.0),
16     decoration: BoxDecoration(
17       color: colorSet, borderRadius: BorderRadius.circular(20)), // BoxDecoration
18     height: sizeConHeight,
19     child: Row(
20       crossAxisAlignment: CrossAxisAlignment.center,
21       children: [
22         Text(
23           title,
24           style: TextStyle(
25             fontSize: 30, color: Colors.white, fontWeight: FontWeight.bold), // TextStyle
26         ), // Text
27         Expanded(
28           child: Text(
29             amount.toString(),
30             style: TextStyle(
31               fontSize: 30,
32               color: Colors.white, fontWeight: FontWeight.bold), // TextStyle
33             textAlign: TextAlign.right,
34           ), // Text
35         ), // Expanded
36       ],
37     ), // Row
38   ); // Container
```

ทำการ import MoneyBox.dart ใน main.dart



```
lib > main.dart > main
1 import 'package:flutter/material.dart';
2 import 'dart:ui';
3 import 'MoneyBox.dart';
4
5 void main() {
6   runApp(MyApp());
7 }
```

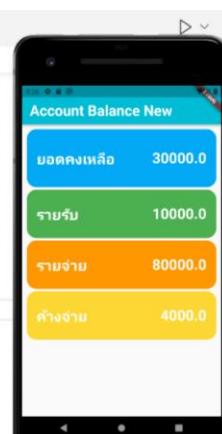
จากนั้น ลบ Container เดิมที่สร้างใน Main.dart และสร้าง container ใหม่จาก class MoneyBox



```
lib > main.dart > _MyHomePageState
  ...
  style: TextStyle(
    ...
  ), // Text
), // AppBar
body: Padding(
  padding: const EdgeInsets.all(10.0),
  child: Column(
    children: [
      MoneyBox("ยอดคงเหลือ", 30000, 120, Colors.lightBlue),
      MoneyBox("รายรับ", 10000, 100, Colors.green),
      MoneyBox("รายจ่าย", 80000, 100, Colors.orange),
      MoneyBox("คงเหลือ", 4000, 100, Colors.yellow.shade600)
    ],
  ), // Column
); // Padding // Scaffold
}
}
```

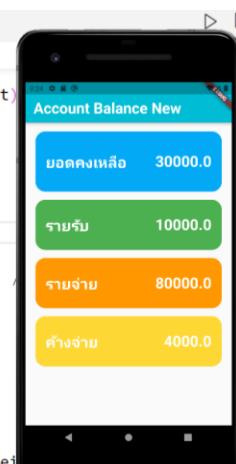
หากต้องการ เว้นระยะห่างให้แต่ละ container จะทำอย่างไรได้บ้าง

1.แทรก SizedBox โดยกำหนดความสูงให้กับกล่อง เพื่อเว้นระยะห่างของ container



```
lib > main.dart > _MyHomePageState > build
  ...
  body: Padding(
    padding: const EdgeInsets.all(10.0),
    child: Column(
      children: [
        MoneyBox("ยอดคงเหลือ", 30000, 120, Colors.lightBlue),
        SizedBox(height: 5),
        MoneyBox("รายรับ", 10000, 100, Colors.green),
        SizedBox(height: 5),
        MoneyBox("รายจ่าย", 80000, 100, Colors.orange),
        SizedBox(height: 5),
        MoneyBox("คงเหลือ", 4000, 100, Colors.yellow.shade600)
      ],
    ), // Column
  ); // Padding // Scaffold
}
}
```

2.กำหนด Margin หรือ padding ใน MoneyBox.dart



```
lib > MoneyBox.dart > MoneyBox > build
  ...
  MoneyBox(this.title, this.amount, this.sizeConHeight, this.colorSet)
  ...
  @override
  Widget build(BuildContext context) {
    return Container(
      padding: const EdgeInsets.all(20.0),
      margin: EdgeInsets.symmetric(vertical: 8, horizontal: 10),
      decoration: BoxDecoration(
        color: colorSet, borderRadius: BorderRadius.circular(20),
        height: sizeConHeight,
        child: Row(
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            Text(
              title,
              style: TextStyle(
                ...
              ),
            )
          ]
        )
      );
  }
}
```

การกำหนดรูปแบบให้กับตัวเลข

เป็นการกำหนด NumberFormat ให้กับข้อมูลที่เป็นตัวเลข

1. ติดตั้ง package intl จาก <https://pub.dev>



int 0.17.0

Published 12 months ago • [dart.dev](#) Null safety

SDK DART FLUTTER PLATFORM ANDROID IOS LINUX MACOS WEB WINDOWS

Readme Changelog [Installing](#) Versions Scores

Use this package as a library

Depend on it

Run this command:

With Dart:

```
$ dart pub add intl
```

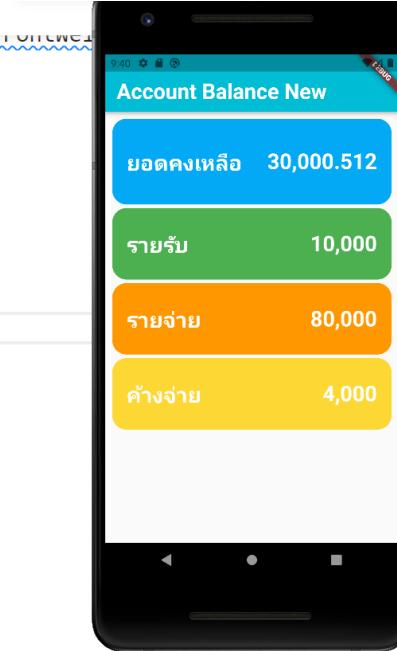
เมื่อติดตั้งแล้วให้ตรวจสอบที่ pubspec.yaml

A screenshot of the Android Studio IDE. The left sidebar shows the project structure with files like main.dart, pubspec.yaml, MoneyBox.dart, etc. The pubspec.yaml file is currently selected and open in the main editor area. The code in the editor is:

```
23 # Dependencies specify other packages that you
24 # To automatically upgrade your package depend
25 # consider running `flutter pub upgrade --maj
26 # dependencies can be manually updated by chan
27 # the latest version available on pub.dev. To
28 # versions available, run `flutter pub outdate
29 dependencies:
30   flutter:
31     sdk: flutter
32
33   # The following adds the Cupertino Icons for
34   # Use with the CupertinoIcons class for iOS
35   cupertino_icons: ^1.0.2
36   intl: ^0.17.0
37
38 dev_dependencies:
39   flutter_test:
40     sdk: flutter
41
42   # The "flutter_lints" package below contains
43   # encourage good coding practices. The lint
44   # activated in the `analysis_options.yaml` f
45   # package. See that file for information abo
```

A red box highlights the line "intl: ^0.17.0" in the dependencies section.

การกำหนด numberFormat ให้กับตัวเลข จากตัวแปร amount ได้ตามความต้องการ



The screenshot shows the Flutter IDE with the MoneyBox.dart file open. A red box highlights the line of code where the number format is applied:

```
1 import 'package:flutter/material.dart';
2 import 'package:intl/intl.dart';
3
4 lib > MoneyBox.dart > MoneyBox > build
5   ...
6   ...
7   ...
8   ...
9   ...
10  ...
11  ...
12  ...
13  ...
14  ...
15  ...
16  ...
17  ...
18  ...
19  ...
20  ...
21  ...
22  ...
23  ...
24  ...
25  ...
26  ...
27  ...
28  ...
29  ...
30  ...
31  ...
32  ...
33  ...
34  ...
35  ...
36  ...
37  ...
38  ...
39  ...
40  ...
41  ...
42  ...
43  ...
44  ...
45  ...
46  ...
47  ...
48  ...
49  ...
50  ...
51  ...
52  ...
53  ...
54  ...
55  ...
56  ...
```

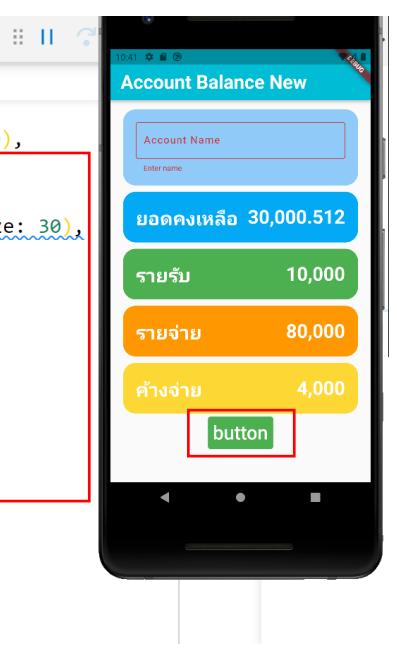
The highlighted code is:

```
'${NumberFormat("#,###.###").format(amount)}',
```

The resulting app screen shows account balance details:

Account Balance New	
ยอดคงเหลือ	30,000.512
รายรับ	10,000
รายจ่าย	80,000
คงเหลือ	4,000

The amount "30,000.512" is formatted with commas as thousands separators and two decimal places.



The screenshot shows the Flutter IDE with the main.dart file open. A red box highlights the addition of a TextButton to the Container:

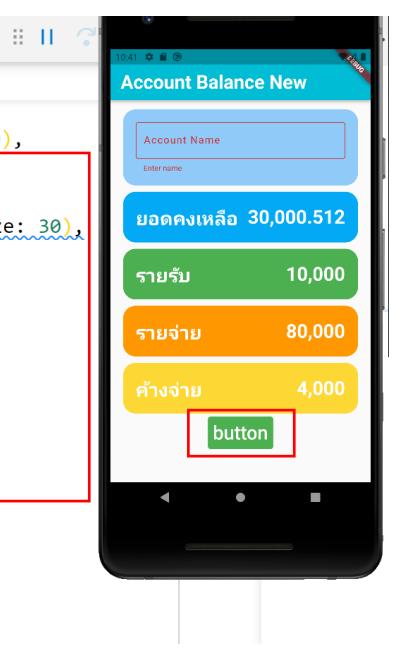
```
56 MoneyBox("รายจ่าย", 80000, 80, Colors.orange),
57 MoneyBox("คงเหลือ", 4000, 80, Colors.yellow.shade600),
58 Container(
59   child: TextButton(
60     child: Text("button", style: TextStyle(fontSize: 30)),
61   ), // Text
62   style: TextButton.styleFrom(
63     primary: Colors.white,
64     backgroundColor: Colors.green,
65   ),
66   onPressed: () {
67   },
68   ), // TextButton
69   ), // Container
70   ],
71   ), // Column
72   ); // Padding // Scaffold
73   }
74 }
```

The resulting app screen shows the button added to the Container:

Account Balance New	
Account Name	<input type="text"/>
ยอดคงเหลือ	30,000.512
รายรับ	10,000
รายจ่าย	80,000
คงเหลือ	4,000

A green button labeled "button" is visible at the bottom right of the screen.

เพิ่ม Widget TextButton ใน Container



The screenshot shows the Flutter IDE with the main.dart file open. A red box highlights the addition of a TextButton to the Container:

```
56 MoneyBox("รายจ่าย", 80000, 80, Colors.orange),
57 MoneyBox("คงเหลือ", 4000, 80, Colors.yellow.shade600),
58 Container(
59   child: TextButton(
60     child: Text("button", style: TextStyle(fontSize: 30)),
61   ), // Text
62   style: TextButton.styleFrom(
63     primary: Colors.white,
64     backgroundColor: Colors.green,
65   ),
66   onPressed: () {
67   },
68   ), // TextButton
69   ), // Container
70   ],
71   ), // Column
72   ); // Padding // Scaffold
73   }
74 }
```

The resulting app screen shows the button added to the Container:

Account Balance New	
Account Name	<input type="text"/>
ยอดคงเหลือ	30,000.512
รายรับ	10,000
รายจ่าย	80,000
คงเหลือ	4,000

A green button labeled "button" is visible at the bottom right of the screen.

เพิ่ม Widget TextFormField ใน Container โดยการสร้าง Class : InputDecoratorExample เพิ่ม ที่ main.dart

```
main.dart × MoneyBox.dart
lib > main.dart > _MyHomePageState > build
41   body: Padding(
42     padding: const EdgeInsets.all(10.0),
43     child: Column(
44       children: [
45         Container(
46           margin: EdgeInsets.symmetric(vertical: 5, horizontal: 10),
47           height: 120,
48           padding: const EdgeInsets.all(20.0),
49           decoration: BoxDecoration(
50             color: Colors.blue.shade200,
51             borderRadius: BorderRadius.circular(20)), // BoxDecoration
52             child: InputDecoratorExample(),
53           ), // Container
54           MoneyBox("ยอดคงเหลือ", 30000.512, 80, Colors.lightBlue),
55           MoneyBox("รายรับ", 10000, 80, Colors.green),
56           MoneyBox("รายจ่าย", 80000, 80, Colors.orange),
57           MoneyBox("ค้างจ่าย", 4000, 80, Colors.yellow.shade600),

```

lib > main.dart > InputDecoratorExample

```
76
77 class InputDecoratorExample extends StatelessWidget {
78   const InputDecoratorExample({Key? key}) : super(key: key);
79   @override
80   Widget build(BuildContext context) {
81     return TextFormField(
82       decoration: InputDecoration(
83         border: const OutlineInputBorder(),
84         labelText: 'Account Name',
85         labelStyle:
86           MaterialStateTextStyle.resolveWith((Set<MaterialState> states) {
87             final Color color = states.contains(MaterialState.error)
88               ? Theme.of(context).errorColor
89               : Colors.orange;
90             return TextStyle(color: color, letterSpacing: 1.3);
91           }),
92       ), // InputDecoration
93       validator: (String? value) {
94         if (value == null || value == '') {
95           return 'Enter name';
96         }
97         return null;
98       },
99       autovalidateMode: AutovalidateMode.always,
100     ); // TextFormField
101   }
102 }
103 }
```

** นอกเหนือไปนี้ ให้ศึกษาการนำ Widget อื่น ๆ มาใช้งานร่วมกันใน app ได้

