อัพเดทครั้งที่ 3

สูตรลัด Flutter

เขียน Flutter ง่ายๆ จากโค้ดตัวอย่าง

อนุชิต ชโลธร

สำนักพิมพ์ก๊อปวาง

สูตรลัด Flutter

เขียน Flutter ง่ายๆ จากโค้ดตัวอย่าง

พิมพ์ครั้งที่ 3

อนุชิต ชโลธร

สำนักพิมพ์ก๊อปวาง

สารบัญ

บทนำ

หนังสือเล่มนี้เน้นการเรียนรู้ Flutter และภาษา Dart จากตัวอย่างโค้ด ซึ่งเป็นโค้ดตัวอย่างที่ได้รวบรวมมาจากโครงการต่างๆ ที่ผู้เขียนได้พัฒนาแอปให้กับลูกค้าทั้งในประเทศและต่างประเทศ ผู้อ่านสามารถเรียนรู้แนวคิด การออกแบบหน้าจอ การจัดวางเลย์เอาท์ ตลอดจนเทคนิคการตกแต่งวิดเจ็ตต่างๆ ได้จากตัวอย่างโค้ด และนำไปประยุกต์ใช้งานได้

อนุชิต ชโลธร

วิธีการอ่าน

หนังสือเล่มนี้เน้นความเข้าใจ ผู้อ่านสามารถเรียนรู้แนวคิดการออกแบบหน้าจอ การจัดวางเลย์เอาท์ ตลอดจนเทคนิคการตกแต่งวิดเจ็ตต่างๆ จากตัวอย่างโค้ด ดังนั้น การอ่านหนังสือเล่มนี้ให้เกิดประโยชน์ ผู้อ่านควรลงมือทดลองทำทันที

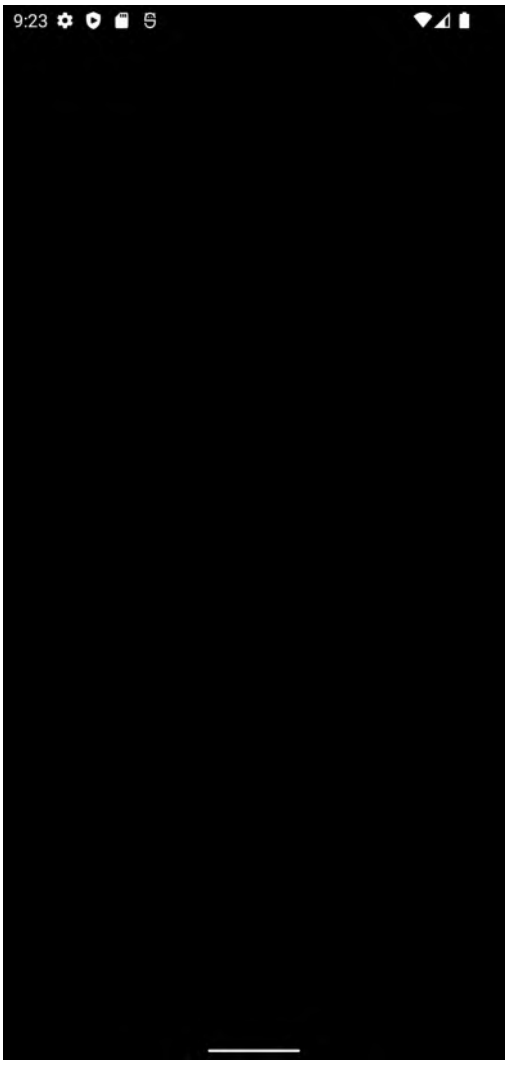
เนื้อหาในแต่ละบทจะแบ่งเนื้อหาออกเป็น 2 ส่วน

1. ส่วนเนื้อหา อธิบายหลักการ และแนวคิดต่างๆ
2. ส่วนซอร์สโค้ดตัวอย่าง

ผู้อ่านสามารถ “ก๊อปวาง” ซอร์สโค้ดตัวอย่าง เพื่อทดลองได้ด้วยตนเอง

1. หน้าไม่ดำต้องใช้ Scaffold

Scaffold เป็น Widget สำหรับขึ้นโครงหลักใน Material Design ซึ่ง Widget นี้ทำหน้าที่เป็นกำหนดโครงสร้างและธีม ดังนั้นแต่ละหน้าจอจะควรขึ้นต้นด้วย Scaffold หากไม่ได้ใช้ Scaffold เราจะพบว่าหน้าจอจะไม่มีการเรียกใช้ธีม และหน้าจอจะแสดงผลเป็นสีดำ ดังรูป ดังนั้นควรใช้ Scaffold เป็น Widget หลักของทุกๆ หน้า



หน้าจอแบบไม่ใช้ Scaffold

ตัวอย่างโค้ด

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold();
  }
}
```

ตัวอย่างหน้าจอที่ใช้ Scaffold เพื่อเรียกใช้ธีมจะแสดงผลได้ถูกต้อง

หน้าจอแบบใช้ Scaffold

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return const Scaffold();
  }
}
```

2. บอกหน่อยว่าหน้าจออะไร ผ่าน AppBar

โครงสร้างหน้าจอตาม Material Design จะแบ่งออกเป็น 2 ส่วนหลักๆ คือ AppBar และ Body การเรียกใช้ AppBar จะอยู่ภายใต้ Widget Scaffold เราจะใช้ Widget AppBar ในการแสดงชื่อของหน้าจอ และแสดงปุ่ม Action Button ต่างๆ ได้

หน้าจอแสดง AppBar

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      body: Container(),
    );
  }
}
```

ใน Scaffold นี้ นอกจาก AppBar แล้วยังมีส่วนที่สามารถตั้งค่าเพิ่มเติมได้อีกหลายส่วน เช่น AppBar, Drawer, Navigation Bottom และ Floating Action Button เป็นต้น

ตัวอย่างหน้าจอ เพิ่ม AppBar, Bottom Navigation และ Drawer

เพิ่ม AppBar, Bottom Navigation และ Drawer

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: Container(),
      drawer: Drawer(
        // put drawer items here
        child: Container(),
      ),
      // bottom navigation bar
      bottomNavigationBar: BottomNavigationBar(
        items: const [
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Home',
          ),
```

```
          BottomNavigationBarItem(
            icon: Icon(Icons.notifications),
            label: 'Notifications',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.account_circle),
            label: 'Profile',
          ),
        ],
      ),
    );
  }
}
```

3. ใส่ Action ให้ AppBar

ในส่วน AppBar สามารถแสดงชื่อหน้าจอ (Title) และใส่ปุ่ม action ต่างๆ ได้ ผ่านการกำหนดค่า action ซึ่งใน action นี้จะเป็น List ของ Widget เราสามารถใส่ปุ่มที่เป็น Eleveted Button, Text Button หรือแม้กระทั่ง Icon Button ได้

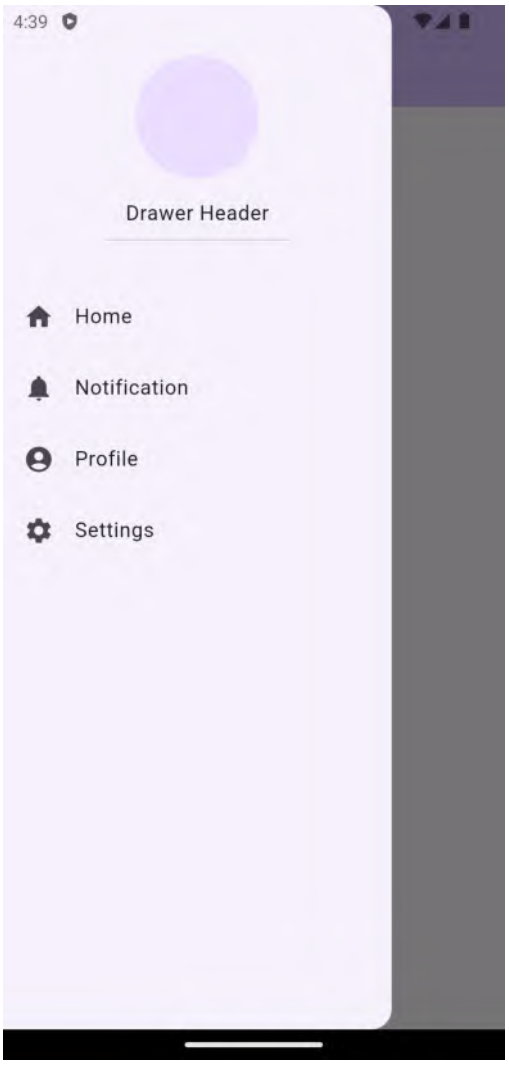
ใส่ Action ใน AppBar

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
        actions: [
          IconButton(
            onPressed: () {},
            icon: const Icon(
              Icons.settings,
            ),
          ),
        ],
      ),
      // body
      body: Container(),
    );
  }
}
```

4. เพิ่มเมนู Drawer กันหน่อย

จากที่ทราบกันในเบื้องต้นแล้วว่า Scaffold มีส่วนตั้งค่า Drawer หรือแถบเมนูเพิ่มได้ด้วย ซึ่งในส่วนของ Drawer เราสามารถสร้างเมนูผ่าน Drawer Widget ได้ ทั้งนี้ขึ้นอยู่กับการออกแบบหน้าจอของแอปนั้นๆ ซึ่ง Widget Scaffold จะเพิ่มเมนูเพื่อเรียก Drawer ให้อัตโนมัติ โดยที่เราไม่ต้องทำปุ่มเพื่อเปิด/ปิด Drawer เพิ่มเติมแต่อย่างใด



แสดง Navigation Drawer

ตัวอย่างโค้ด

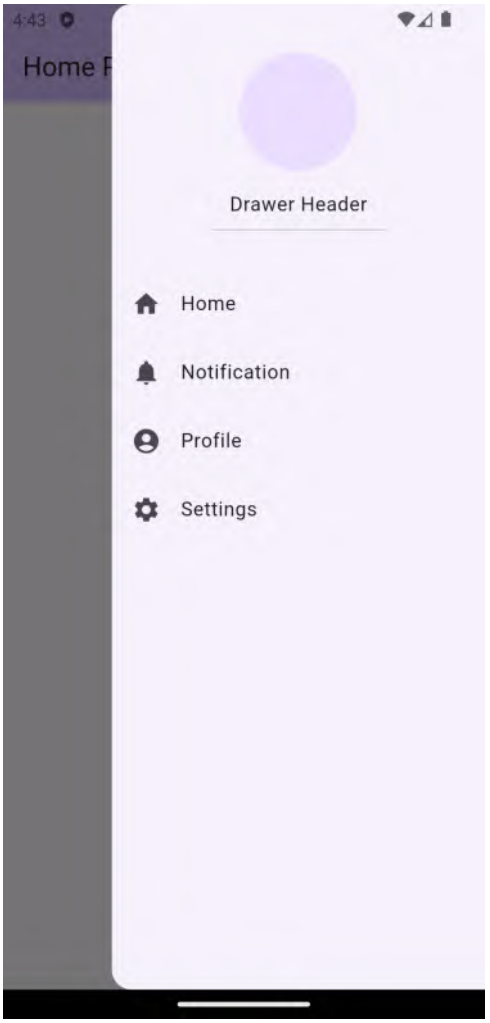
```
import 'package:flutter/material.dart';

class HomePage extends StatelessWidget {
  const HomePage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: Container(),
      // drawer
      drawer: Drawer(
        child: Column(
          children: [
            // drawer header
            const DrawerHeader(
              child: Column(
                mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                children: [
                  CircleAvatar(
                    radius: 48,
```

```
            ),
            Text("Drawer Header"),
          ],
        ),
      ),
      // list menu
      Expanded(
        child: ListView(
          children: const [
            ListTile(
              leading: Icon(Icons.home),
              title: Text("Home"),
            ),
            ListTile(
              leading: Icon(Icons.notifications),
              title: Text("Notification"),
            ),
            ListTile(
              leading: Icon(Icons.account_circle),
              title: Text("Profile"),
            ),
            ListTile(
              leading: Icon(Icons.settings),
              title: Text("Settings"),
            ),
          ],
        ),
      )
```

```
          ],
        ),
      ),
    );
  }
}
```

5. กำหนด Drawer ให้อยู่ทางขวา

Navigation Drawer ยังสามารถกำหนดการแสดงผลทางซ้ายหรือทางขวาของหน้าจอได้ เราไม่จำเป็นต้องสร้าง Widget ขึ้นมาใหม่ เพียงแค่กำหนดค่าใน drawer เพื่อแสดง Navigation Drawer เท่านั้น หากต้องการให้ Drawer แสดงผลทางซ้าย ให้ใส่ค่าลงใน drawer หากต้องการแสดง Navigation Drawer ทางขาว ให้ใส่ค่าลงใน endDrawer ซึ่งเป็นการระบุตำแหน่งของ Navigation Drawer ได้อย่างง่ายๆ



แสดง Drawer ด้านขวา

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: Container(),
      // drawer
      endDrawer: Drawer(
        child: Column(
          children: [
            // drawer header
            const DrawerHeader(
              child: Column(
                mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                children: [
```

```
            CircleAvatar(
              radius: 48,
            ),
            Text("Drawer Header"),
          ],
        ),
      ),
      // list menu
      Expanded(
        child: ListView(
          children: const [
            ListTile(
              leading: Icon(Icons.home),
              title: Text("Home"),
            ),
            ListTile(
              leading: Icon(Icons.notifications),
              title: Text("Notification"),
            ),
            ListTile(
              leading: Icon(Icons.account_circle),
              title: Text("Profile"),
            ),
            ListTile(
              leading: Icon(Icons.settings),
              title: Text("Settings"),
            ),
          ],
```

```
              ),
            )
          ],
        ),
      ),
    );
  }
}
```

6. ใส่เมนูข้างล่างหน้าจอ (Bottom Navigation Bar)

ใน Material Design เมนูข้างล่างหน้าจอหรือ Bottom Navigation Bar เราสามารถใช้ Widget ที่ชื่อ Bottom Navigation Bar หรือ Navigation Bar ได้ ซึ่งรูปแบบการแสดงผล และการใช้งานจะแตกต่างกัน เล็กน้อย หากเน้นการแสดงผลรูปแบบของ Material Design 3 แนะนำให้ใช้ Navigation Bar ทั้งนี้ขึ้นอยู่กับงานออกแบบว่าจะเลือกใช้ Bottom Navigation Bar แบบใด

Bottom Navigation Bar

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: Container(),
      // bottom navigation bar
      bottomNavigationBar: BottomNavigationBar(
        items: const [
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Home',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.notifications),
            label: 'Notifications',
          ),
```

```
          BottomNavigationBarItem(
            icon: Icon(Icons.account_circle),
            label: 'Profile',
          ),
        ],
      ),
    );
  }
}
```

ตัวอย่างหน้าจอที่ใช้ Navigation Bar

Navigation Bar

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: Container(),
      // navigation bar
      bottomNavigationBar: NavigationBar(
        destinations: const [
          NavigationDestination(
            icon: Icon(Icons.home),
            label: 'Home',
          ),
          NavigationDestination(
            icon: Icon(Icons.notifications),
            label: 'Notifications',
          ),
```

```
          NavigationDestination(
            icon: Icon(Icons.account_circle),
            label: 'Profile',
          ),
        ],
      ),
    );
  }
}
```

จากตัวอย่างข้างต้น เป็นการใช้งานหน้าจอหลักที่ขึ้นต้นด้วย Scaffold เรา สามารถปรับเปลี่ยนการแสดงผลได้หลากหลายขึ้นอยู่กับงานออกแบบหน้าจอ

7. AppBar เดิมไม่ถูกใจสร้างใหม่ซะเลย

เราสามารถออกแบบ AppBar ได้เอง โดยไม่ใช้ Widget AppBar ได้เช่นกัน โดยอ้างอิงจากงานออกแบบของ designer ได้ ซึ่งอย่าลืมว่าใน Flutter ทุกอย่างที่แสดงผลบนหน้าจอคือ Widget ดังนั้นเราสามารถสร้าง AppBar ตามที่เราต้องการได้ ทั้งนี้ขึ้นอยู่กับงานออกแบบเป็นหลัก

หน้าจอ AppBar ออกแบบเอง

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        children: [
          // custom design appbar
          Padding(
            padding: const EdgeInsets.only(
              top: 32.0,
              bottom: 16.0,
            ),
            child: Row(
              children: [
                // profile
                const Padding(
                  padding: EdgeInsets.all(8.0),
                  child: CircleAvatar(),
                ),
                // title
                Text(
                  "John Doe",
                  style:
```

```
Theme.of(context).textTheme.titleLarge,
                ),
                // spacer
                const Spacer(),
                // botton
                IconButton(
                  onPressed: () {},
                  icon: const Icon(Icons.arrow_back),
                )
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

เราจะเห็นได้ว่า เราสามารถสร้าง Widget ใหม่ๆ ได้ตามต้องการ และมีรูปแบบตามงานออกแบบของ Designer ได้ ทั้งนี้ขึ้นอยู่กับความละเอียดของงานออกแบบ

8. ใส่ปุ่ม FAB

Floating Action Button หรือ FAB เป็นปุ่มที่ลอยอยู่บนหน้าจอ เราสามารถปรับแต่งการตั้งค่าได้หลายอย่าง เช่น การใส่ Icon การใส่ข้อความ การจัดตำแหน่ง เป็นต้น การใช้งาน Fab ควรระมัดระวังเรื่องการแสดงผลของปุ่ม ซึ่งอาจไปทับเนื้อหาที่อยู่ในหน้าจอได้

Floating Action Button

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: Container(),
      // floating action button
      floatingActionButton: FloatingActionButton(
        onPressed: () {},
        child: const Icon(
          Icons.add,
        ),
      ),
    );
  }
}
```

ตัวอย่าง Floating Action Button แบบกำหนดตำแหน่ง กึ่งกลางหน้าจอ

Floating Action Button แบบกำหนดตำแหน่ง

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: Container(),
      // floating action button
      floatingActionButtonLocation:
FloatingActionButtonLocation.centerFloat,
      floatingActionButton: FloatingActionButton(
        onPressed: () {},
        child: const Icon(
          Icons.add,
        ),
      ),
    );
  }
}
```

นอกจากการเรียกใช้ Floating Action button แบบปกติแล้ว ในเว็บไซต์ pub.dev ยังมี plugin อื่นๆ ที่น่าสนใจที่เกี่ยวข้องกับ Floating Action Button อยู่หลายตัว เช่น floating action bubble, animated floating buttons, form floating action button เป็นต้น สามารถนำมาประยุกต์ใช้งานได้

9. จัดกึ่งกลางด้วย Center

Widget ที่เกี่ยวกับการจัดวางมีหลายตัวขึ้นอยู่กับว่าเราเรียกใช้ Widget อะไรอยู่ ตัวอย่างเช่น การจัดกึ่งกลางหน้าจอเราจะใช้ Widget ที่ชื่อว่า Center แต่หากต้องการจัดในรูปแบบอื่น เช่น ชิดซ้าย ชิดขวา เราสามารถใช้ Widget ที่ชื่อว่า Align เข้ามาช่วยใช้ในการจัดรูปแบบได้เช่นกัน และยังมี Widget อื่นๆ อีกหลายตัวที่ช่วยจัดรูปแบบได้ ขึ้นอยู่กับการประยุกต์ใช้งาน

Widget Center

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      body: const Center(
        child: Text("Center Text"),
      ),
    );
  }
}
```

10. จัดตำแหน่งด้วย Align

Align เป็น Widget สำหรับกำหนดตำแหน่งของ Widget ที่อยู่ภายใต้ Widget นี้ในหลายรูปแบบ เช่น จัดกึ่งกลาง, ซ้ายบน, ขวาบน, ซ้ายล่าง, ขวาล่าง, ซ้ายกลาง, ขวากลาง และ กึ่งกลาง เป็นต้น

ใช้ Align

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: const Align(
        alignment: Alignment.center,
        child: Text("Center Text"),
      ),
    );
  }
}
```

11. ใช้ Align จัดตำแหน่งใน Stack

Stack เป็น Widget ที่ใช้แสดง Widget ซ้อนทับกัน เป็นชั้นๆ (layer) การใช้ Align ร่วมกับ Stack จะทำให้เราสามารถจัด Widget ในแต่ละ layer ให้อยู่ในตำแหน่งต่างๆ ได้อย่างอิสระ

การใช้ Align ร่วมกับ Stack

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: Stack(
        children: [
          // top left
          Align(
            alignment: Alignment.topLeft,
            child: Container(
              width: 100,
              height: 100,
              color: Colors.red,
            ),
          ),
          // center
```

```
          Align(
            alignment: Alignment.center,
            child: Container(
              width: 100,
              height: 100,
              color: Colors.green,
            ),
          ),
          // bottom right
          Align(
            alignment: Alignment.bottomRight,
            child: Container(
              width: 100,
              height: 100,
              color: Colors.blue,
            ),
          ),
        ],
      ),
    );
  }
}
```

12. จัดเรียงแนวตั้งด้วย Column

การจัดรูปแบบแนวตั้งโดยใช้ Column Widget ที่อยู่ใน Column นั้นจะมีการเรียงตัวแบบแนวตั้ง ดังนั้นการจัดรูปแบบข้อมูลที่อยู่ใน Column สามารถทำได้ 2 แบบ ได้แก่ การจัดวางเรียงตัวแนวตั้งและการจัดวางเรียงตัวแนวนอน

การจัดวางแนวตั้งของ Column เราสามารถกำหนดขนาดของ Column ได้ซึ่งโดยปกติจะอิงขนาดของ Widget ที่อยู่ก่อนหน้าเป็นหลัก หาก Widget ก่อนหน้ามีขนาดเท่ากับหน้าจอหลัก Column ที่วางจะมีขนาดความสูงเท่ากับขนาดของหน้าจอหลักเช่นกัน

สีเทาใน Container แสดงให้เห็นว่า Column มีความสูงเท่ากับหน้าจอ

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: Container(
        color: Colors.grey,
        child: const Column(
          children: [
            Text("Text 1"),
            Text("Text 2"),
          ],
        ),
      ),
    );
  }
}
```

13. กำหนดขนาด Column

การกำหนดขนาดของ Column ให้สูงเท่ากับ Widget ที่อยู่ภายใน Column สามารถกำหนดขนาด ด้วย MainAxisSize

สีใน Container แสดงให้เห็นว่า Column มีความสูงเท่ากับขนาดของ Widget ที่อยู่ข้างใน

ตัวอย่างโค้ด

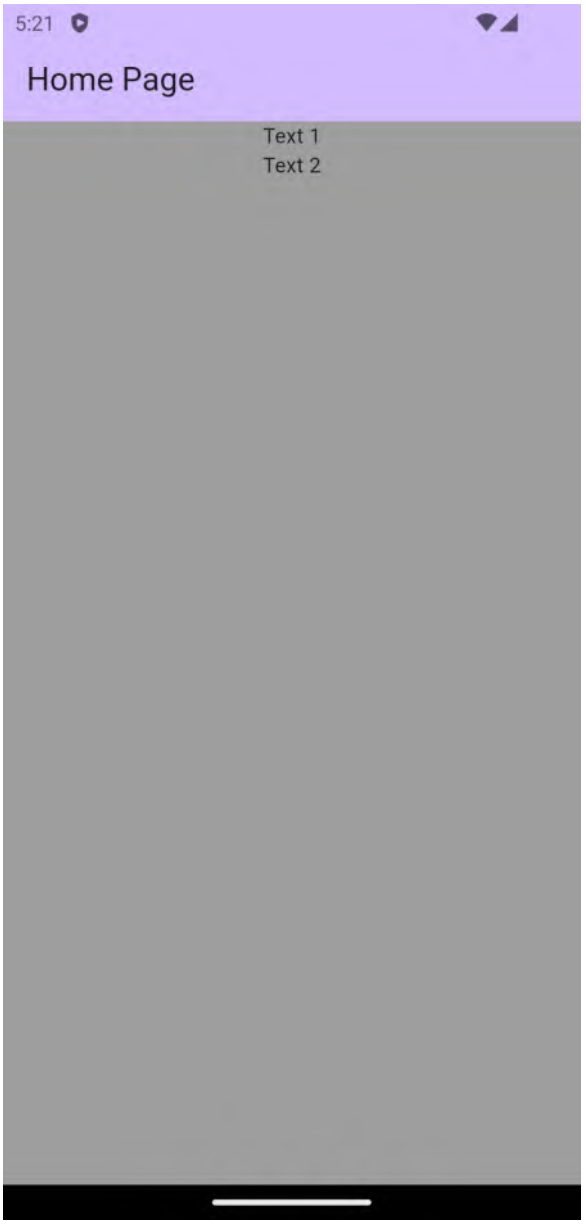
```
import 'package:flutter/material.dart';

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: Container(
        color: Colors.grey,
        child: const Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            Text("Text 1"),
            Text("Text 2"),
          ],
        ),
      ),
    );
  }
```

```
}
```

การเรียงตัวของ Widget ที่อยู่ภายใน Column จะถูกบังคับให้เรียงตัวตามแนวตั้งโดยอัตโนมัติ เราสามารถจัดวาง Widget ในรูปแบบแนวตั้งได้หลายแบบ เช่น กำหนดให้อยู่ข้างบน กำหนดกึ่งกลาง กำหนดให้อยู่ด้านล่าง หรือกระจาย ทั้งนี้ขึ้นอยู่กับงานออกแบบของนักออกแบบ

14. จัดอยู่ข้างบน Column

การจัดเรียง Widget ใน Column ยังสามารถทำได้หลายแบบ เช่น การจัดวางแนวตั้ง จัด Widget ให้อยู่ด้านบนของ Column



จัด Widget ให้อยู่ด้านบนของ Column

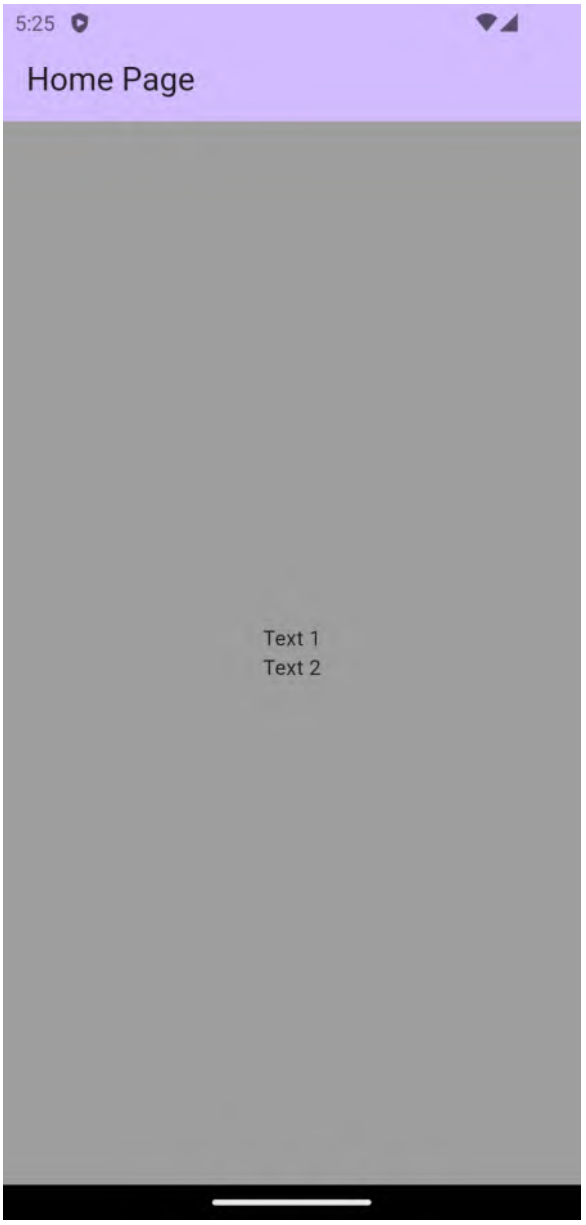
ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        // appbar
        appBar: AppBar(
          title: const Text("Home Page"),
          backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
        ),
        // body
        body: Container(
          width: MediaQuery.of(context).size.width,
          color: Colors.grey,
          child: const Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
              Text("Text 1"),
              Text("Text 2"),
            ],
          ),
```

```
        ));
  }
}
```

15. จัดอยู่ตรงกลาง Column

การจัดวาง Widget ให้อยู่ตรงกลางของ Column



จัด Widget ให้อยู่ตรงกลางของ Column

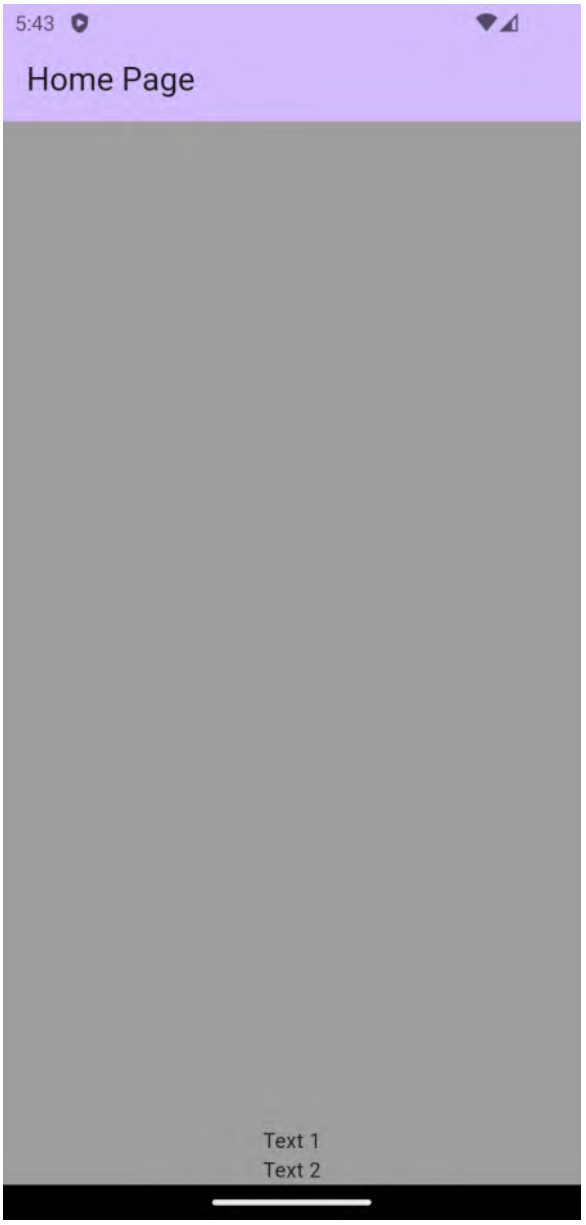
ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: Container(
        width: MediaQuery.of(context).size.width,
        color: Colors.grey,
        child: const Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text("Text 1"),
            Text("Text 2"),
          ],
        ),
```

```
      ),
    );
  }
}
```

16. จัดอยู่ตรงข้างล่าง Column

การจัดวาง Widget ให้อยู่ตรงด้านล่างของ Column



จัดวาง Widget ให้อยู่ตรงด้านล่างของ Column

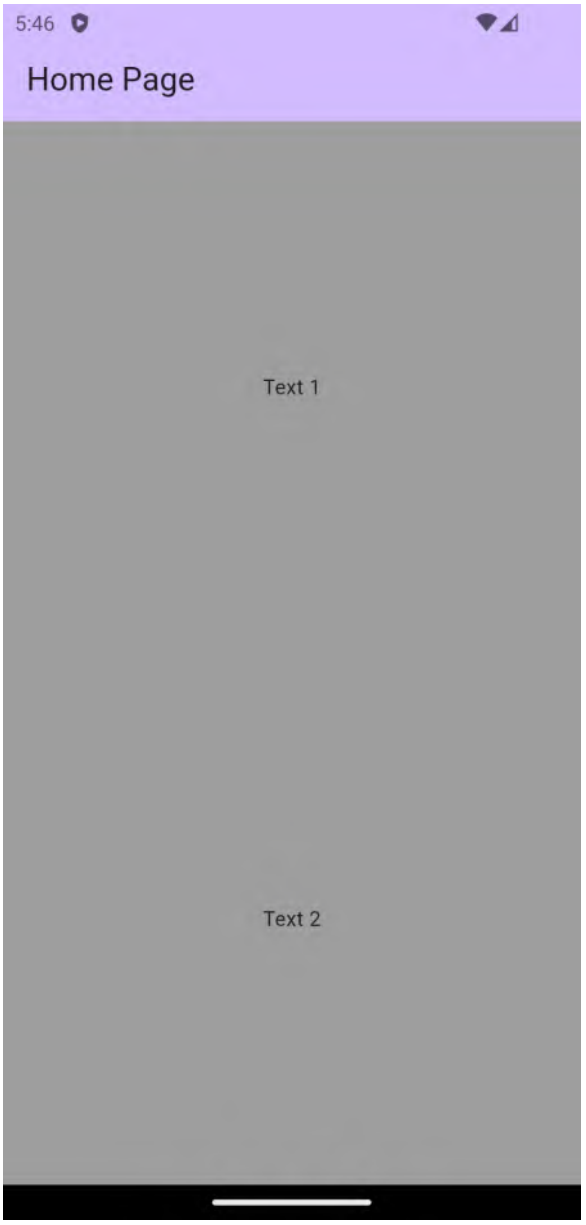
ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        // appbar
        appBar: AppBar(
          title: const Text("Home Page"),
          backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
        ),
        // body
        body: Container(
          width: MediaQuery.of(context).size.width,
          color: Colors.grey,
          child: const Column(
            mainAxisAlignment: MainAxisAlignment.end,
            children: [
              Text("Text 1"),
              Text("Text 2"),
            ],
          ),
```

```
        ));
  }
}
```

17. จัดวางแบบกระจายใน Column

การจัดวางแบบกระจายโดยมีช่องว่างระหว่าง Widget ขนาดเท่าๆกัน



จัดวางแบบกระจายโดยช่องว่างระหว่าง Widget มีขนาดเท่าๆกัน

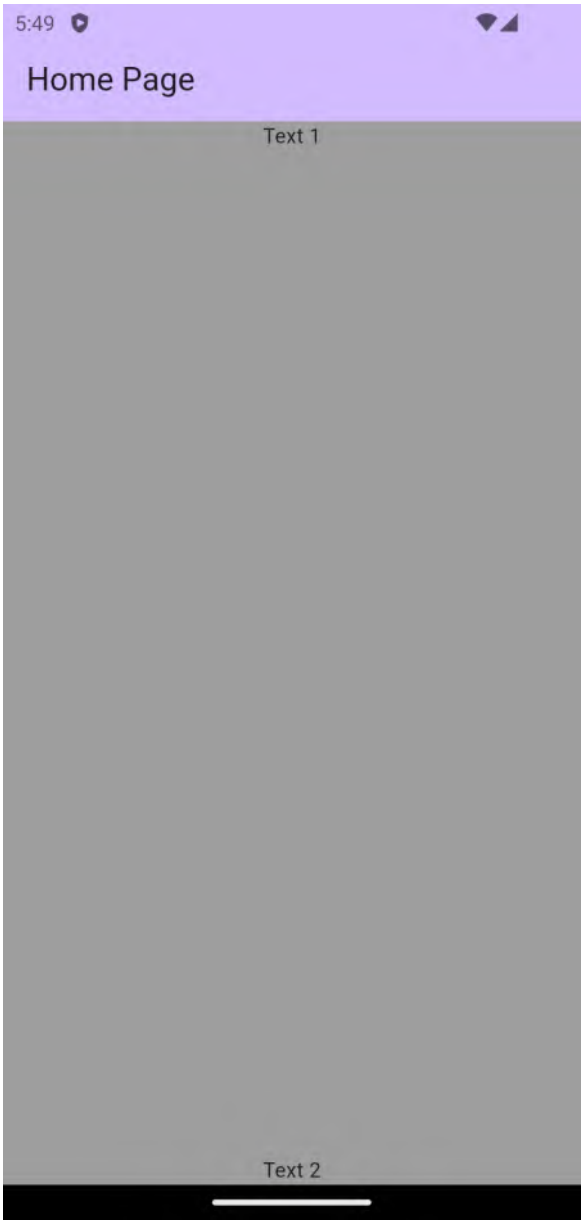
ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: Container(
        width: MediaQuery.of(context).size.width,
        color: Colors.grey,
        child: const Column(
          mainAxisAlignment: MainAxisAlignment.spaceAround,
          children: [
            Text("Text 1"),
            Text("Text 2"),
          ],
        ),
```

```
      ),
    );
  }
}
```

18. จัดวางแบบแยกออกจากกัน Column

การจัดวางแบบกระจายโดยแสดง Widget แยกกัน แสดงอยู่ข้างบนและข้างล่าง



การจัดวางแบบกระจาย แยก Widget แสดงอยู่ข้างบนและข้างล่าง

ตัวอย่างโค้ด

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appbar
      appBar: AppBar(
        title: const Text("Home Page"),
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      ),
      // body
      body: Container(
        width: MediaQuery.of(context).size.width,
        color: Colors.grey,
        child: const Column(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: [
            Text("Text 1"),
            Text("Text 2"),
          ],
        ),
      ),
    );
```