Week 8 : OOP and AppBar Widget

ผศ. ดร. เก็จแก้ว ธเนศวร

kejkaew.tha@mail.kmutt.ac.th

ครั้งที่	วันที่	หัวข้อ
6	21/2/2566	Flutter – Basics of Dart Programming
7	28/2/2566	Article: Methodology + submit background
8	7/3/2566	Flutter – OOP, asynchronous programming, and Introduction to Widgets
9	14/3/2566	Flutter - Introduction to Layouts
10	21/3/2566	Article: Experiment and results + submit methodology
12	28/3/2566	Flutter - Animations and Graphics
11	4/4/2566	Article: Conclusions and future work + submit experiment and results
13	11/4/2566	ไม่มีเรียน (GDM443/DMT443)
14	18/4/2566	Flutter - Data Storage and Management
15	25/4/2566	Article: Abstract
16	2/5/2566	Flutter - Data Storage and Management (2)
17	9/5/2566	ส่ง research paper
18	16/5/2566	Flutter - Deploying Flutter Applications
19	23/5/2566	Flutter - ส่งโปรเจค

Last week

Variables and Operation

Null Safety

Conditions and Loops

Functions

Today outline

OOP

AppBar widget

ทบทวนก่อนเรียน

เขียนฟังก์ชันใน Dart สำหรับ reverse String โดยใช้ function

OOP In Dart

Object-oriented programming (OOP) is a programming method that uses objects and their interactions to design and program applications.

It is one of the most popular programming paradigms and is used in many programming languages, such as Dart, Java, C++, Python, etc.

Advantages

It is easy to understand and use.

It increases reusability and decreases complexity.

The productivity of programmers increases.

It makes the code easier to maintain, modify and debug.

It promotes teamwork and collaboration.

It reduces the repetition of code.

Features Of OOP

Class

Object

Encapsulation

Inheritance

Polymorphism

Declaring Class In Dart

The class keyword is used for defining the class.

ClassName is the name of the class and must start with capital letter.

Body of the class consists of properties and functions.

Properties are used to store the data. It is also known as fields or attributes.

Functions are used to perform the operations. It is also known as methods.

Example

Syntax

```
class ClassName {
// properties or fields
// methods or functions
}
```

```dart
class Person {
  String? name;
  String? lastname;
  String? phone;
  int? age;

  // Default Constructor
  Person() {
    // กำหนดค่าในนี้ได้
    print("This is a default constructor");
  }
```

Declaring Object In Dart

Once you have created a class, it's time to declare the object. You can declare an object by the following syntax:

Syntax

```
ClassName objectName = ClassName();
```

Example: Class and Object

```
102    Person a = Person();
103    a.name = "Jane";
104    a.lastname = "Doe";
105    a.age = 25;
106    a.phone = "0214563";
107    a.displayInfo();
```

Output

```
This is a default constructor
Person name: Jane Doe.
Phone number: 0214563.
Age: 25.
```

Constructor

If you don’t define a constructor for class, then you need to set the values of the properties manually.

The constructor’s name should be the same as the class name.

Constructor doesn’t have any return type.

Syntax

```
class ClassName {
  // Constructor declaration: Same as class name
  ClassName() {
    // body of the constructor
  }
}
```

Constructor

You can also write the constructor in short form. You can directly assign the values to the properties.

```
22  // Constructor in short form, ใช้แบบนี้ดีกว่า
23  Person.anotherConstructor(this.name, this.lastname, this.phone, this.age);
```

Named constructor

In most programming languages like java, c++, c#, etc., we can create multiple constructors with the same name.

But in Dart, this is not possible. Well, there is a way. We can create multiple constructors with the same name using named constructors.

```
22  // Constructor in short form, ใช้แบบนี้ดีกว่า
23  Person.anotherConstructor(this.name, this.lastname, this.phone, this.age);
```

Example

```
108    Person b = Person.anotherConstructor("John", "Doe", "0321456", 30);
109    b.displayInfo();
```

Output

```
Person name: John Doe.
Phone number: 0321456.
Age: 30.
```

Activity 1

สร้าง class Car ที่ประกอบด้วย properties 3 ตัว คือ name, color, price

มี constructor 2 แบบ คือแบบที่ไม่รับ parameters เลย กับ แบบที่รับ parameters 3 ตัว

มี method 1 อัน คือ display สำหรับแสดงข้อมูลทั้ง 3 อย่าง

Encapsulation

Encapsulation means hiding data within a library, preventing it from outside factors.

It helps you control your program and prevent it from becoming too complicated.

Encapsulation can be achieved by:

Declaring the class properties as private by using underscore(_).

Providing public getter and setter methods to access and update the value of private property.

Getter

Getter is used to get the value of a property. It is mostly used to access a private property's value.

Syntax

```
return_type get property_name {
  // Getter body
}
```

Setter

Setter is used to set the value of a property. It is mostly used to update a private property's value.

Syntax

```
set property_name (value) {
  // Setter body
}
```

Getter and Setter

Use Of Getter and Setter

- Validate the data before reading or writing.
- Restrict the read and write access to the properties.
- Making the properties read-only or write-only.
- Perform some action before reading or writing the properties.

Example

```
111  // Getter and setter
112  Person2 c = Person2("Jane", "Doe", "02145", 27);
113  print(c.fullName);
114  print("I am ${c.age} years old");
115  c.phone = "0";
116  print(c.phone);
```

Output

```
Hello, Jane Doe.
I am 27 years old
Exception: Phone number is not correct.
My phone number is 02145.
```

Inheritance

Inheritance is a sharing of behaviour between two classes. It allows you to define a class that extends the functionality of another class. The extend keyword is used for inheriting from parent class.

Syntax

```
class ParentClass {
  // Parent class code
}

class ChildClass extends ParentClass {
  // Child class code
}
```

Example:

```
119  var student = Student();
120  student.name = "John";
121  student.lastname = "Doe";
122  student.age = 20;
123  student.schoolName = "ABC School";
124  student.schoolAddress = "New York";
125  student.displayInfo();
126  student.displaySchoolInfo();
```

Output

```
This is a default constructor
Person name: John Doe.
Phone number: null.
Age: 20.
John's School Name: ABC School
School Address: New York
```

Super

Super is used to refer to the parent class. It is used to call the parent class's properties and methods.

```
128  // Super
129  var student2 = Student.anotherConstructor("John", "Doe", "0321456", 20);
130  student2.schoolName = "XYZ School";
131  student2.schoolAddress = "New York";
132  student2.displayInfo();
133  student2.displaySchoolInfo();
```

Output

```
Person name: John Doe.
Phone number: 0321456.
Age: 20.
John's School Name: XYZ School
School Address: New York
```

Polymorphism

Method overriding is a technique in which you can create a method in the child class that has the same name as the method in the parent class.

The method in the child class overrides the method in the parent class.

Syntax

```
class ParentClass{
void functionName(){
  }
}
class ChildClass extends ParentClass{
@override
void functionName(){
  }
}
```

Example

```
135  // Override
136  var teacher = Teacher.anotherConstructor("Jim", "Doe", "0321456", 35);
137  teacher.schoolName = "XYZ School";
138  teacher.expertise = "Math";
139  teacher.displayInfo();
```

Output

```
Jim's expertise: Math
School name: XYZ School
```

Activity 2

สร้าง class Animal ที่มี properties id, name, color และ method displayAnimal แสดงผล id, color และ name ของสัตว์ตัวนั้น

สร้าง class Cat ที่ extends มาจาก Animal และมี property ใหม่คือ sound ที่เป็น String ไว้ใส่ค่าคำว่า Meow

สร้าง method สำหรับแสดงผลของ cat ด้วย ว่าชื่ออะไร id อะไร ร้องเสียงยังไง และมีสีอะไร ซึ่ง method นี้มาจากการ override method displayAnimal ใน class Animal

Widgets

MaterialApp class

Flutter has widgets specific to a particular platform - Android or iOS.

Android specific widgets are designed in accordance with Material design guideline by Android OS.

- Android specific widgets are called as Material widgets.

iOS specific widgets are designed in accordance with Human Interface Guidelines by Apple and they are called as Cupertino widgets.

The most used material widgets

- Scaffold
- AppBar
- BottomNavigationBar
- TabBar
- TabBarView
- ListTile
- RaisedButton
- FloatingActionButton
- FlatButton
- IconButton
- DropdownButton
- PopupMenuButton
- ButtonBar
- TextField
- Checkbox
- Radio
- Switch
- Slider
- Date & Time Pickers
- SimpleDialog
- AlertDialog

New Flutter project in VS Code

Open the Command Palette (Ctrl+Shift+P on Windows หรือ Cmd+Shift+P on macOS).

Select the ”Flutter: New Project” command and press Enter.

Select ”Application” and press Enter.

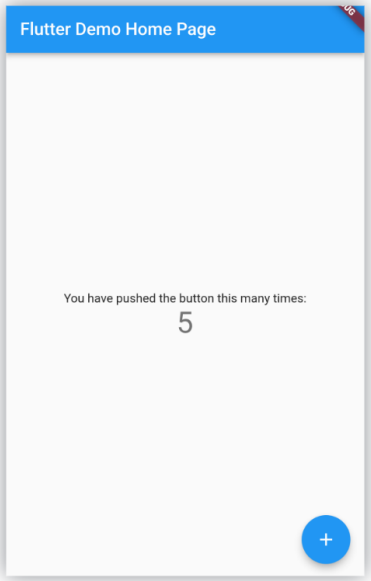
Select a Project location.

Enter your desired Project name.

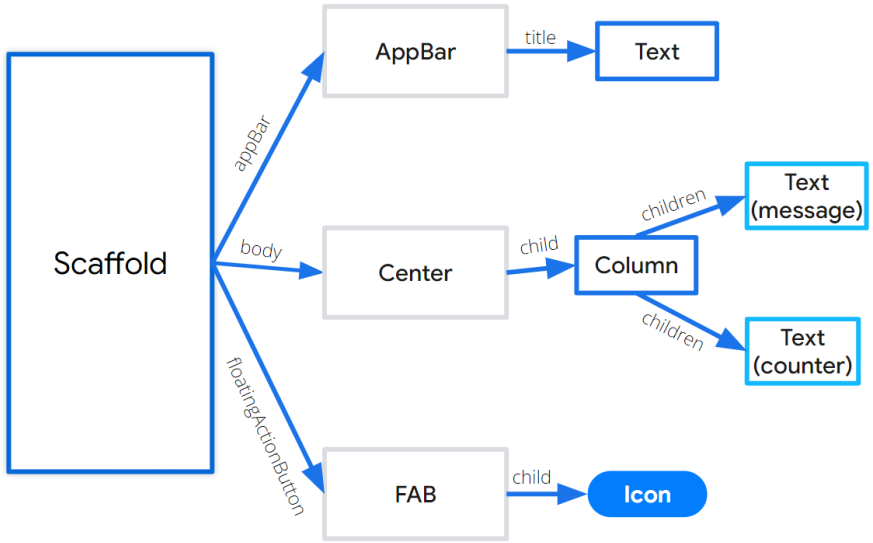
สร้างแล้วจะ setting file ที่จะเป็น main() ใน launch.json file ของ vs code

Example (main.dart)

UI



Breakdown



State maintenance widgets

In Flutter, all widgets are either derived from StatelessWidget or StatefulWidget.

A stateless widget never changes. Icon, IconButton, and Text are examples of stateless widgets.

Stateless widgets subclass StatelessWidget.

State maintenance widgets

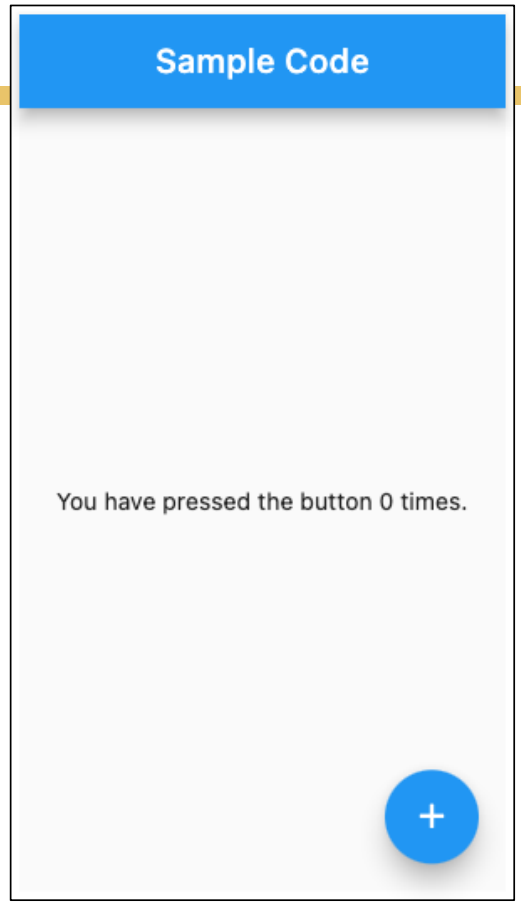
A stateful widget is dynamic.

For example, it can change its appearance in response to events triggered by user interactions or when it receives data. Checkbox, Radio, Slider, InkWell, Form, and TextField are examples of stateful widgets.

Stateful widgets subclass StatefulWidget.

Scaffold

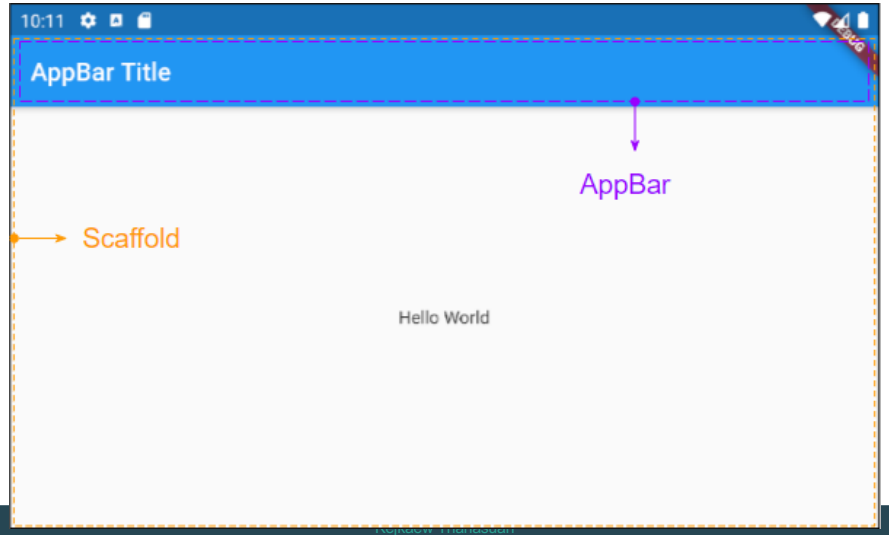
Scaffold is a class in flutter which provides many widgets or we can say APIs like Drawer, Snack-Bar, Bottom-Navigation-Bar, Floating-Action-Button, App-Bar, etc.



AppBar

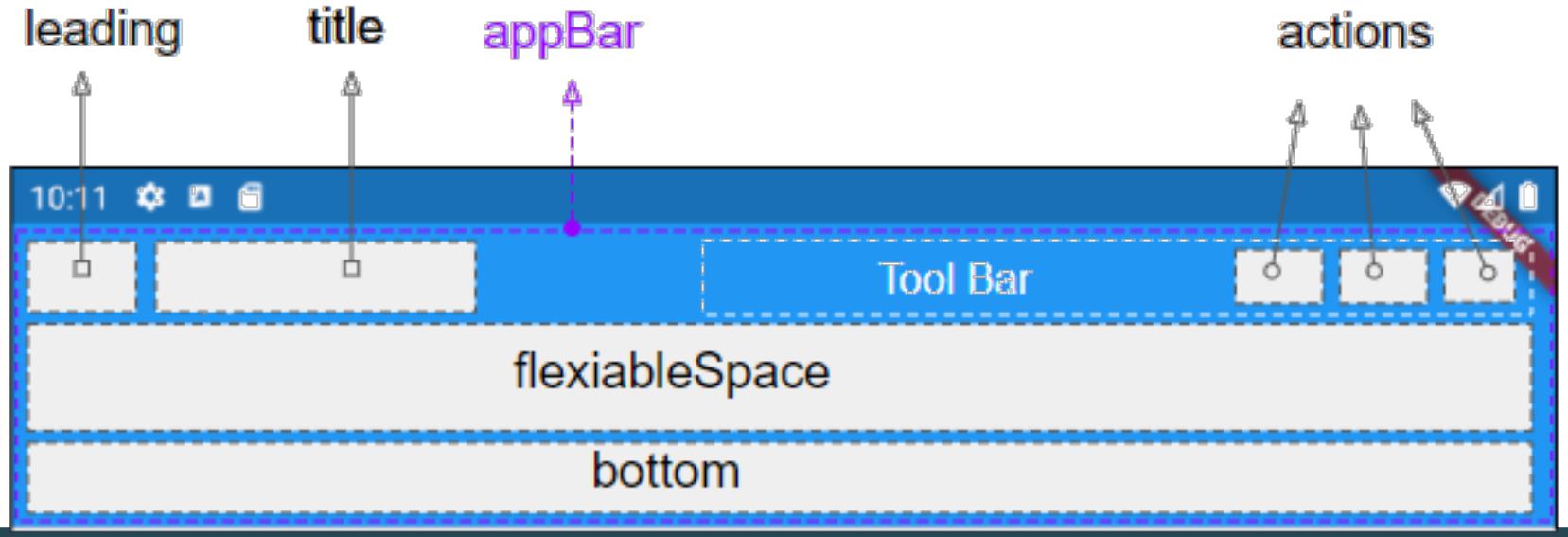
AppBar is usually the topmost component of the app (or sometimes the bottom-most).

It contains the toolbar and some other common action buttons.



AppBar: areas

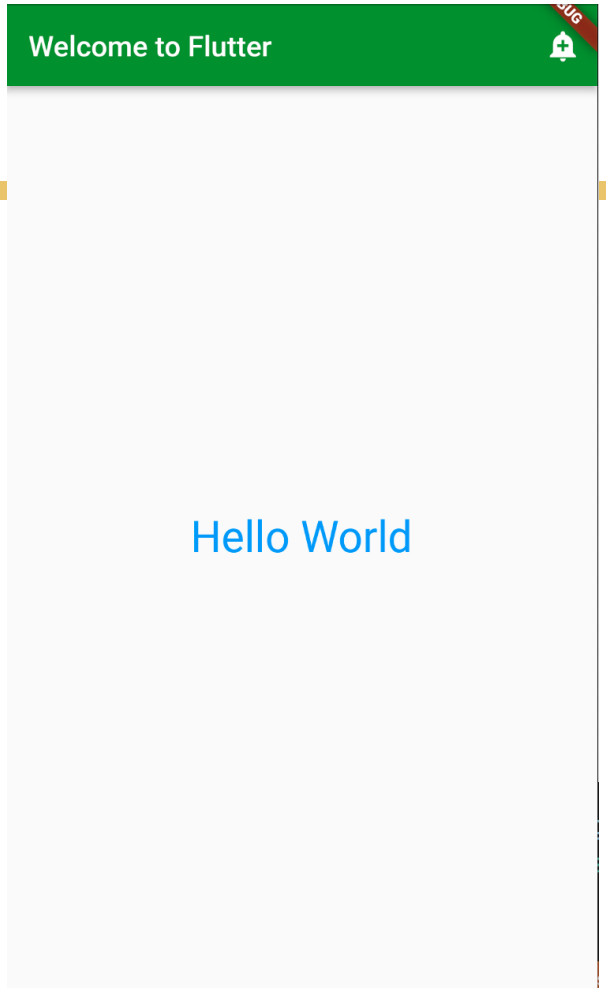
AppBar is divided into five areas, leading, title, Tool Bar (actions), flexiableSpace, and bottom.



AppBar: title

A simple AppBar consists of a title put in a Scaffold.

It will appear on the top of the Scaffold.



AppBar: Leading

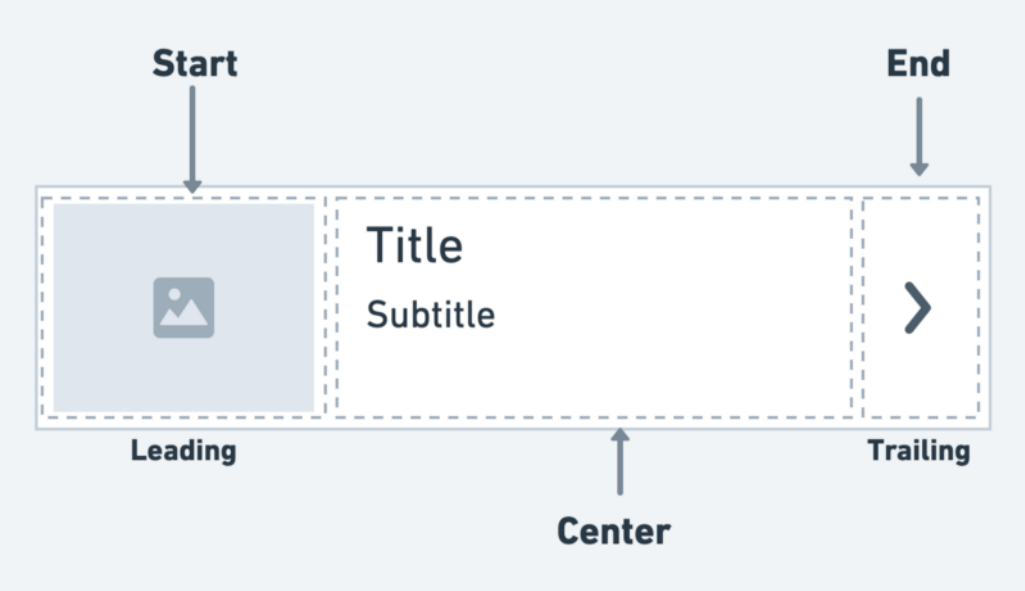
leading takes in a widget and can be assigned anything — text, an icon, or even multiple widgets within a row.

automaticallyImplyLeading is an optional property of the AppBar, whose default value is true.

When you do not place any Widget in the leading area, an appropriate Widget may be automatically put in it contextually.

ListTile

The ListTile widget in Flutter is a UI element that displays related information.



AppBar: Action and Icon

actions property allows you to add action(s) to the Tool bar of the AppBar.

Normally, IconButton will be used for each common action.

List of all icons : https://api.flutter.dev/flutter/material/Icons-class.html

ShowDialog/ Alert Box

We will use Alert Box to show the alert message.

Key Properties Of Alert Dialog

action: the set of action that displays bottom of the box.

title: The text which shows top in the dialog box.

content: This is used to give the message to the user according to the title.

elevation: It gives default show to the box.

background color: Used to set background color.

AppBar: bottom

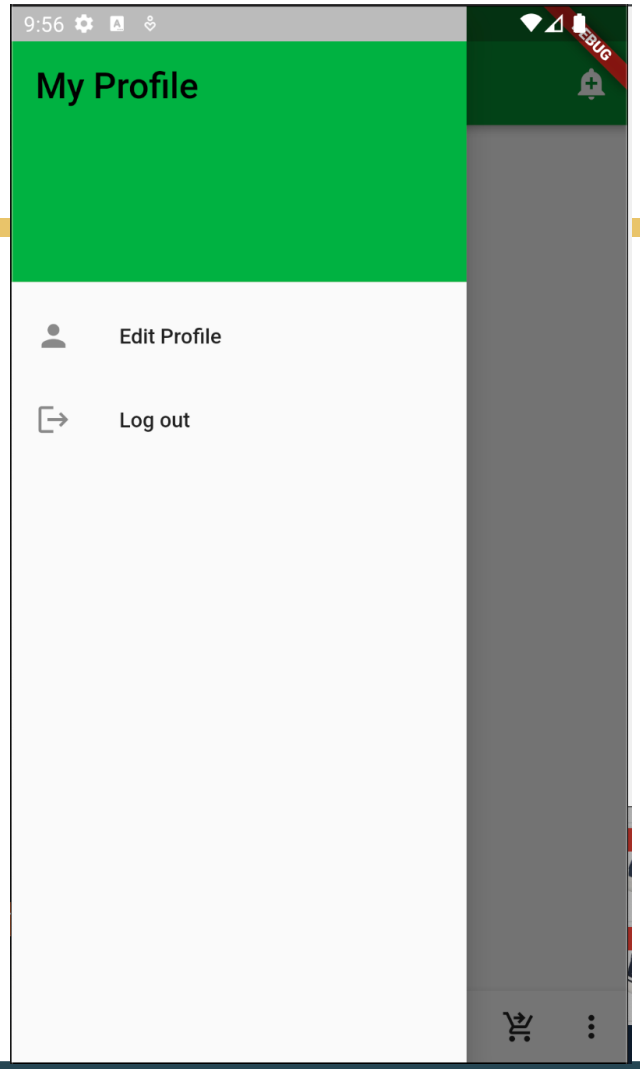
A container that is typically used with bottomNavigationBar in Scaffold

FloatingActionButton, which illustrates the FloatingActionButtonLocations in relation to the BottomAppBar.

Activity 3

ลองเพิ่ม Setting option ใน leading menu และใส่ icon ด้วยก็ได้

ลองเพิ่ม Alert Box ตอนกด Setting และให้แสดง ผลว่า ”Click Setting”



Week 8: Classroom game

ตอบคำถามตาม link นี้เลย...

https://forms.gle/zLDGAktFoPr59Mct6

Reference

https://dart-tutorial.com/introduction-and-basics/

https://www.tutorialspoint.com/flutter/flutter_introduction_to_widgets.htm

https://docs.flutter.dev/development/ui/widgets-intro