



LẬP TRÌNH ĐA NỀN TẢNG

GV: TRẦN THANH TUẤN

tttuan@caothang.edu.vn

```
elif_operation == "MIRROR_X":
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
elif_operation == "MIRROR_Z":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

#selection at the end -add back the deselected mirror modifier object
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("Selected" + str(modifier_ob)) # modifier ob is the active ob
#mirror_ob.select = 0
name = bpy.context.selected_objects[0]
obj.data.objects[name].select = 1

print("Please select exactly one object, the last one gets the modifier added")
```



WIDGET

CHƯƠNG 2: CÁC KHÁI NIỆM CƠ BẢN

Cấu trúc của project Flutter

▼ TEST_01

> .dart_tool

> .idea

> android

Thư mục code cho ứng dụng trên nền tảng
Android

> build

> ios

Thư mục code cho ứng dụng trên nền t

▼ lib

Thư mục chính chứa Dart code của

main.dart

Entry-point của ứng dụng Flu

> linux

Thư mục code cho ứng dụng trên nền t

> macos

Thư mục code cho ứng dụng trên nền tảng
MaxOS

> test

> web

> windows

Thư mục code cho ứng dụng trên nền tảng
Windows

◆ .gitignore

≡ .metadata

! analysis_options.yaml

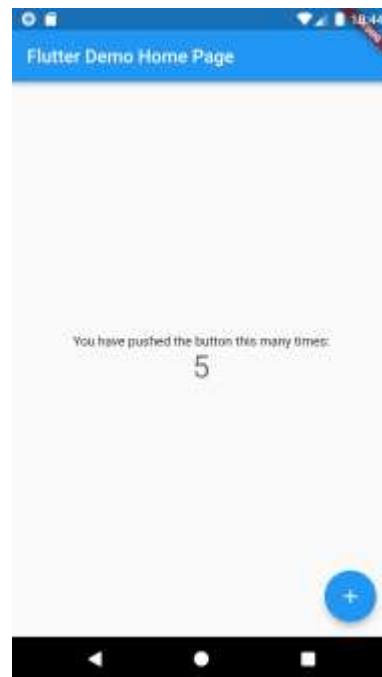
≡ pubspec.lock

! pubspec.yaml

Chứa các thông tin cấu hình của project, các thư viện, các tài nguyên (hình ảnh
, icon ...)

① README.md

🔥 test_01.iml



Cấu trúc của project Flutter

main.dart > ...

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

Khai báo các package

Run | Debug | Profile

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

Hàm main(): chạy ứng dụng

```
class MyApp extends StatelessWidget { ...
```

```
class MyHomePage extends StatefulWidget { ...
```

```
class _MyHomePageState extends State<MyHomePage> { ...
```



Cấu trúc của project Flutter

```
class MyApp extends StatelessWidget {  
  const MyApp({Key? key}) : super(key: key);  
  
  // This widget is the root of your application.  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      title: 'Flutter Demo',  
      theme: ThemeData(  
        primarySwatch: Colors.blue,  
      ), // ThemeData  
      home: 'Flutter Demo Home Page',  
    ); // MaterialApp  
  }  
}
```



Cấu trúc của project Flutter

```
class MyHomePage extends StatefulWidget {  
  const MyHomePage({Key? key, required this.title}) : super(key: key);  
  
  final String title;  
  
  @override Quản lý trạng thái của widget  
  State<MyHomePage> createState() => _MyHomePageState();  
}
```



Cấu trúc của project Flutter

```
class _MyHomePageState extends State<MyHomePage> {  
  int _counter = 0;  
  
  void _incrementCounter() {  
    setState(() {  
      _counter++;  
    });  
  }  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text(widget.title),  
      ), // AppBar  
      body: Center(  
        child: Column(  
          mainAxisAlignment: MainAxisAlignment.center,  
          children: <Widget>[  
            const Text(  
              'You have pushed the button this many times:',  
            ), // Text  
            Text(  
              '$_counter',  
              style: Theme.of(context).textTheme.headline4,  
            ), // Text  
          ], // <Widget>[]  
        ), // Column  
      ), // Center  
      floatingActionButton: FloatingActionButton(  
        onPressed: _incrementCounter,  
        tooltip: 'Increment',  
        child: const Icon(Icons.add),  
      ), // FloatingActionButton  
    ); // Scaffold  
  }  
}
```



Hello Word

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to Flutter',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Welcome to Flutter'),
        ),
        body: const Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```



Widget

Everything's a widget in Flutter

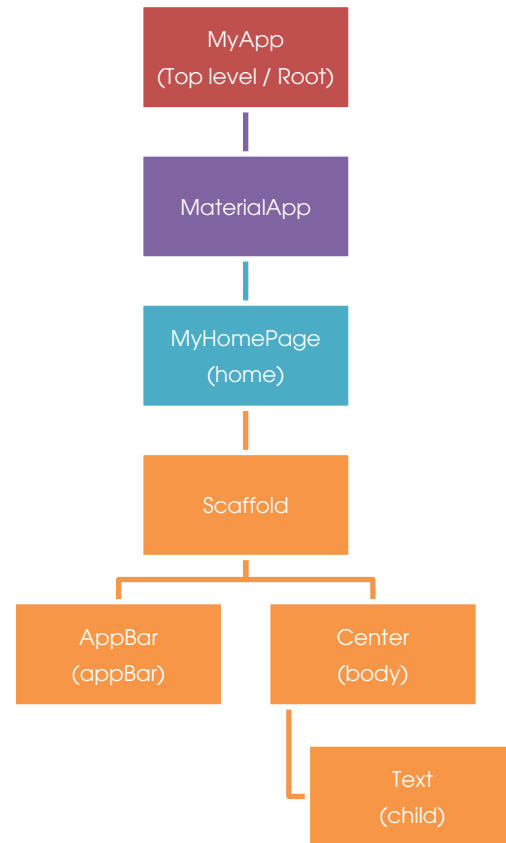
- Là thành phần giao diện cơ bản nhất tạo nên toàn bộ giao diện người dùng của ứng dụng.
- Mỗi ứng dụng chính là một **Top-level** widget.
- Mỗi widget có thể bao gồm một hoặc nhiều widget con → tạo thành cây widget



Widget

Cấu trúc widget của Ứng dụng **Hello World**

```
class MyApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      title: 'Welcome to Flutter',  
      home: Scaffold(  
        appBar: AppBar(  
          title: const Text('Welcome to Flutter'),  
        ), // AppBar  
        body: const Center(  
          child: Text('Hello World'),  
        ), // Center  
      ), // Scaffold  
    ); // MaterialApp  
  }  
}
```



Widget

Có 4 loại dựa trên chức năng:

- Các widget giao diện đặc thù theo từng nền tảng – *Platform specific widgets*
- Các widget hỗ trợ bố trí giao diện - *Layout widgets*
- Các widget quản lý trạng thái - *State maintenance widgets*
- Các widget cơ bản độc lập với nền tảng - *Platform independent / basic widgets*



Widget

Các widget giao diện đặc thù theo từng nền tảng – *Platform specific widgets*

- Là các widget dành riêng cho từng nền tảng Android hay iOS.
- **Material widgets:** các widget dành riêng cho Android được thiết kế theo *Material design guideline* cho Android.

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



Widget

Các widget giao diện đặc thù theo từng nền tảng – *Platform specific widgets*

- **Cupertino widgets:** các widget dành riêng cho iOS được thiết kế theo *Human Interface Guidelines* bởi Apple.

CupertinoButton	CupertinoPicker	CupertinoDatePicker	CupertinoTimerPicker
CupertinoNavigationBar	CupertinoTabBar	CupertinoTabScaffold	CupertinoTabView
CupertinoTextField	CupertinoDialog	CupertinoDialogAction	CupertinoFullscreenDialogTransition
CupertinoPageScaffold	CupertinoPageTransition	CupertinoActionSheet	CupertinoActivityIndicator
CupertinoAlertDialog	CupertinoPopupSurface	CupertinoSlider	



Widget

Các widget hỗ trợ bố trí giao diện - *Layout widgets*

- Một widget có thể được tạo thành từ một hoặc nhiều widget khác (thông qua các layout widget)
 - **Container** – Một hình chữ nhật (*BoxDecoration widgets*) với background (nền), border (đường viền) và shadow (bóng đổ).
 - **Center** – Căn giữa các widget con.
 - **Row** – Sắp xếp các widget con theo hàng ngang (horizontal direction).
 - **Column** – Sắp xếp các widget con theo hàng dọc (vertical direction).
 - **Stack** – Sắp xếp các widget con lên trên cùng.



Widget

Các widget quản lý trạng thái - *State maintenance widgets*

- Tất cả widget được kế thừa từ *StatelessWidget* hoặc *StatefulWidget*



Widget

Các widget cơ bản độc lập với nền tảng - *Platform independent / basic widgets*

- **Text widget** được sử dụng để hiển thị một đoạn văn bản.
- **Image widget** được sử dụng để hiển thị hình ảnh trong ứng dụng.
- ...

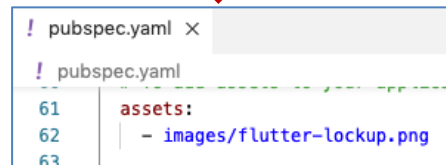
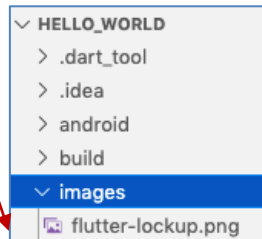


Widget

Ví dụ



```
class MyApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      title: 'Widget Flutter',  
      home: Scaffold(  
        appBar: AppBar(  
          title: const Text('Bài tập 1'),  
        ), // AppBar  
        body: Center(  
          child: Column(  
            children: [  
              Padding(  
                padding: const EdgeInsets.all(15),  
                child: Image.asset('images/flutter-lockup.png'),  
              ), // Padding  
              const Padding(  
                padding: EdgeInsets.all(15),  
                child: TextField(  
                  decoration: InputDecoration(  
                    border: OutlineInputBorder(),  
                    labelText: 'Số Nguyên N',  
                  ), // InputDecoration  
                ), // TextField  
              ), // Padding  
              Padding(  
                padding: const EdgeInsets.all(15),  
                child: ElevatedButton(  
                  onPressed: () {},  
                  child: const Text('Đọc số'),  
                ), // ElevatedButton  
              ), // Padding  
            ],  
          ), // Column  
        ), // Center  
      ), // Scaffold  
    ); // MaterialApp  
  }  
}
```



Widget

Bài tập



Text Widget

- Hiển thị và định dạng đoạn văn bản

```
Text(  
  'Hello, Tuan! How are you?',  
  style: TextStyle(  
    fontWeight: FontWeight.bold,  
  ),  
)
```



Hello, Tuan! How are you?

```
Text(  
  'Hello, Tuan! How are you?',  
  style: TextStyle(  
    fontStyle: FontStyle.italic,  
  ),  
)
```



Hello, Tuan! How are you?



Text Widget

- Hiển thị và định dạng đoạn văn bản

```
Text(  
  'Hello, Tuan! How are you?',  
  style: TextStyle(  
    height: 5,  
    fontSize: 25,  
  ),  
)
```

the line height is set to 5 times the font size, so that the text is very spaced out. Since the fontSize is set to 25, the final height of the line is 125 pixels.



Hello, Tuan! How are you?



Text Widget

- Hiển thị và định dạng đoạn văn bản

```
Text(  
  'Hello, Tuan!',  
  style: TextStyle(  
    decoration: TextDecoration.underline,  
    decorationColor: Colors.red,  
    decorationStyle: TextDecorationStyle.wavy,  
  ),  
)
```



Hello, Tuan!

```
Text(  
  'Hello, Tuan! How are you?',  
  style: TextStyle(  
    color: Colors.red,  
  ),  
)
```



Hello, Tuan! How are you?



Text Widget

Hiển thị và định dạng đoạn văn bản

```
Text(  
  'Hello, Tuan! How are you?',  
  style: TextStyle(  
    color: Color(0xFF08080),  
  ),  
)
```

Opacity:
- FF (100%)
- F2 (95%)
- E6 (90%)
- ...

#F08080

Hello, Tuan! How are you?



Text Widget

- Hiển thị và định dạng đoạn văn bản

```
Stack(  
  children: [  
    // Stroked text as border.  
    Text(  
      'Welcome to!',  
      style: TextStyle(  
        fontSize: 40,  
        foreground: Paint()  
          ..style = PaintingStyle.stroke  
          ..strokeWidth = 6  
          ..color = Colors.blue[700]!,  
      ),  
    ),  
    // Solid text as fill.  
    Text(  
      'Welcome to!',  
      style: TextStyle(  
        fontSize: 40,  
        color: Colors.grey[300],  
      ),  
    ),  
  ],  
)
```



Welcome to!



Text Widget

Hiển thị và định dạng đoạn văn bản

```
Text(  
  'Welcome to!',  
  style: TextStyle(  
    fontSize: 40,  
    foreground: Paint()  
      ..shader = ui.Gradient.linear(  
        const Offset(0, 20),  
        const Offset(150, 20),  
        [  
          Colors.red,  
          Colors.yellow,  
        ],  
      ),  
  ),  
)
```

```
import 'dart:ui' as ui;
```



Welcome to!



Text Widget

Hiển thị và định dạng đoạn văn bản

```
Text(  
  'Welcome to!',  
  style: TextStyle(  
    fontFamily: 'UTMNeoSansIntel',  
  ),  
)
```

Welcome to!

! pubspec.yaml

```
fonts:  
  - family: UTMNeoSansIntel  
    fonts:  
      - asset: fonts/UTM-Neo-Sans-Intel.ttf  
      - asset: fonts/UTM-Neo-Sans-Intel-Bold.ttf  
      weight: 500
```

✓ HELLO_WORLD

- > .dart_tool
- > .idea
- > android
- > build

✓ fonts

- ⚠ UTM-Neo-Sans-Intel-Bold.ttf
- ⚠ UTM-Neo-Sans-Intel.ttf



Text Widget

Hiển thị và định dạng đoạn văn bản

```
Text.rich(  
  TextSpan(  
    text: 'Welcome', // default text style  
    children: <TextSpan>[  
      TextSpan(  
        text: ' to ',  
        style: TextStyle(fontStyle: FontStyle.italic)),  
      TextSpan(  
        text: 'Flutter',  
        style: TextStyle(fontWeight: FontWeight.bold)),  
    ],  
  ),  
)
```



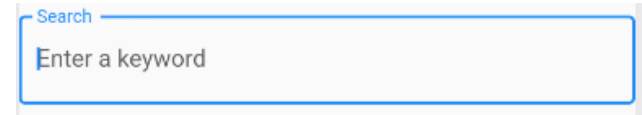
Welcome to **Flutter**



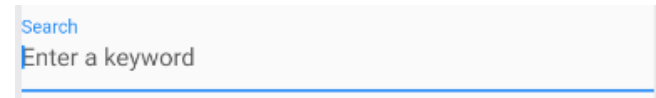
TextField Widget

Được sử dụng để nhập dữ liệu văn bản.

```
TextField(  
  decoration: InputDecoration(  
    border: OutlineInputBorder(),  
    hintText: 'Enter a keyword',  
    labelText: 'Search',  
  ),  
)
```

A visual representation of a TextField widget with an OutlineInputBorder. It features a light gray background, a blue border, and a blue label 'Search' at the top left. The placeholder text 'Enter a keyword' is displayed in a light gray font.

```
TextField(  
  decoration: InputDecoration(  
    border: UnderlineInputBorder(),  
    hintText: 'Enter a keyword',  
    labelText: 'Search',  
  ),  
)
```

A visual representation of a TextField widget with an UnderlineInputBorder. It features a light gray background, a blue underline, and a blue label 'Search' at the top left. The placeholder text 'Enter a keyword' is displayed in a light gray font.

TextField Widget

Được sử dụng để nhập dữ liệu văn bản.

```
TextField(  
  obscureText: true,  
  decoration: InputDecoration(  
    border: OutlineInputBorder(),  
    hintText: 'Enter a password',  
    labelText: 'Password',  
  ),  
)
```

Two visual representations of the TextField widget. The top one shows the widget with the label 'Password' and the hint text 'Enter a password'. The bottom one shows the widget with the label 'Password' and three dots '...' indicating the input field.

TextFormField, which integrates with the Form widget.



TextField Widget

Trích xuất giá trị trong TextField

- Tạo một đối tượng TextEditingController và gán cho một TextField hoặc TextFormField.

```
final textFieldController = TextEditingController();

@override
void dispose() {
  // Clean up the controller when the widget is disposed.
  textFieldController.dispose();
  super.dispose();
}
```

```
TextField(
  decoration: const InputDecoration(
    border: OutlineInputBorder(),
    labelText: 'Văn bản',
  ),
  controller: textFieldController,
)
```



TextField Widget

Trích xuất giá trị trong TextField

- Lấy / gán giá trị của TextField: sử dụng thuộc tính text của TextEditingController

```
textFieldController.text = 'New Value';  
Text(textFieldController.text) |
```



TextField Widget

Trích xuất giá trị trong TextField

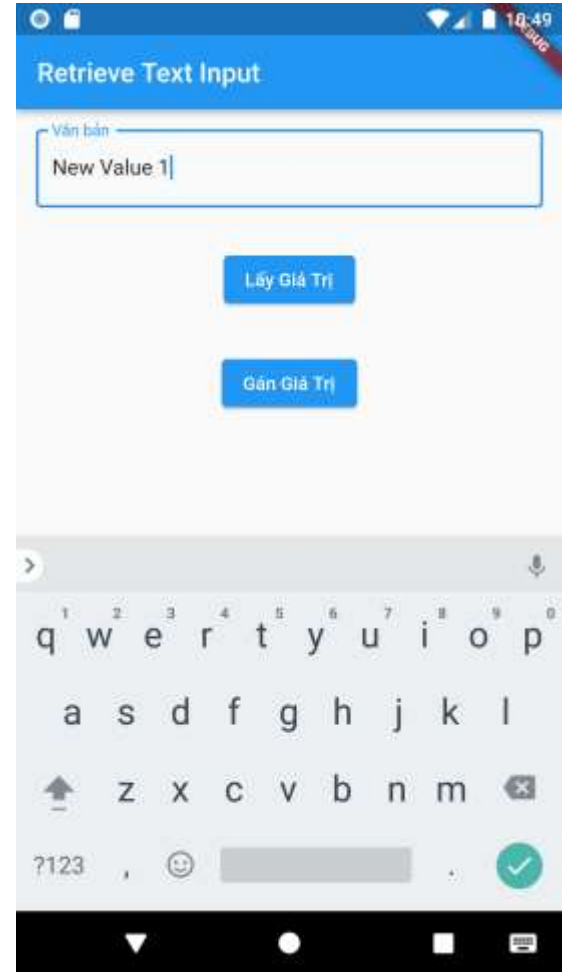
Ví dụ:

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

Run | Debug | Profile
void main() => runApp(const MyApp());

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

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      title: 'Widget Flutter',
      home: MyHomePage(),
    ); // MaterialApp
  }
}
```



TextField Widget

Trích xuất giá trị trong TextField

Ví dụ (tiếp theo):

```
class MyHomePage extends StatefulWidget {  
  const MyHomePage({Key? key}) : super(key: key);  
  
  @override  
  _MyCustomState createState() => _MyCustomState();  
}
```



TextField Widget

Trích xuất giá trị trong TextField

Ví dụ (tiếp theo):

```
class _MyCustomState extends State<MyHomePage> {  
  final textFieldController = TextEditingController();  
  
  @override  
  void dispose() {  
    // Clean up the controller when the widget is disposed.  
    textFieldController.dispose();  
    super.dispose();  
  }  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: const Text('Retrieve Text Input'),  
      ), // AppBar  
      body: Center(  

```



TextField Widget

Trích xuất giá trị trong TextField

Ví dụ (tiếp theo):

```
child: Column(  
  children: [  
    Padding(  
      padding: const EdgeInsets.all(15),  
      child: TextField(  
        decoration: const InputDecoration(  
          border: OutlineInputBorder(),  
          labelText: 'Văn bản',  
        ), // InputDecoration  
        controller: textFieldController,  
      ), // TextField  
    ), // Padding  
    Padding(  
      padding: const EdgeInsets.all(15),  
      child: ElevatedButton(  
        onPressed: () {  
          print(textFieldController.text);  
        },  
        child: const Text('Lấy Giá Trị'),  
      ), // ElevatedButton  
    ), // Padding  
  ],  
)
```

```
        Padding(  
          padding: const EdgeInsets.all(15),  
          child: ElevatedButton(  
            onPressed: () {  
              textFieldController.text = 'New Value';  
            },  
            child: const Text('Gán Giá Trị'),  
          ), // ElevatedButton  
        ), // Padding  
      ],  
    ), // Column  
  ), // Center  
}
```



AlertDialog Widget

Hiển thị hộp thoại thông báo

```
showDialog(  
  context: context,  
  builder: (context) {  
    return AlertDialog(  
      content: Text(messageText),  
    ); // AlertDialog  
  },  
);
```

Đối tượng BuilderContext

Xin chào

messageText = 'Xin chào';



AlertDialog Widget

Hiển thị hộp thoại thông báo

Đối tượng BuilderContext

```
showDialog(  
  context: context,  
  builder: (context) => AlertDialog(  
    title: const Text('AlertDialog Title'),  
    content: const Text('AlertDialog description'),  
    actions: [  
      TextButton(  
        onPressed: () => Navigator.pop(context, 'Cancel'),  
        child: const Text('Cancel'),  
      ), // TextButton  
      TextButton(  
        onPressed: () => Navigator.pop(context, 'OK'),  
        child: const Text('OK'),  
      ), // TextButton  
    ],  
  ), // AlertDialog  
);
```

AlertDialog Title

AlertDialog description

Cancel OK

barrierDismissible: false, // user must tap button!



ScaffoldMessenger Widget

Hiển thị Snackbar ở biên dưới màn hình.

Chứa nội dung văn bản thông báo và tùy chọn một hành động nào đó.
Tự động hiện ẩn đi, có thể vuốt để ẩn đi hoặc chứa nút bấm hành động.

```
ScaffoldMessenger.of(context)
  ..removeCurrentSnackBar()
  ..showSnackBar(
    const SnackBar(
      content: Text('Hello world'),
    ), // SnackBar
  );
```

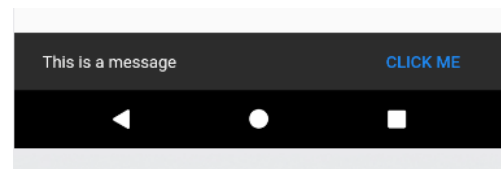


ScaffoldMessenger Widget

Hiển thị Snackbar ở biên dưới màn hình.

Chứa nội dung văn bản thông báo và tùy chọn một hành động nào đó.
Tự động hiện ẩn đi, có thể vuốt để ẩn đi hoặc chứa nút bấm hành động.

```
ScaffoldMessenger.of(context)
  ..removeCurrentSnackBar()
  ..showSnackBar(SnackBar(
    content: const Text('This is a message'),
    duration: const Duration(seconds: 5),
    action: SnackBarAction(
      label: 'CLICK ME',
      onPressed: () {
        print('Hello world');
      },
    ), // SnackBarAction
  )); // SnackBar
```



Flutter Widget of the Week

<https://www.youtube.com/playlist?list=PLjxrf2q8roU23XGwz3Km7sQZFTdB996iG>

Flutter Widget of the Week

Flutter

164 videos • 5,344,388 total views • Cập nhật lần cuối vào 3...

Phát tất cả Trộn bài

Fighting the good fight for Widget Awareness!
Widget of the Week is a series of quick, animated videos, each of which covers a particular widget from the Flutter SDK. Need to know why you'd choose to use a SafeArea widget, or how to position a child within a Stack, but don't have much time? Then, this series is for you!

Subscribe to Flutter --> <http://goo.gle/FlutterYT>

Video Number	Video Title	Category	Views
1	Uint8List	Technique of the Week	1.46
2	Mix	Package of the Week	1.37
3	Completers	Technique of the Week	2.09
4	firebase_vertexai	Package of the Week	2.14
5	Future.wait	Technique of the Week	1.09
6	List.generate	Technique of the Week	1.56
7	Tween	Technique of the Week	2.27
8	Isolates	Technique of the Week	2.14



Bài tập

Xử lý các tác vụ khi nhấn vào các Button Đọc số, Ước Chung Lớn Nhất, Giải phương trình bậc hai và hiển thị kết quả lên AlertDialog / SnackBar.

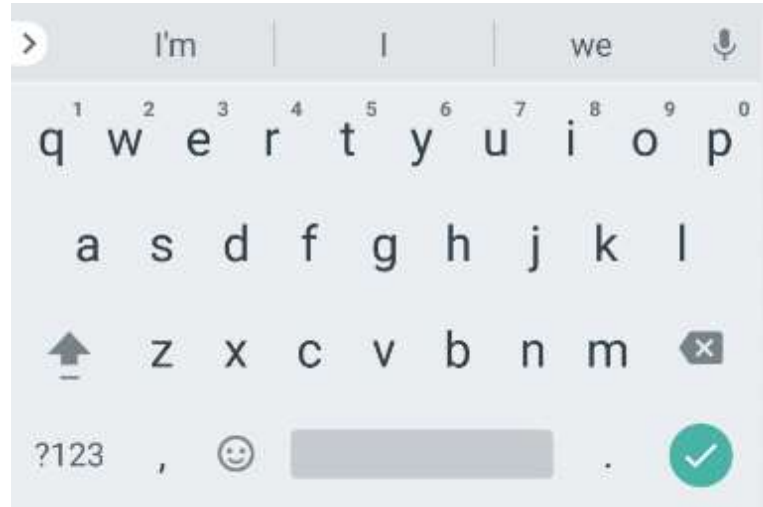


TextField Widget (tt)



Các loại bàn phím ảo nhập liệu (keyboardType):

Text (TextInputType.text)

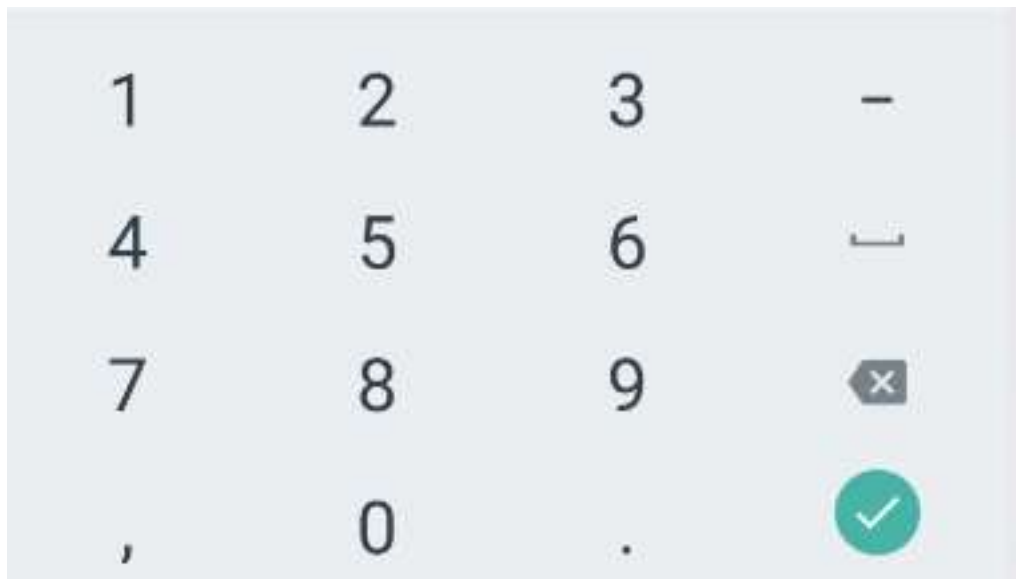


TextField Widget (tt)



Các loại bàn phím ảo nhập liệu (keyboardType):

Number (TextInputType.number)

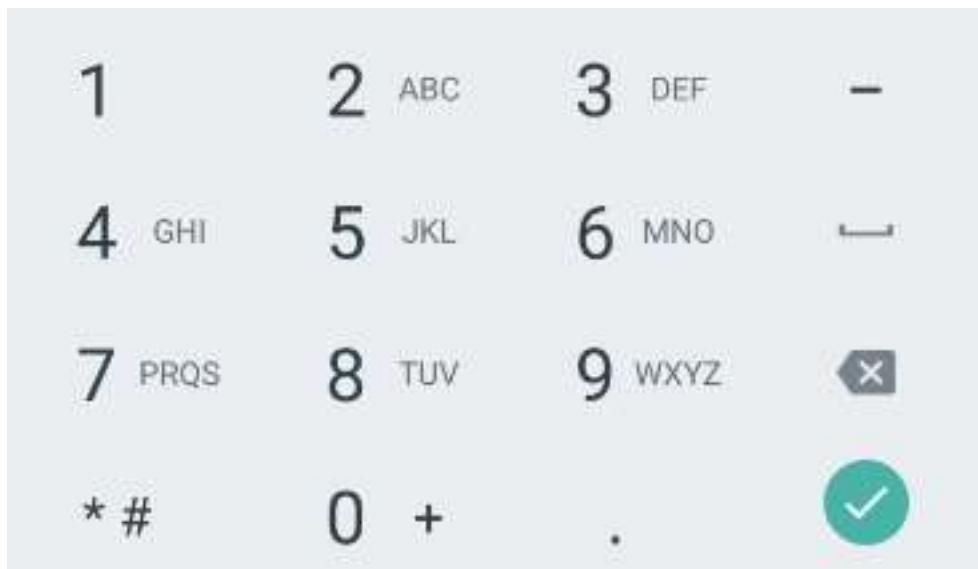


TextField Widget (tt)



Các loại bàn phím ảo nhập liệu (keyboardType):

Phone (TextInputType.phone)



TextField Widget (tt)



Các loại bàn phím ảo nhập liệu (keyboardType):

Email (TextInputType.emailAddress)



TextField Widget (tt)



Các loại bàn phím ảo nhập liệu (keyboardType):

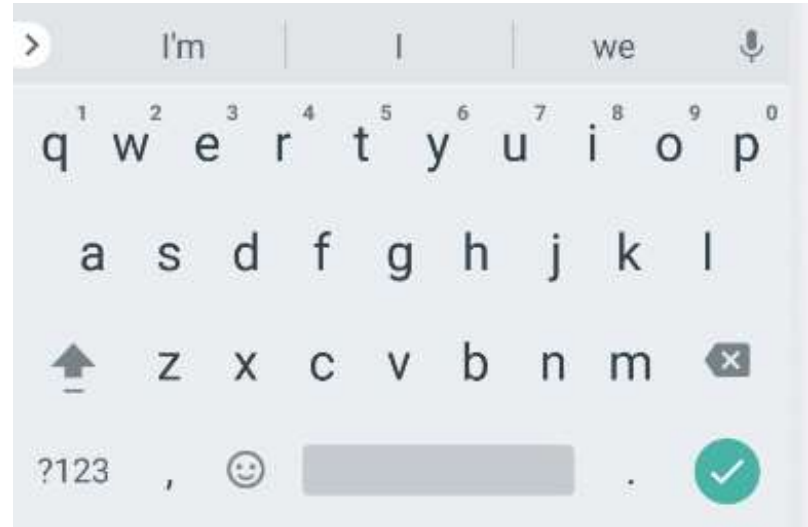
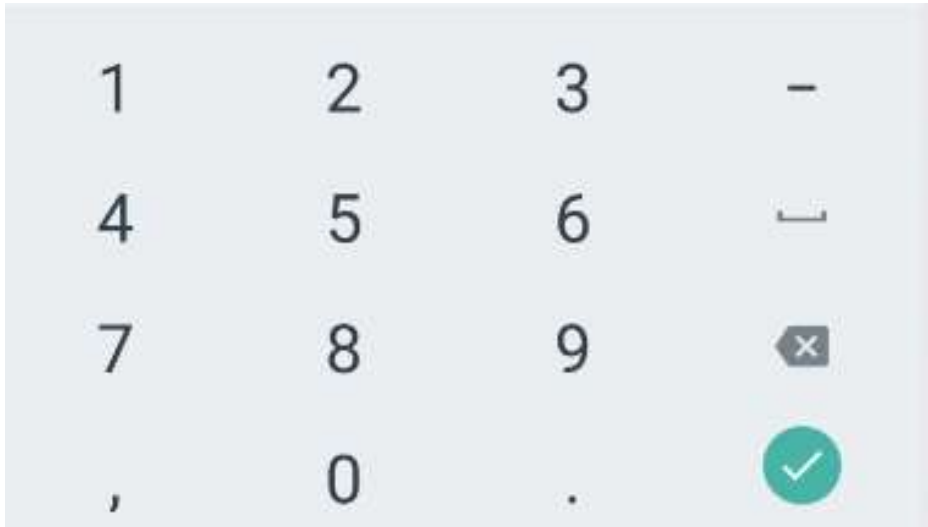
URL (TextInputType.url)



TextField Widget (tt)



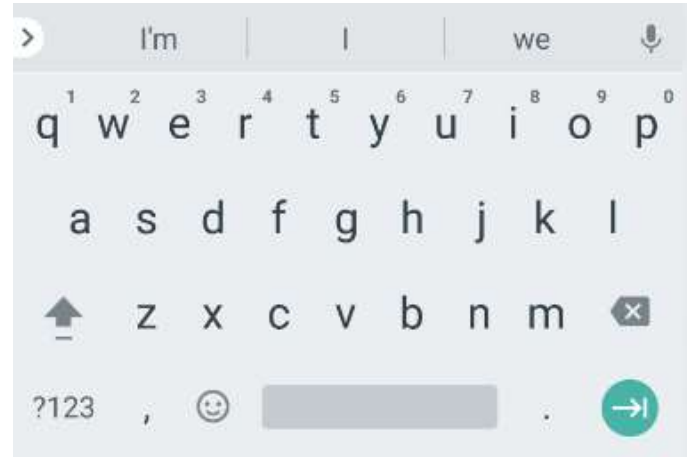
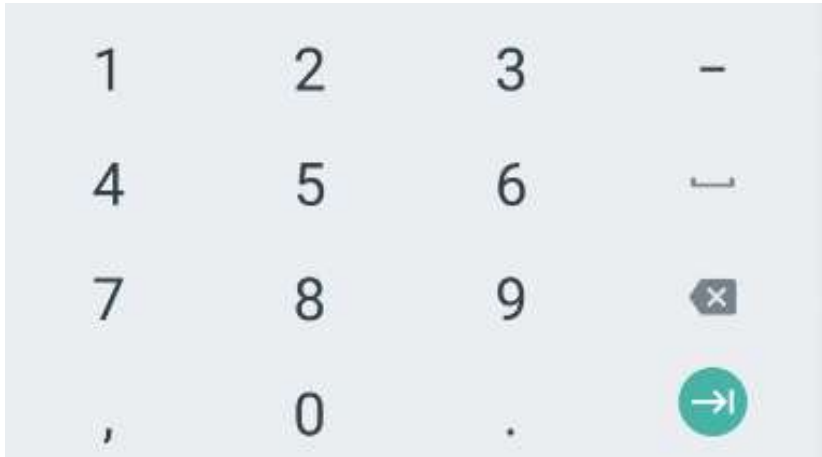
textInputAction:
TextInputAction.done



TextField Widget (II)



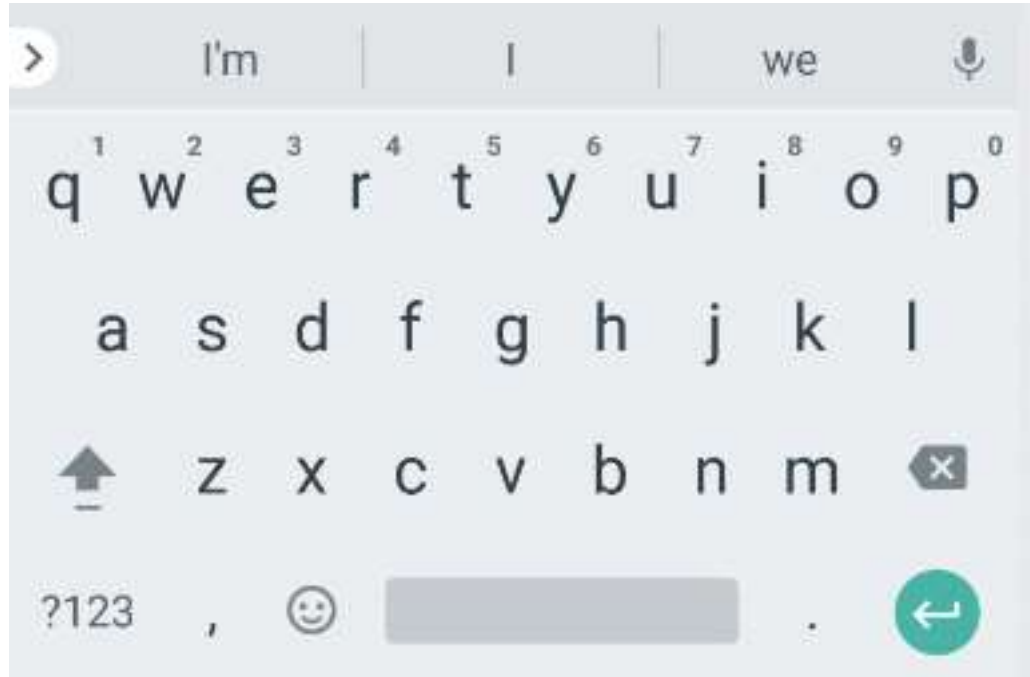
textInputAction:
TextInputAction.next



TextField Widget (II)



```
textInputAction:  
TextInputAction.newline
```



TextField Widget (tt)



textInputAction:
TextInputAction.go
TextInputAction.previous

TextField Widget (tt)



Ấn bàn phím ảo

```
@override
Widget build(BuildContext context) {
  ketQua.text = '0';
  return GestureDetector(
    onTap: () {
      FocusScopeNode currentFocus = FocusScope.of(context);
      if (!currentFocus.hasPrimaryFocus) {
        currentFocus.unfocus();
      }
    },
    child: Scaffold(
      appBar: AppBar(
        title: const Text('Ấn bàn phím ảo'),
      ), // AppBar
      body: SingleChildScrollView(
        keyboardDismissBehavior: ScrollViewKeyboardDismissBehavior.onDrag,
        child: Column(
          children: [
            Padding(
```

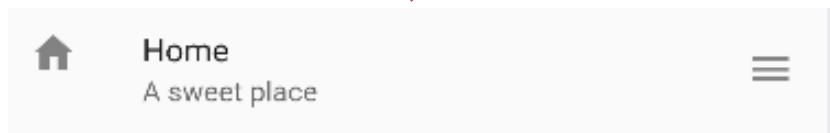
ListTile Widget



Có thể có từ 1 đến 3 dòng

Chứa các widget khác: Text, Icon ...

```
ListTile(  
  leading: Icon(Icons.home),  
  title: Text('Home'),  
  subtitle: Text('A sweet place'),  
  trailing: Icon(Icons.menu),  
)
```

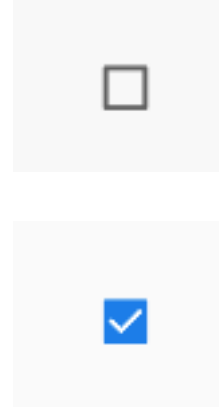


```
tileColor: Colors.red,
```



Checkbox Widget

```
bool? isChecked = true;  
  
Checkbox(  
  value: isChecked,  
  onChanged: (bool? value) {  
    setState(() {  
      isChecked = value;  
    });  
  },  
)
```



Checkbox Widget

```
activeColor: Colors.red,  
checkColor: Colors.yellow,  
tristate: true,  
side: const BorderSide(  
  color: Colors.red,  
  style: BorderStyle.solid,  
  width: 2,  
), // BorderSide
```



CheckboxListTile Widget

```
CheckboxListTile(  
  title: const Text('Flutter'),  
  value: isChecked,  
  onChanged: (bool? value) {  
    setState(() {  
      isChecked = value;  
    });  
  },  
  secondary: const Icon(Icons.accessibility_new),  
)
```



Flutter



Switch Widget

Thay đổi trạng thái on/off

```
bool isChecked = true;

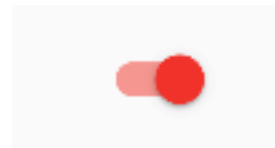
Switch(
  value: isChecked,
  onChanged: (bool value) {
    setState(() {
      isChecked = value;
    });
  },
)
```



Switch Widget

Thay đổi trạng thái on/off

```
activeColor: Colors.red,  
inactiveThumbColor: Colors.green,  
inactiveTrackColor: Colors.blue,
```



SwitchListTile Widget

```
SwitchListTile(  
  title: const Text('Home'),  
  value: isChecked,  
  onChanged: (bool value) {  
    setState(() {  
      isChecked = value;  
    });  
  },  
  secondary: const Icon(Icons.cabin),  
)
```

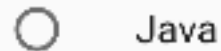


Home



Radio<T> Widget

```
String? _subject;  
  
ListTile(  
  title: const Text('Java'),  
  leading: Radio<String>(  
    value: 'java',  
    groupValue: _subject,  
    onChanged: (String? value) {  
      setState(() {  
        _subject = value;  
      });  
    },  
  ),  
)
```



Radio<T> Widget

```
enum GioiTinh { nam, nu }
```

```
GioiTinh _gioiTinh = GioiTinh.nam;
```

```
ListTile(  
  title: const Text('Nam'),  
  leading: Radio<GioiTinh>(  
    value: GioiTinh.nam,  
    groupValue: _gioiTinh,  
    onChanged: (GioiTinh? value) {  
      setState(() {  
        _gioiTinh = value!  
      });  
    },  
  ),  
)
```



Nam



Nữ

```
ListTile(  
  title: const Text('Nữ'),  
  leading: Radio<GioiTinh>(  
    value: GioiTinh.nu,  
    groupValue: _gioiTinh,  
    onChanged: (GioiTinh? value) {  
      setState(() {  
        _gioiTinh = value!  
      });  
    },  
  ),  
)
```



RadioListTile<T> Widget

```
RadioListTile<GioiTinh>(  
  title: const Text('Nam'),  
  value: GioiTinh.nam,  
  groupValue: _gioiTinh,  
  onChanged: (GioiTinh? value) {  
    setState(() {  
      _gioiTinh = value!;  
    });  
  },  
)  
,  
RadioListTile<GioiTinh>(  
  title: const Text('Nữ'),  
  value: GioiTinh.nu,  
  groupValue: _gioiTinh,  
  onChanged: (GioiTinh? va  
    setState(() {  
      _gioiTinh = value!;  
    });  
  },  
)  
,
```

☒ Nam

☐ Nữ



Slider Widget

Được sử dụng để chọn giá trị trong đoạn (a, b)

```
double _currentSliderValue = 20;

Slider(
  value: _currentSliderValue,
  min: 0,
  max: 100,
  divisions: 10,
  label: _currentSliderValue.round().toString(),
  onChanged: (double value) {
    setState(() {
      _currentSliderValue = value;
    });
  },
)
```



RangeSlider Widget

```
var _currentRangeValues = RangeValues(0.3, 0.7);


RangeSlider(
  values: _currentRangeValues,
  min: 0,
  max: 1,
  divisions: 10,
  labels: RangeLabels(
    '${_currentRangeValues.start}',
    '${_currentRangeValues.end}',
  ),
  onChanged: (RangeValues values) {
    setState(() {
      _currentRangeValues = values;
    });
  },
)
```



Bài tập 1

Nhập số thứ nhất, số thứ 2 → Chọn phép tính → Bấm vào nút “Kết quả” → hiển thị kết quả vào TextField Kết quả.

Lưu ý: làm tròn số hay không làm tròn số thì phụ thuộc vào Checkbox ‘Làm tròn số’



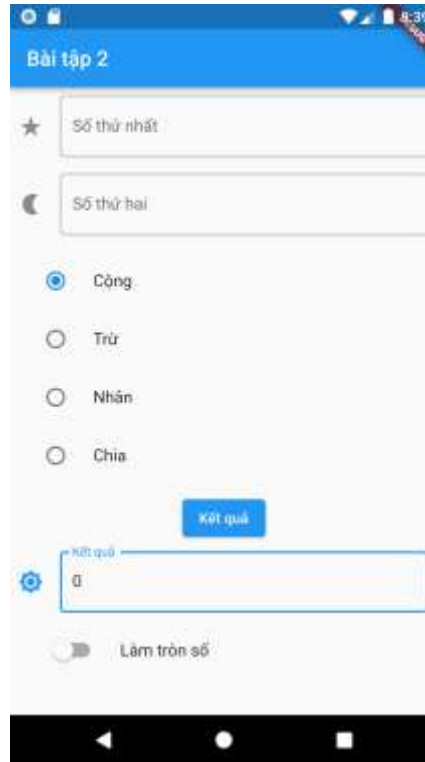
Bàn phím số

Readonly



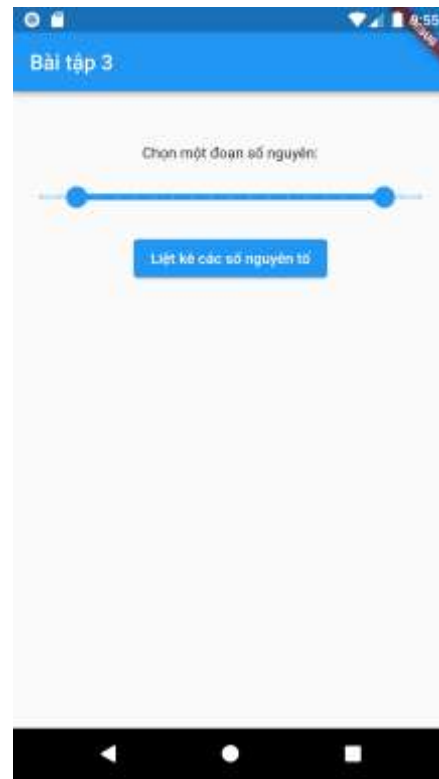
Bài tập 2

Tương tự như bài tập 1, nhưng thay thế Checkbox 'Làm tròn số' thành Switch Widget.



Bài tập 3

Chọn 1 đoạn số nguyên trên RangeSlider (min = 0, max = 1000, divisions = 20).
Hãy liệt kê các số nguyên tố trong đoạn vừa chọn
trên Dialog khi click vào button
"Liệt kê các số nguyên tố".



Review Widgets



- **Everything's a widget in Flutter**
- Widget tree

Review Widgets

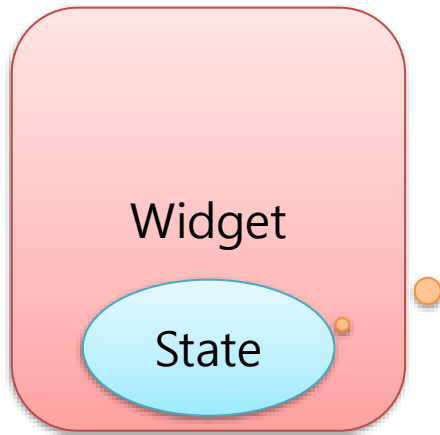


- Phương thức `Widget build(BuildContext context)`: thêm các widget “con” vào cây widget để tạo thành giao diện người dùng (UI)
- Tham số `BuildContext context`: cho biết thông tin quan trọng về vị trí của widget trên cây widget.
 - **Flutter** sử dụng thể hiện **BuildContext** để biết chi tiết về widget khi duyệt cây widget.
 - Trả về widget ***ExampleWidget*** gần nhất trên cây Widget: ***ExampleWidget.of(context)***
 - Mỗi widget có một thể hiện **BuildContext** → là “parent” context của các widget được trả về trong phương thức **build**.

Stateless và Stateful Widgets



- StatelessWidget
- StatefulWidget



Change or
Not?

- Người dùng chạm vào 1 button và "vài thứ" trên UI cần thay đổi
- Xoay thiết bị → UI cần được "vẽ lại"

...

Stateless và Stateful Widgets



- Stateless widget
 - Thành phần giao diện không thay đổi theo thời gian
 - Là một khối độc lập với các sự kiện
 - Chỉ dựa trên hàm khởi tạo và dữ liệu nội bộ bên trong

```
class MyName extends StatelessWidget {  
  const MyName({Key? key}) : super(key: key);  
  
  @override  
  Widget build(BuildContext context) => const Text('John Doe');  
}
```

```
class MyName extends StatelessWidget {  
  final String name;  
  const MyName({Key? key, required this.name}) : super(key: key);  
  
  @override  
  Widget build(BuildContext context) => Text(name);  
}
```

Stateless và Stateful Widgets



- Statefull widget
 - Thành phần giao diện sẽ thay đổi theo thời gian
 - Khi Fullter 'rebuild' cây widget để làm mới giao diện, phương thức **build(...)** của **State<T>** sẽ được gọi thực thi
 - Phương thức **setState(...)**: 'rebuild' widget
 - Các thuộc tính trong lớp con của **State<T>** vẫn tồn tại (không thay đổi), chỉ có bên trong phương thức **build()** mới bị thay đổi.

Stateless và Stateful Widgets



- Statefull widget

```
class Counter extends StatefulWidget {  
  const Counter({Key? key}) : super(key: key);  
  
  @override  
  _CounterState createState() => _CounterState();  
}
```

```
class _CounterState extends State<Counter> {  
  int _counter = 0;  
  @override  
  Widget build(BuildContext context) {  
    return Column(  
      children: [  
        Text("$_counter"),  
        ElevatedButton(  
          child: const Icon(Icons.add),  
          onPressed: () {  
            setState(() => _counter++);  
          }), // ElevatedButton  
      ],  
    ); // Column  
  }  
}
```

Stateless và Stateful Widgets



- Statefull widget
 - Sử dụng 'getter' **widget** để tham chiếu đến dữ liệu của **StatefulWidget**

```
class MyName extends StatefulWidget {  
  final String name;  
  const MyName({Key? key, required this.name}) : super(key: key);  
  
  @override  
  _MyNameState createState() => _MyNameState();  
}
```

```
class _MyNameState extends State<MyName> {  
  @override  
  Widget build(BuildContext context) => Text(widget.name);  
}
```


Rebuilds và Tối ưu hoá



- Flutter thường xuyên “duyet” cây widget → ‘rebuild’ các widget để làm mới UI
 - Phương thức **build()** được gọi thực thi nhiều hơn 1 lần trong suốt vòng đời của ứng dụng khi:
 - Lần đầu tiên UI được hiển thị
 - Gọi setState
 - Xoay màn hình thiết bị
 - ...
- **“allow rebuilds of widgets only when it’s really needed”**
- Sử dụng phương thức khởi tạo const của widget
- Flutter build các const widget chỉ 1 lần duy nhất

Row Widget



- Chứa các widget theo chiều ngang

```
Row(  
  mainAxisAlignment: MainAxisAlignment.center,  
  children: const [  
    Text("Hello"),  
    Text("Flutter!"),  
    Text("!!"),  
  ],  
),
```

- Mặc định Row widget sẽ bao phủ toàn bộ khoảng không gian theo chiều ngang (chiều rộng widget chứa Row widget), để Row widget “co lại” vừa với nội dung bên trong: `mainAxisSize: MainAxisSize.min`

Row Widget



- Canh lề các widget (`mainAxisAlignment`)

- center



- start



- end



- spaceAround



- spaceBetween



Column Widget



- Chứa các widget theo chiều dọc

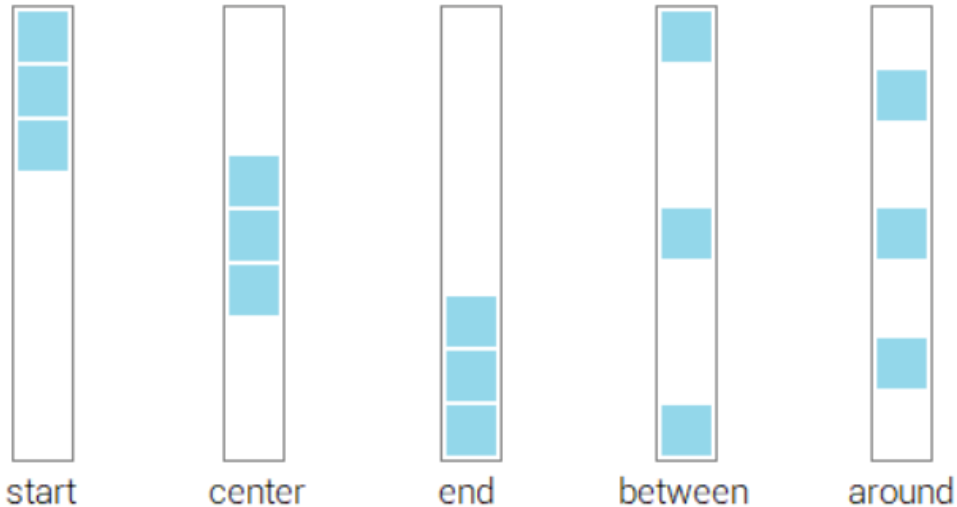
```
Column(  
  mainAxisAlignment: MainAxisAlignment.center,  
  children: const [  
    Text("Hello"),  
    Text("Flutter!"),  
    Text("!!!"),  
  ],  
),
```

- Tương tự như Row widget để Column widget “co lại” vừa với nội dung bên trong
: `mainAxisSize: MainAxisSize.min`

Column Widget



- Canh lề các widget (mainAxisAlignment)



- Lưu ý: Column widget không thể cuộn (scroll) nên nếu không đủ khoảng không gian để chứa các widget bên trong thì sẽ xuất hiện lỗi trong quá trình thực thi (run time error).

Listview Widget



- Chứa các widget theo chiều dọc

```
ListView(  
  children: const [  
    Text("Hello"),  
    Text("Flutter!"),  
    Text("!!"),  
  ],  
),
```

- Có thể cuộn (scroll)
 - Theo chiều dọc: `scrollDirection: Axis.vertical` (mặc định)
 - Theo chiều ngang: `scrollDirection: Axis.horizontal`

Listview Widget



- Với một số lượng lớn phần tử trong danh sách NÊN sử dụng phương thức khởi tạo định danh `ListView.builder(...)`

```
// Somewhere in the code there's a list of 100 integers
final myList = List<int>.generate(100, (i) => i);

// The 'builder' named constructor builds a list of widgets
// by taking the 'myList' list as data source.
ListView.builder(
  itemCount: myList.length,
  itemBuilder: (context, index) {
    return Text("${myList[index]}"),
  },
),
```

Container Widget



- Tương tự như thẻ `<div></div>` trong HTML

```
Container(  
  height: 80,  
  width: 260,  
  color: Colors.blueGrey,  
  alignment: Alignment.center,  
  transform: Matrix4.rotationZ(-0.25),  
  child: const Text(  
    "Containers!",  
    style: TextStyle(  
      color: Colors.white,  
      fontSize: 25  
    )  
  )  
)
```



Stack và Positioned Widget



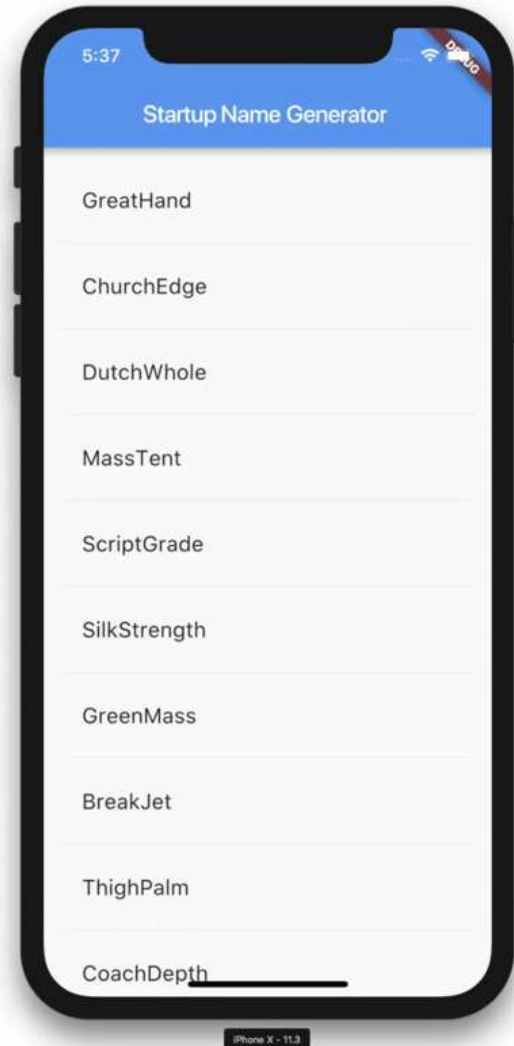
- **Stack** cho phép xếp chồng các widget và “tự do” xác định vị trí của widget trên màn hình bằng **Positioned**

```
Stack(  
  children: [  
    Container(  
      width: 40,  
      height: 40,  
      decoration: const BoxDecoration(  
        color: Colors.red  
      ),  
    ),  
    const Text("Hello"),  
  ],  
)
```

Positioned(
 top: 40,
 left: 65,
 child: Text("Hello"),
)

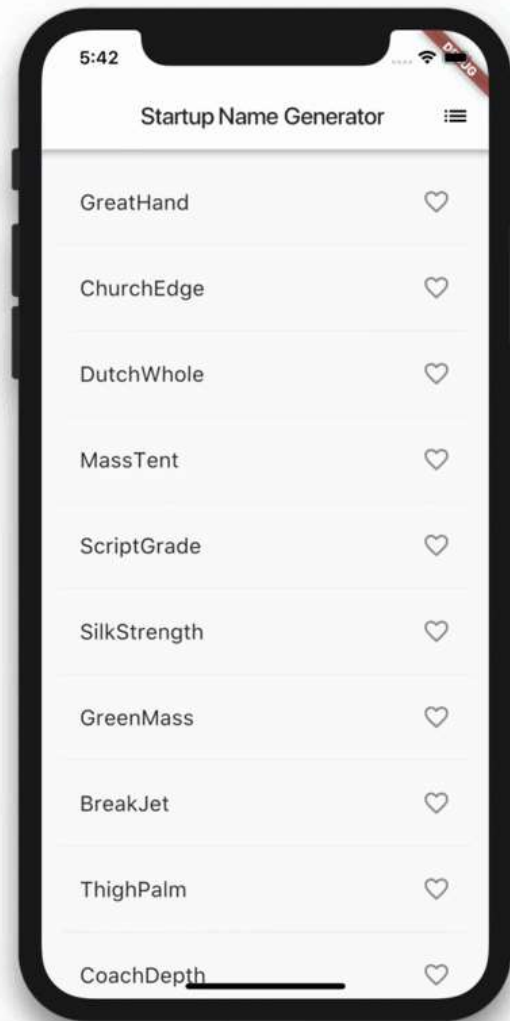
Bài tập 1

Xây dựng màn hình chứa
danh sách tên của các thành viên
trong lớp như hình minh họa bên cạnh



Bài tập 2

Xây dựng màn hình chứa
danh sách tên của các thành viên
trong lớp như hình minh họa bên cạnh





THANK YOU !!!

```
def operation == "MIRROR_X":
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
elif operation == "MIRROR_Z":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

#selection at the end -add back the deselected mirror modifier object
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("selected: " + str(modifier_ob)) # modifier ob is the active ob
mirror_ob.select = 0
# = bpy.context.selected_objects[0]
# bpy.data.objects[mirror_name].select = 1
print("please select exactly the objects, the last one gets the mirror modifier")
```