

Flutter: Các widget cơ bản

Bùi Võ Quốc Bảo

Khoa CNTT | Trường CNTT-TT | Đại học Cần Thơ



Tài liệu tham khảo

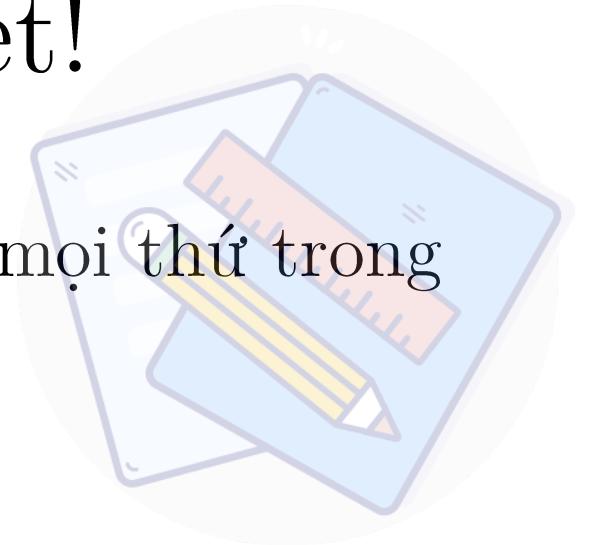
- Chương 3-4-5, **Flutter Apprentice** by Vincenzo Guzzi, Kevin D Moore, Vincent Ngo and Michael Katz
- <https://docs.flutter.dev/development/ui/widgets-intro>
- <https://docs.flutter.dev/development/ui/layout>



Các widget cơ bản



(Hầu như) mọi thứ đều là widget!

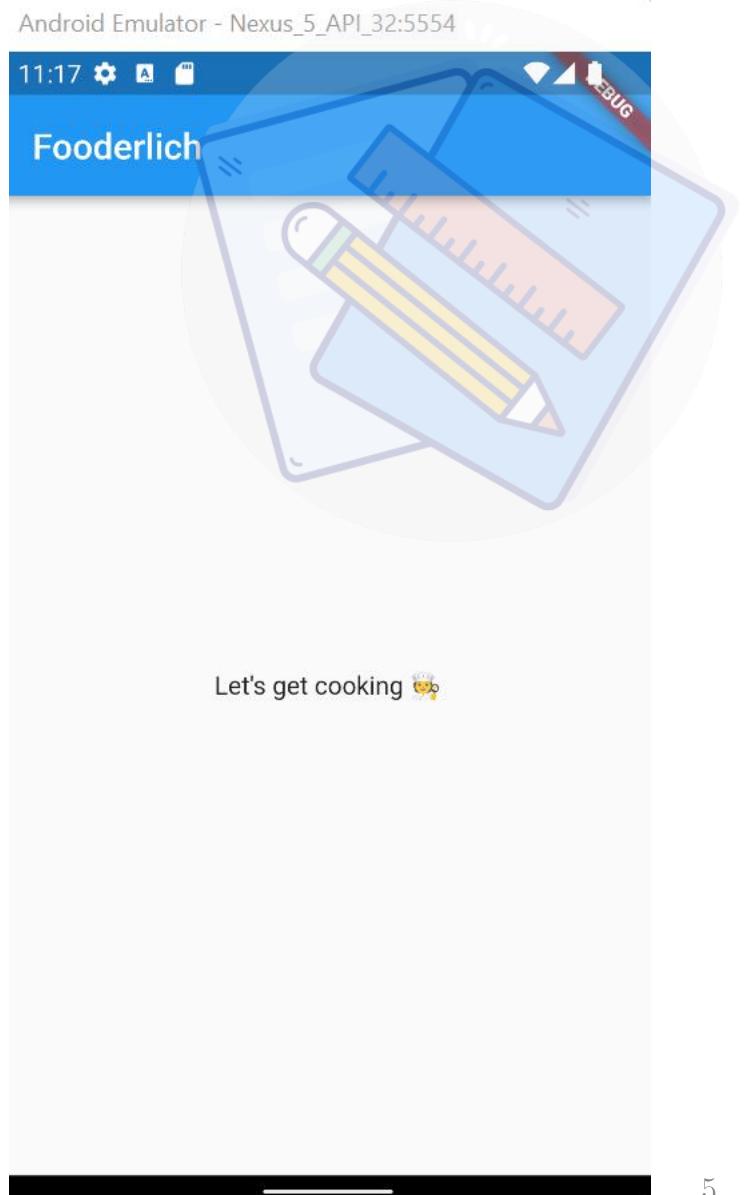


- Xét theo cách nhìn của nhà phát triển, hầu như mọi thứ trong Flutter đều là widget
- Các widget có thể được sử dụng cho:
 - Hiển thị thông tin (painting)
 - Bố cục / định vị trí (layout)
 - Nhận dạng cử chỉ (hit-testing)

Fooderlich

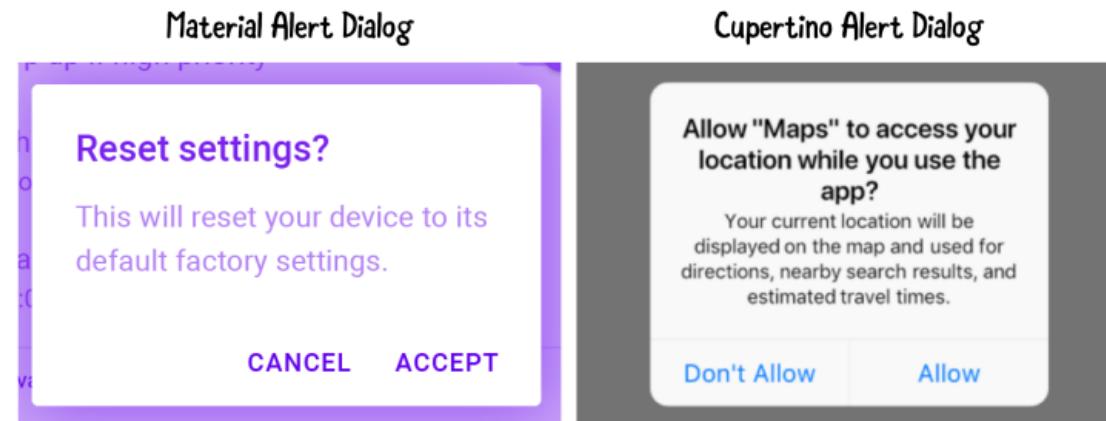
lib/main.dart

```
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(const Fooderlich());
5 }
6
7 class Fooderlich extends StatelessWidget {
8   const Fooderlich({Key? key}) : super(key: key);
9
10  @override
11  Widget build(BuildContext context) {
12    return MaterialApp(
13      title: 'Fooderlich',
14      home: Scaffold(
15        appBar: AppBar(title: const Text('Fooderlich')),
16        body: const Center(child: Text('Let\'s get cooking 🎩')),
17      ), // Scaffold
18    ); // MaterialApp
19  }
20 }
```

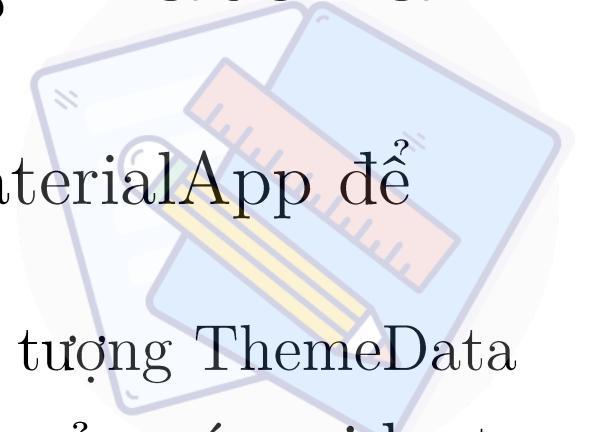


Các hệ thống thiết kế UI

- Android sử dụng hệ thống thiết kế Material
 - import 'package:flutter/material.dart';
 - <https://docs.flutter.dev/development/ui/widgets/material>
- iOS sử dụng hệ thống Cupertino
 - import 'package:flutter/cupertino.dart';
 - <https://docs.flutter.dev/development/ui/widgets/cupertino>



Định nghĩa theme cho ứng dụng Material

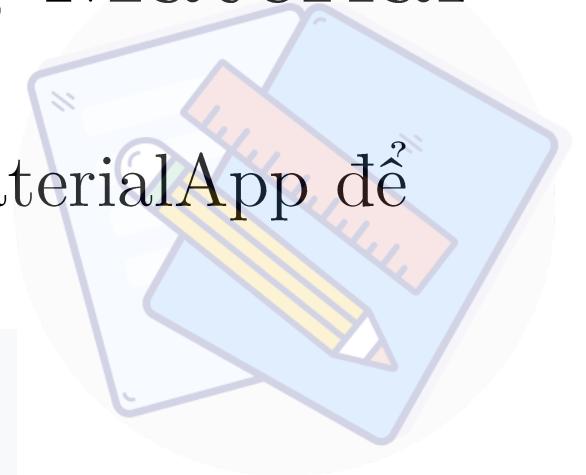


- Cần cung cấp một đối tượng [ThemeData](#) cho MaterialApp để định nghĩa theme cho toàn bộ ứng dụng
 - Theme mặc định sẽ được tạo nếu không cung cấp đối tượng ThemeData
- Có thể truy xuất theme trong phương thức build của các widget với [Theme.of\(context\)](#)
- Để tùy biến theme cho *một phần UI* của ứng dụng có thể sử dụng widget [Theme](#)
 - Tạo mới một đối tượng ThemeData hoặc
 - Mở rộng đối tượng ThemeData hiện có với .copyWith()

Định nghĩa theme cho ứng dụng Material

- Cần cung cấp một đối tượng [ThemeData](#) cho MaterialApp để định nghĩa theme cho toàn bộ ứng dụng, ví dụ:

```
MaterialApp(  
    title: appName,  
    theme: ThemeData(  
        // Define the default brightness and colors.  
        brightness: Brightness.dark,  
        primaryColor: Colors.lightBlue[800],  
  
        // Define the default font family.  
        fontFamily: 'Georgia',  
  
        // Define the default 'TextTheme'. Use this to specify the default  
        // text styling for headlines, titles, bodies of text, and more.  
        textTheme: const TextTheme(  
            headline1: TextStyle(fontSize: 72.0, fontWeight: FontWeight.bold),  
            headline6: TextStyle(fontSize: 36.0, fontStyle: FontStyle.italic),  
            bodyText2: TextStyle(fontSize: 14.0, fontFamily: 'Hind'),  
        ),  
    ),  
    home: const MyHomePage(  
        title: appName,  
    ),  
);
```



Định nghĩa theme cho ứng dụng Material

- TextTheme: tạo theme cho các loại văn bản theo đặc tả của hệ thống thiết kế Material, chia làm 5 nhóm:
 - Display{Large/Medium/Small}
 - Headline{Large/Medium/Small}
 - Title{Large/Medium/Small}
 - Label{Large/Medium/Small}
 - Body{Large/Medium/Small}

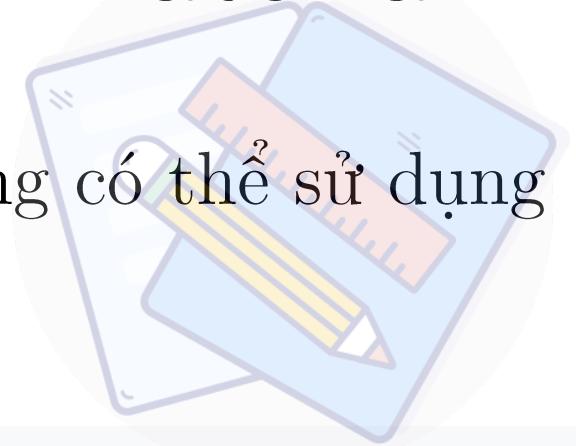


Định nghĩa theme cho ứng dụng Material

- Để tùy biến theme cho *một phần UI* của ứng dụng có thể sử dụng widget Theme, ví dụ:

```
Theme(  
  // Create a unique theme with `ThemeData`  
  data: ThemeData(  
    splashColor: Colors.yellow,  
  ),  
  child: FloatingActionButton(  
    onPressed: () {},  
    child: const Icon(Icons.add),  
  ),  
);
```

```
Theme(  
  // Find and extend the parent theme using `copyWith`. See the next  
  // section for more info on `Theme.of`.  
  data: Theme.of(context).copyWith(splashColor: Colors.yellow),  
  child: const FloatingActionButton(  
    onPressed: null,  
    child: Icon(Icons.add),  
  ),  
);
```



Định nghĩa theme cho Fooderlich

pubspec.yaml

```
dependencies:  
  flutter:  
    sdk: flutter  
  cupertino_icons: ^1.0.2  
  google_fonts: ^2.3.2
```

Thêm gói thư viện [google_font](#) vào dự án



```
// lib/fooderlich_theme.dart  
import 'package:flutter/material.dart';  
import 'package:google_fonts/google_fonts.dart';  
  
class FooderlichTheme {  
  static TextTheme lightTextTheme = TextTheme(  
    bodyText1: GoogleFonts.openSans(  
      fontSize: 14.0,  
      fontWeight: FontWeight.w700,  
      color: Colors.black,  
    ),  
    ...  
  );  
  
  static TextTheme darkTextTheme = TextTheme(  
    ...  
    headline1: GoogleFonts.openSans(  
      fontSize: 32.0,  
      fontWeight: FontWeight.bold,  
      color: Colors.white,  
    ),  
    ...  
  );  
  
  static ThemeData light() {
```

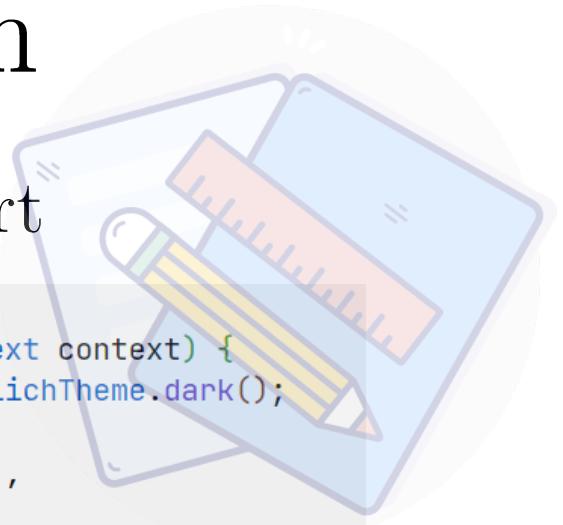
Định nghĩa theme cho Foderlich

lib/fooderlich_theme.dart

```
static ThemeData light() {  
    return ThemeData(  
        brightness: Brightness.light,  
        ...  
        textTheme: lightTextTheme,  
    );  
}  
  
static ThemeData dark() {  
    return ThemeData(  
        brightness: Brightness.dark,  
        ...  
        textTheme: darkTextTheme,  
    );  
}
```

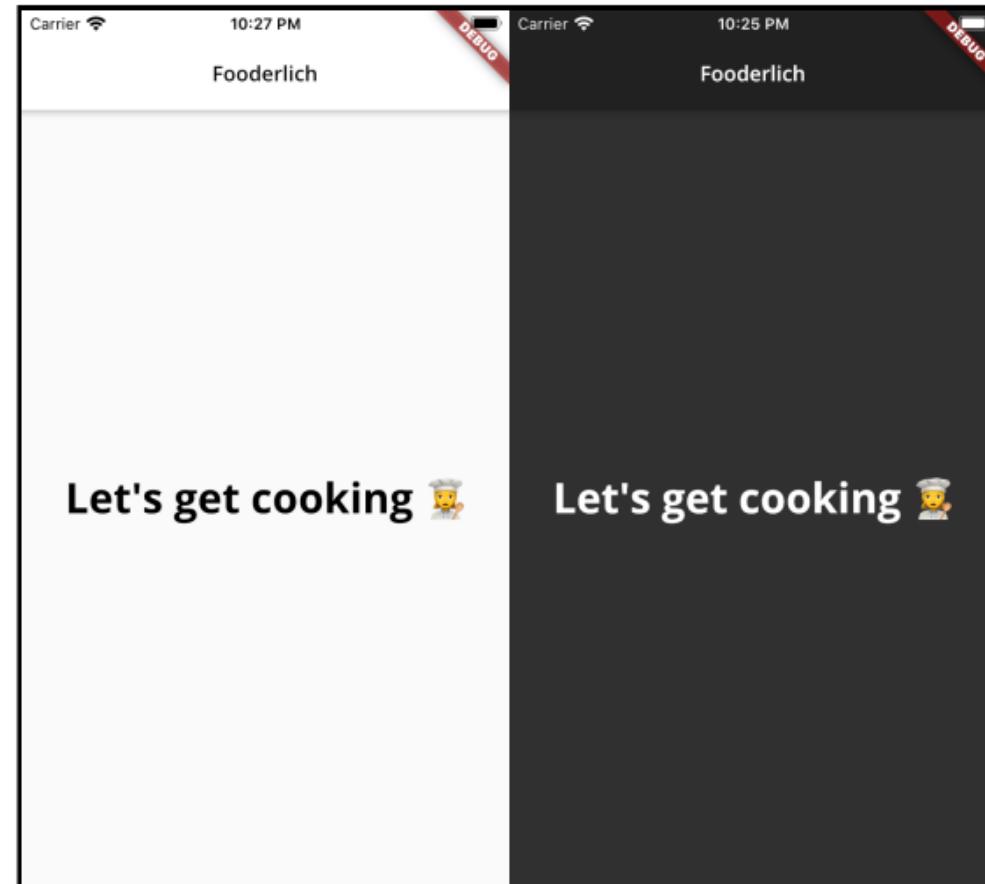
lib/main.dart

```
@override  
Widget build(BuildContext context) {  
    final theme = FoderlichTheme.dark();  
    return MaterialApp(  
        title: 'Foderlich',  
        theme: theme,  
        home: Scaffold(  
            appBar: AppBar(  
                title: Text(  
                    'Foderlich',  
                    style: theme.textTheme.headline6,  
                ), // Text  
            ), // AppBar  
            body: Center(  
                child: Text(  
                    'Let\'s get cooking 🎉',  
                    style: theme.textTheme.headline1,  
                ), // Text  
            ), // Center  
        ), // Scaffold  
    ); // MaterialApp
```



Định nghĩa theme cho Fooderlich

FooderlichTheme.light()



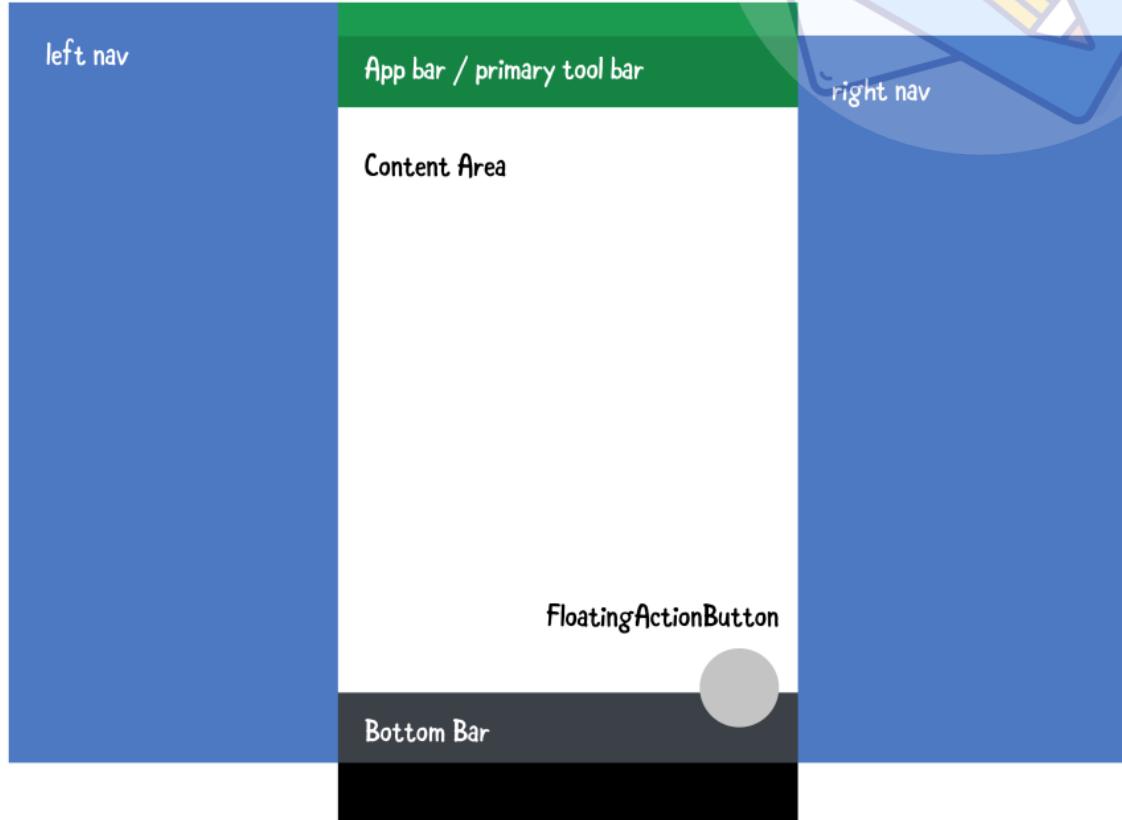
FooderlichTheme.dark()



Scaffold

- **Scaffold** cài đặt cấu trúc bố cục trực quan cơ bản của hệ thống thiết kế material. Bao gồm các thành phần sau:

- [AppBar](#)
- [BottomSheet](#)
- [\(Bottom\)NavigationBar](#)
- [Drawer](#)
- [FloatingActionButton](#)
- [SnackBar](#)



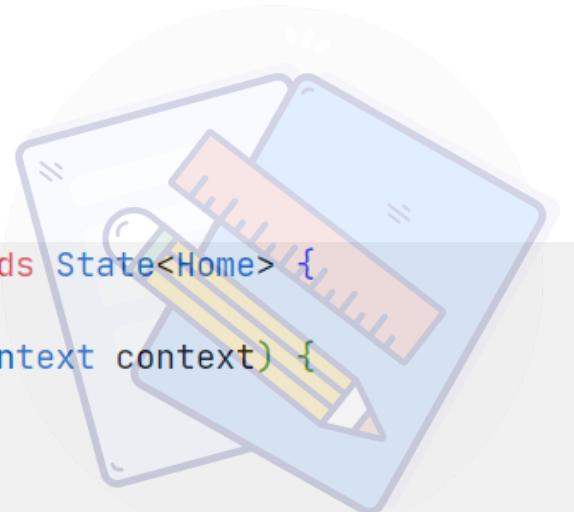
Scaffold

Tạo trang home

- Giữ cho cây widget ngắn gọn -> nên xem xét đặt các widget tùy biến trong các phương thức `_buildXXX()` hoặc tạo lớp widget riêng
- Do cần xử lý thay đổi trạng thái nên sẽ dùng StatelessWidget
- `Theme.of(context)` trả về theme được định nghĩa gần nhất trong cây widget. Trong trường hợp này là theme của MaterialApp

lib/home.dart

```
class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          'Fooderlich',
          style: Theme.of(context).textTheme.headline6,
        ), // Text
      ), // AppBar
      body: Center(
        child: Text(
          'Let\'s get cooking 🎉',
          style: Theme.of(context).textTheme.headline1,
        ), // Text
      ), // Center
    ); // Scaffold
}
```



Scaffold

Tạo trang home

- Import và sử dụng widget Home đã tạo trong main.dart

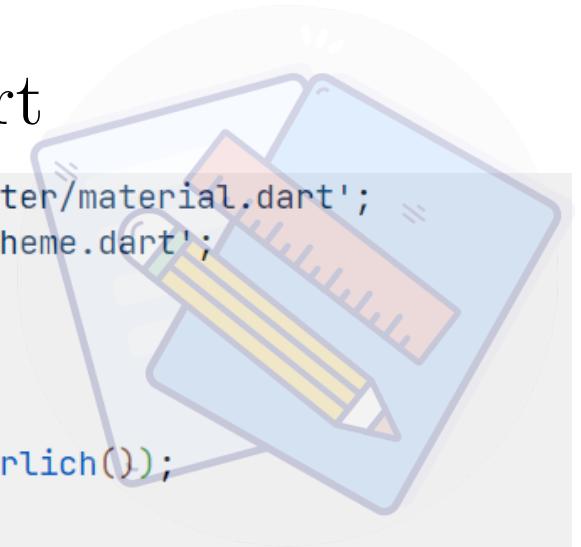
lib/main.dart

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

Run | Debug | Profile
void main() {
  runApp(const Foderlich());
}

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

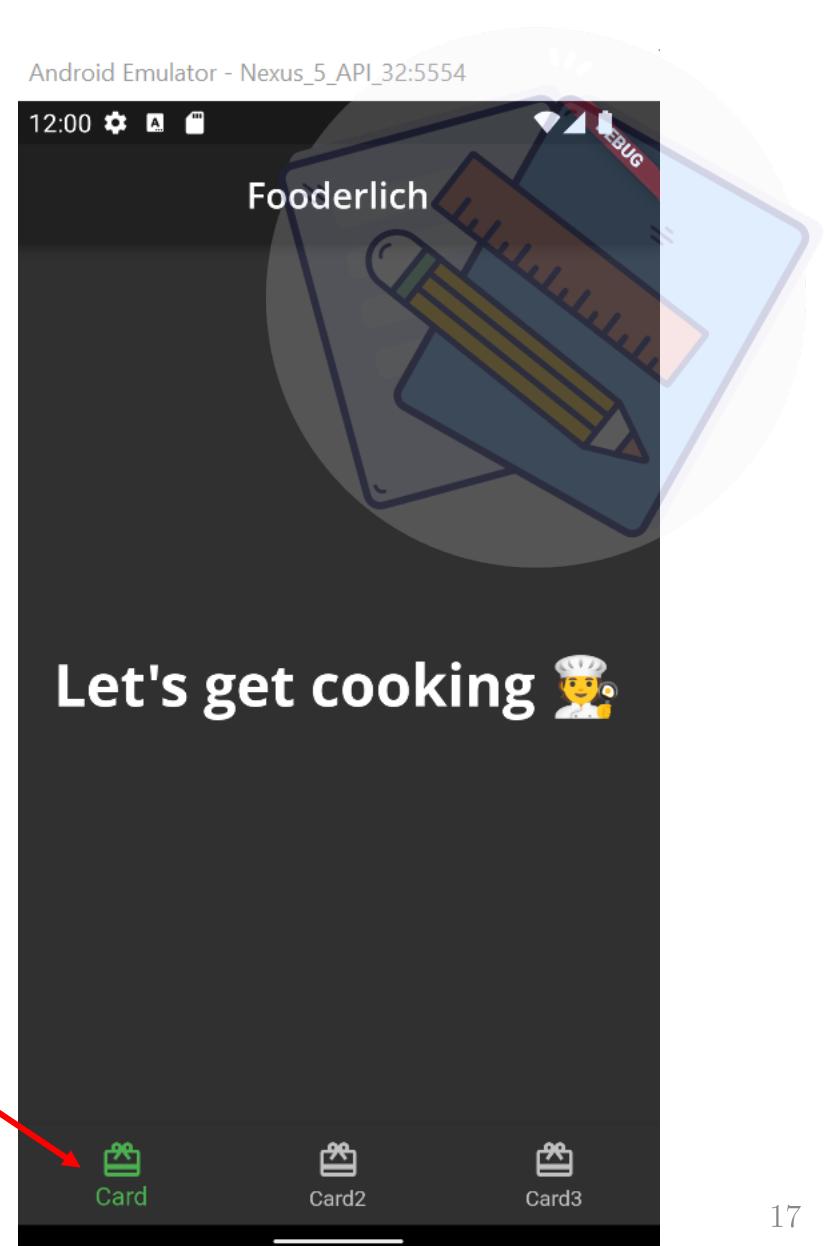
  @override
  Widget build(BuildContext context) {
    final theme = FoderlichTheme.dark();
    return MaterialApp(
      title: 'Foderlich',
      theme: theme,
      home: const Home(),
    ); // MaterialApp
}
```



Scaffold

Tạo thanh điều hướng dưới

```
body: Center(
  child: Text(
    'Let\'s get cooking 🍳',
    style: Theme.of(context).textTheme.headline1,
  ), // Text
), // Center
bottomNavigationBar: BottomNavigationBar(
  selectedTextColor: Theme.of(context).textSelectionTheme.selectionColor,
  items: const <BottomNavigationBarItem>[
    BottomNavigationBarItem(
      icon: Icon(Icons.card_giftcard),
      label: 'Card',
    ), // BottomNavigationBarItem
    BottomNavigationBarItem(
      icon: Icon(Icons.card_giftcard),
      label: 'Card2',
    ), // BottomNavigationBarItem
    BottomNavigationBarItem(
      icon: Icon(Icons.card_giftcard),
      label: 'Card3',
    ), // BottomNavigationBarItem
  ], // <BottomNavigationBarItem>[]
), // BottomNavigationBar
```



Scaffold

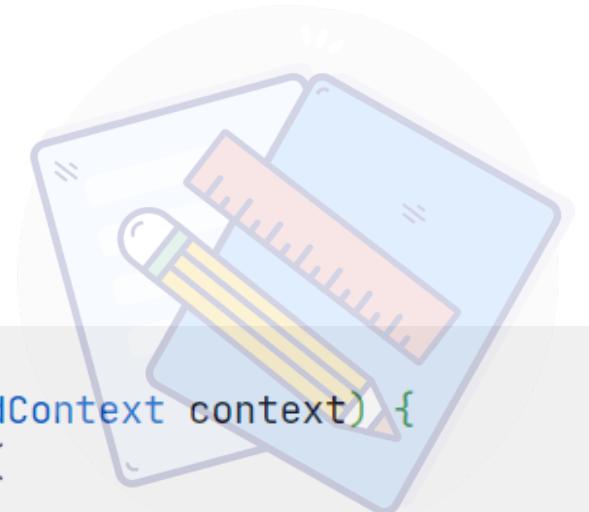
Cài đặt điều hướng giữa các thẻ

```
class _HomeState extends State<Home> {
    int _selectedIndex = 0;

    static final List<Widget> _pages = <Widget>[
        Container(color: Colors.red),
        Container(color: Colors.green),
        Container(color: Colors.blue),
    ]; // <Widget>[]

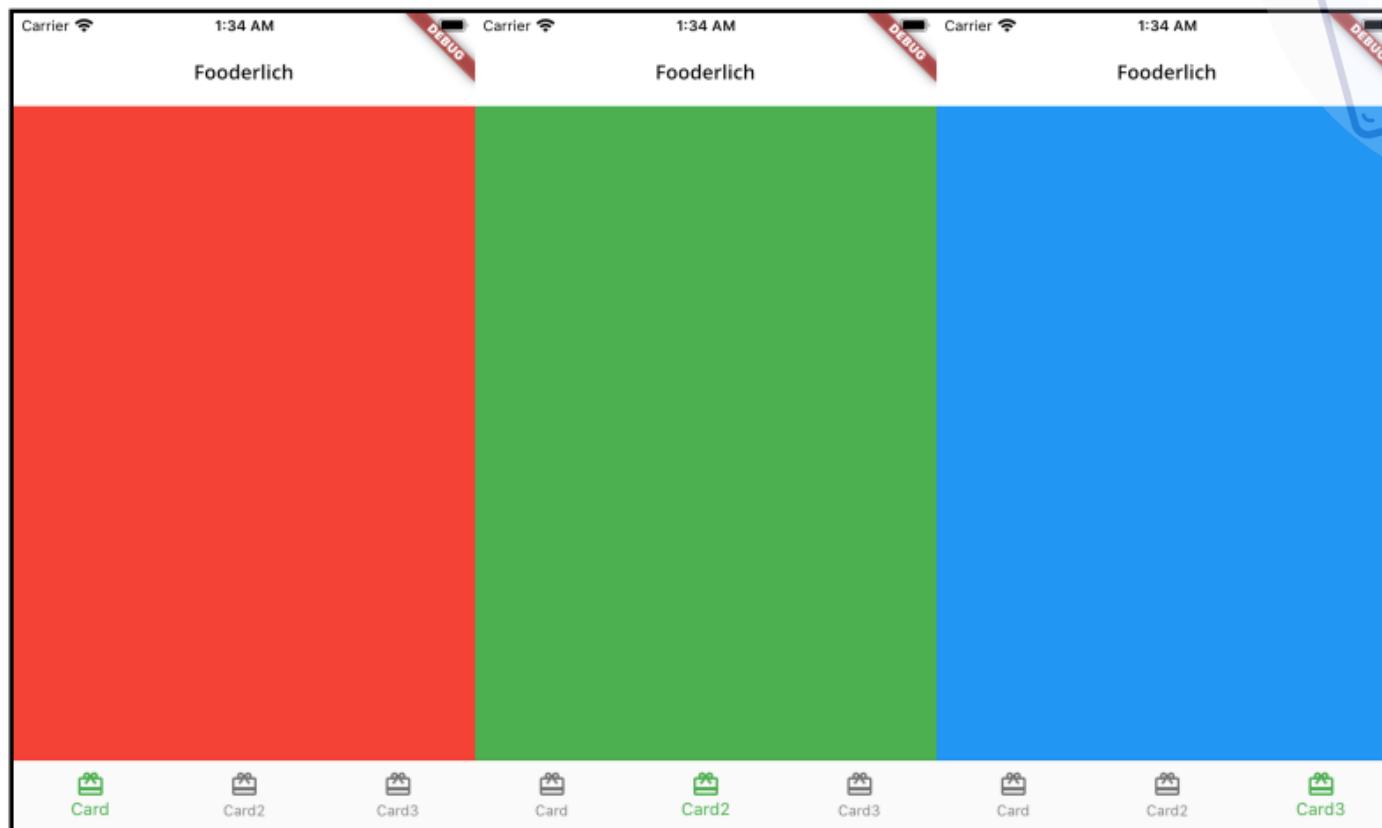
    void _onItemTapped(int index) {
        setState(() {
            _selectedIndex = index;
        });
    }
}
```

```
@override
Widget build(BuildContext context) {
    return Scaffold(
        ...
        body: _pages[_selectedIndex],
        bottomNavigationBar: BottomNavigationBar(
            ...
            currentIndex: _selectedIndex,
            onTap: _onItemTapped,
        ),
    );
}
```



Scaffold

Cài đặt điều hướng giữa các thẻ



Các widget hiển thị và bố cục

Các widget hiển thị: những gì người dùng thấy trên màn hình

- Text, TextSpan, RichText
- Image, Icon
- ElevatedButton, TextButton, IconButton, FloatingActionButton

Các widget bố cục: tổ chức, sắp xếp các widget con

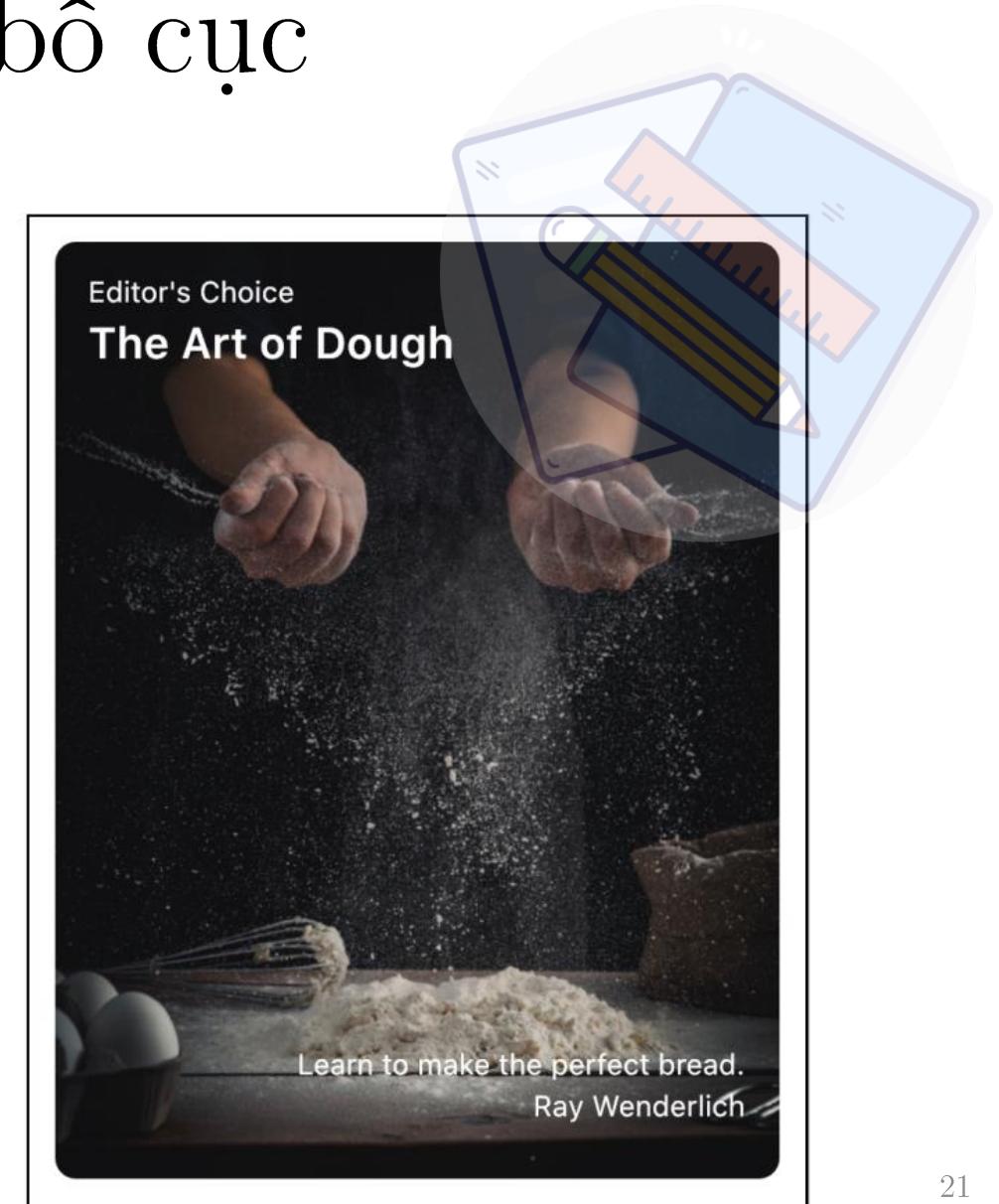
- Container, Padding, SizedBox
- Column, Row, Flexible, Expanded
- Stack, IndexedStack
- ListView, GridView
- Card, ListTile, GridTile



Các widget hiển thị và bố cục

Card1 sử dụng các widget sau:

- Container
- Stack, Positioned
- Text, Image



Các widget hiển thị và bố cục

Xây dựng Card1

```
// lib/card1.dart
import 'package:flutter/material.dart';

class Card1 extends StatelessWidget {
  const Card1({Key? key}) : super(key: key);

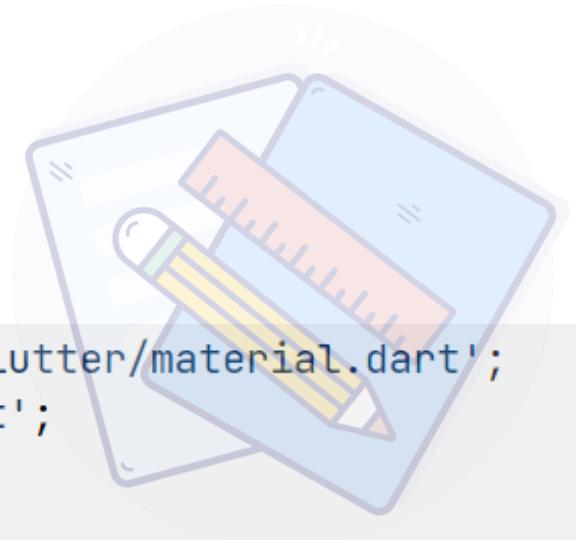
  final String category = 'Editor\'s Choice';
  final String title = 'The Art of Dough';
  final String description = 'Learn to make the perfect bread.';
  final String chef = 'Ray Wenderlich';

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(),
    ); // Center
  }
}
```

```
import 'package:flutter/material.dart';
import 'card1.dart';

...
class _HomeState extends State<Home> {
  int _selectedIndex = 0;

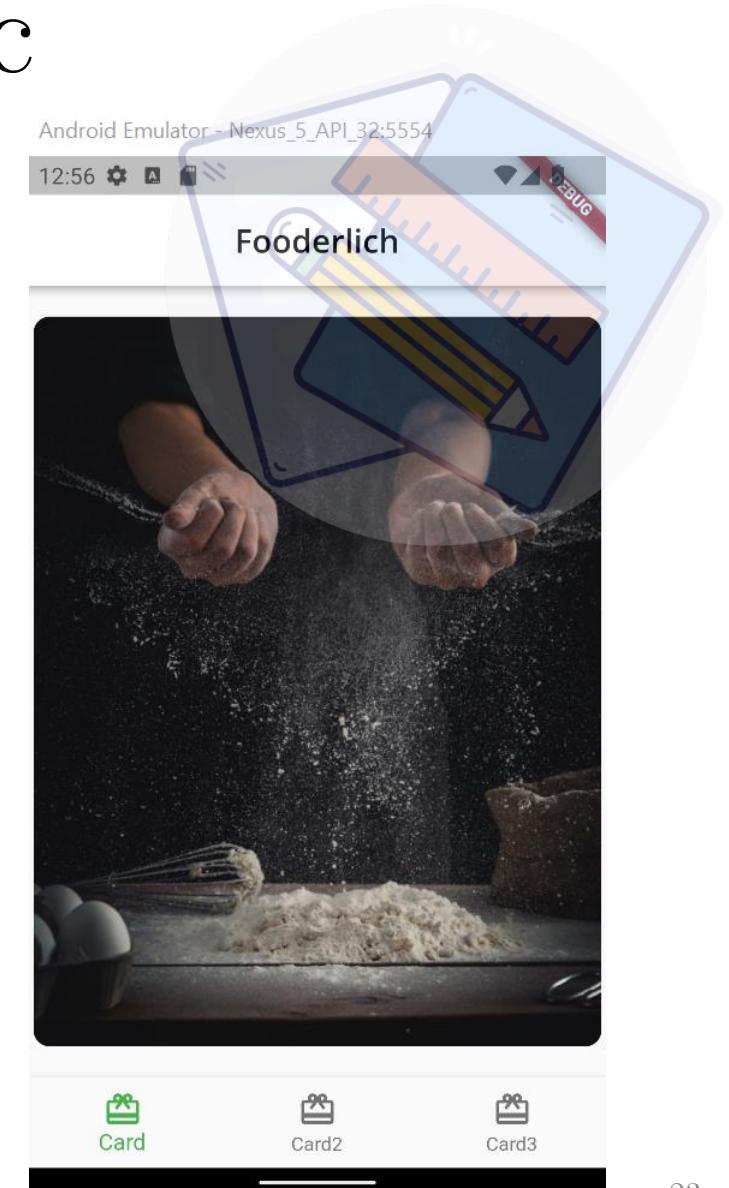
  static final List<Widget> _pages = <Widget>[
    const Card1(),
    Container(color: Colors.green),
    Container(color: Colors.blue),
  ];
  ...
}
```



Các widget hiển thị và bố cục

Xây dựng Card1

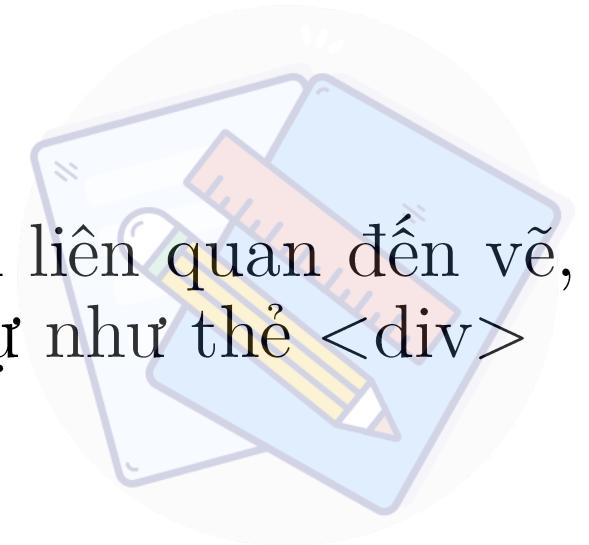
```
@override  
Widget build(BuildContext context) {  
  return Center(  
    child: Container(  
      padding: const EdgeInsets.all(16),  
      constraints: const BoxConstraints.expand(  
        width: 350,  
        height: 450,  
      ), // BoxConstraints.expand  
      decoration: const BoxDecoration(  
        image: DecorationImage(  
          image: AssetImage('assets/mag1.png'),  
          fit: BoxFit.cover,  
        ), // DecorationImage  
        borderRadius: BorderRadius.all(Radius.circular(10.0)),  
      ), // BoxDecoration  
    ), // Container  
  ); // Center  
}
```



Các widget hiển thị và bố cục

Container: widget cung cấp các thuộc tính chuẩn liên quan đến vẽ, định vị trí và định kích thước các widget (tương tự như thẻ `<div>` trong HTML)

- **alignment:** quy định cách căn chỉnh phần tử con
- **width/height:** tùy chỉnh rộng/cao
- **margin/padding:** tùy chỉnh margin/padding
- **constraints:** ràng buộc kích thước cho container
- **decoration:** được vẽ phía sau widget con của container (ví dụ có thể sử dụng [BoxDecoration](#) để miêu tả cách tùy chỉnh bóng đổ và bo tròn các góc)
- **transform:** ma trận biến đổi container (ví dụ như xoay)



Các widget hiển thị và bố cục

Container với các widget cắt: [ClipRRect](#) và [ClipOval](#)

Widget	Result
<pre>Container(width: 70, height: 70, color: Colors.blue,)</pre>	
<pre>ClipRRect(borderRadius: BorderRadius.all(Radius.circular(10),<br),<br=""/> child: Container(width: 70, height: 70, color: Colors.blue,<br),<br=""/>,</pre>	
<pre>ClipOval(child: Container(width: 70, height: 70, color: Colors.blue, ,</pre>	



Các widget hiển thị và bố cục

Xây dựng Card1: thêm text

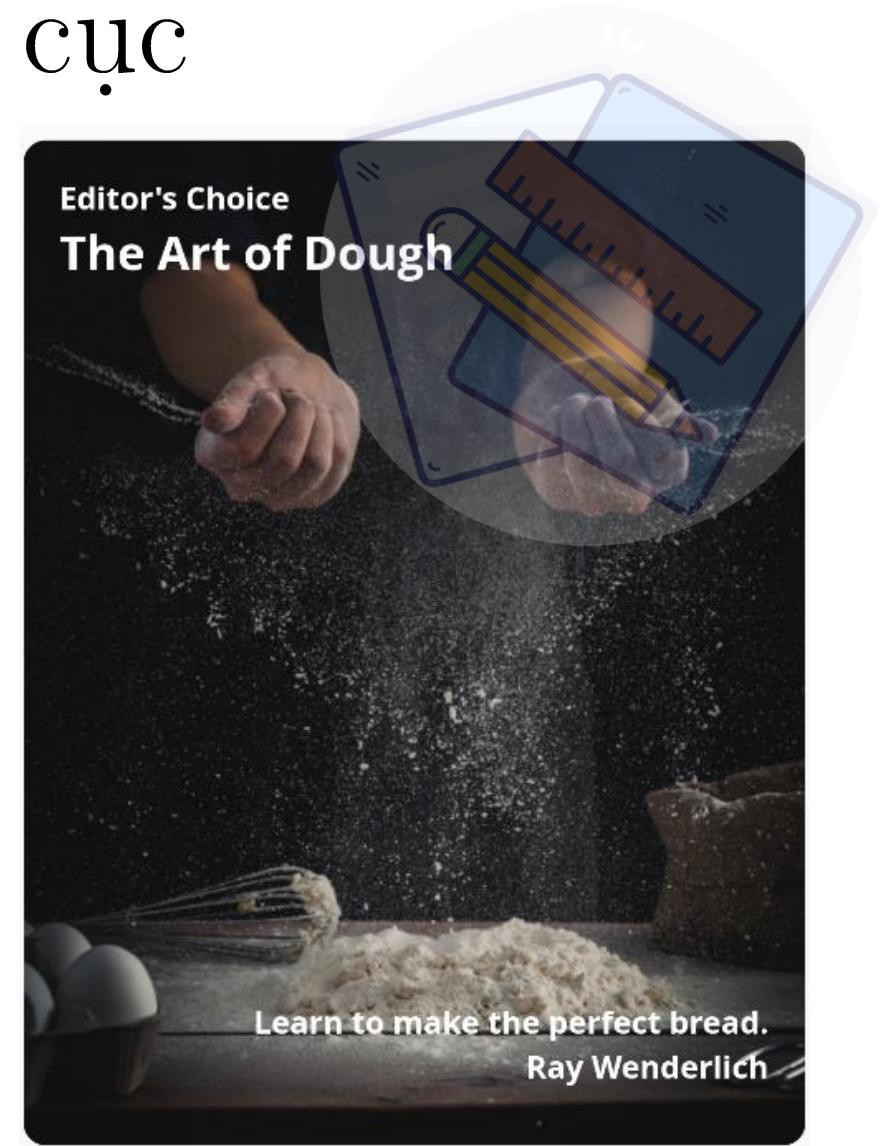
```
child: Container(  
  ...  
  child: Stack(  
    children: [  import 'fooderlich_theme.dart';  
      Text(  
        category,  
        style: FoederlichTheme.darkTextTheme.bodyText1,  
      ),  
      Text(  
        title,  
        style: FoederlichTheme.darkTextTheme.headline5,  
      ),  
      Text(  
        description,  
        style: FoederlichTheme.darkTextTheme.bodyText1,  
      ),  
      Text(  
        chef,  
        style: FoederlichTheme.darkTextTheme.bodyText1,  
      ),  
    ],  
  ),  
,
```



Các widget hiển thị và bố cục

Xây dựng Card1: định vị các text

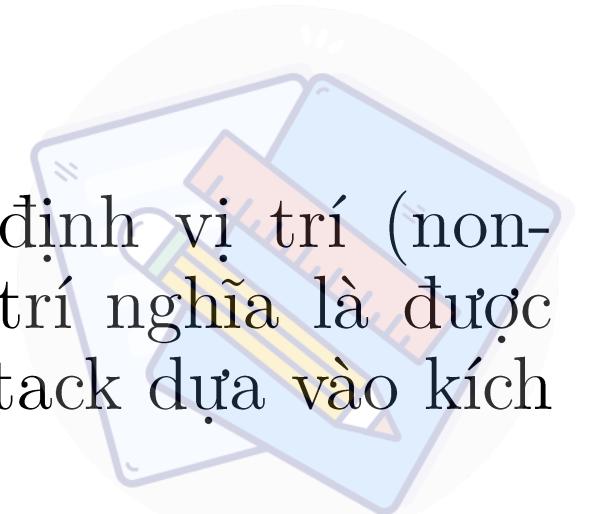
```
child: Stack(  
  children: [  
    Text(category, style: FoderlichTheme.darkTextTheme.bodyText1),  
    Positioned(  
      top: 20,  
      child:  
        Text(title, style: FoderlichTheme.darkTextTheme.headline2),  
    ), // Positioned  
    Positioned(  
      bottom: 30,  
      right: 0,  
      child: Text(description,  
        style: FoderlichTheme.darkTextTheme.bodyText1), // Text  
    ), // Positioned  
    Positioned(  
      bottom: 10,  
      right: 0,  
      child: Text(chef, style: FoderlichTheme.darkTextTheme.bodyText1),  
    ), // Positioned  
  ],  
, // Stack
```



Các widget hiển thị và bố cục

Stack: mỗi con trong stack có thể không được định vị trí (non-positioned) hoặc được định vị trí. Được định vị trí nghĩa là được bao bọc trong một **Positioned**. Kích thước của stack dựa vào kích thước của con non-positioned

- **alignment:** căn chỉnh các widget con (nếu chưa được cố định vị trí, mặc định AlignmentDirectional.topStart)
- **fit:** định kích thước các widget con non-positioned (mặc định StackFit.loose)



Các widget hiển thị và bố cục

Positioned: chỉ định vị trí theo trực đứng và trực ngang của một con trong Stack

- **top/bottom/height:** định vị trí theo trực đứng. Chỉ hai thuộc tính được sử dụng đồng thời
- **left/right/width:** định vị trí theo trực ngang. Chỉ hai thuộc tính được sử dụng đồng thời



Các widget hiển thị và bố cục

Card2 sử dụng các widget sau:

- Container
- Stack, Positioned
- Row, Column, Expanded
- RotatedBox
- Text, Image, CircleAvatar, IconButton



Các widget hiển thị và bố cục

Xây dựng Card2



AuthorCard



AuthorCard

Các widget hiển thị và bố cục

Xây dựng Card2

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

class Card2 extends StatelessWidget {
  const Card2({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        constraints: const BoxConstraints.expand(width: 350, height: 450),
        decoration: const BoxDecoration(
          image: DecorationImage(
            image: AssetImage('assets/mag5.png'),
            fit: BoxFit.cover,
          ), // DecorationImage
          borderRadius: BorderRadius.all(Radius.circular(10.0)),
        ), // BoxDecoration
        child: Column(
          children: const [],
        ), // Column
      ), // Container
    ); // Center
  }
}
```



Các widget hiển thị và bố cục

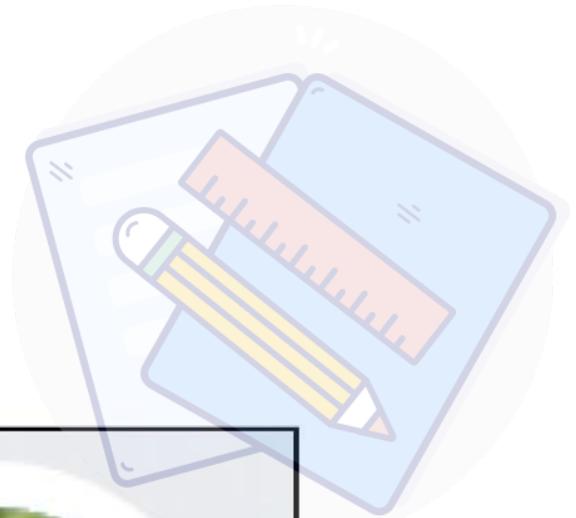
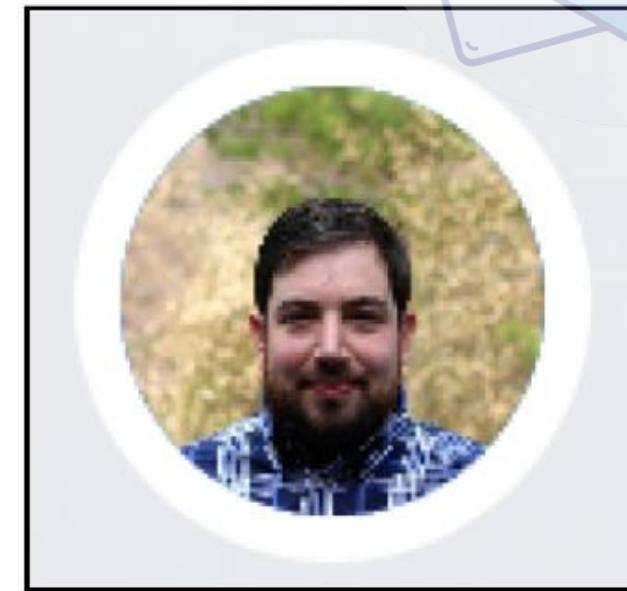
Xây dựng CircleImage

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

class CircleImage extends StatelessWidget {
  const CircleImage({
    Key? key,
    this.imageProvider,
    this.imageRadius = 20,
  }) : super(key: key);

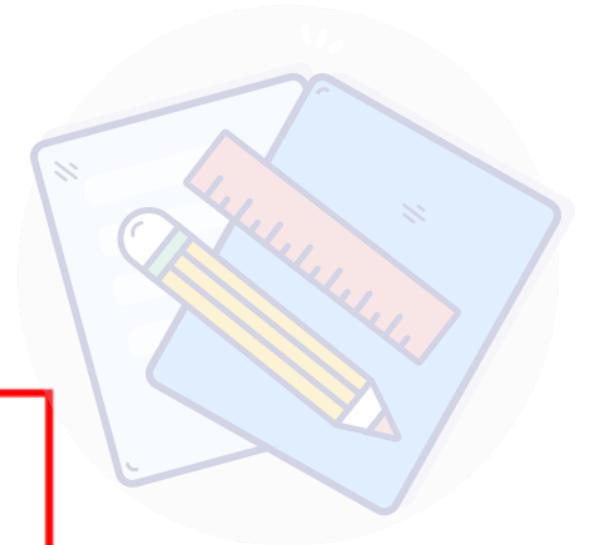
  final double imageRadius;
  final ImageProvider? imageProvider;

  @override
  Widget build(BuildContext context) {
    return CircleAvatar(
      backgroundColor: Colors.white,
      radius: imageRadius,
      child: CircleAvatar(
        radius: imageRadius - 5,
        backgroundImage: imageProvider,
      ), // CircleAvatar
    ); // CircleAvatar
  }
}
```



Các widget hiển thị và bố cục

Xây dựng AuthorCard



AuthorCard

Các widget hiển thị và bố cục

Xây dựng AuthorCard

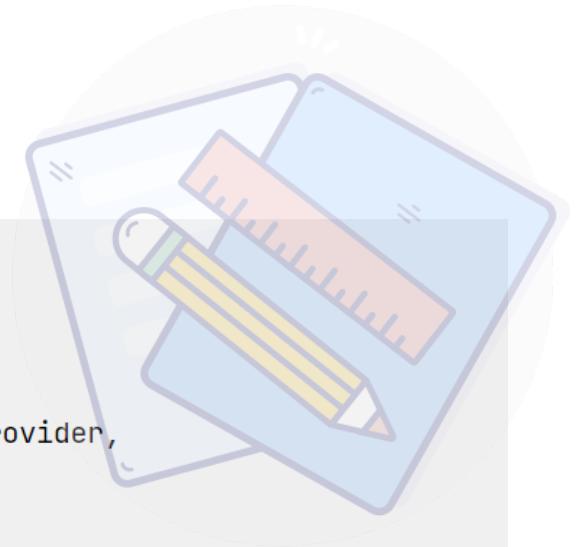
```
import 'package:flutter/material.dart';
import 'fooderlich_theme.dart';
import 'circle_image.dart';

class AuthorCard extends StatelessWidget {
  final String authorName;
  final String title;
  final ImageProvider? imageProvider;

  const AuthorCard({
    Key? key,
    required this.authorName,
    required this.title,
    this.imageProvider,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: const EdgeInsets.all(16),
      child: Row(
        children: [],
      ),
    );
  }
}
```

```
child: Row(
  children: [
    Row(
      children: [
        CircleImage(
          imageProvider: imageProvider,
          imageRadius: 28,
        ), // CircleImage
        const SizedBox(width: 8),
        Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              authorName,
              style: FoderlichTheme.lightTextTheme.headline2,
            ), // Text
            Text(
              title,
              style: FoderlichTheme.lightTextTheme.headline3,
            ), // Text
          ],
        ), // Column
      ],
    ), // Row
  ],
), // Row
```



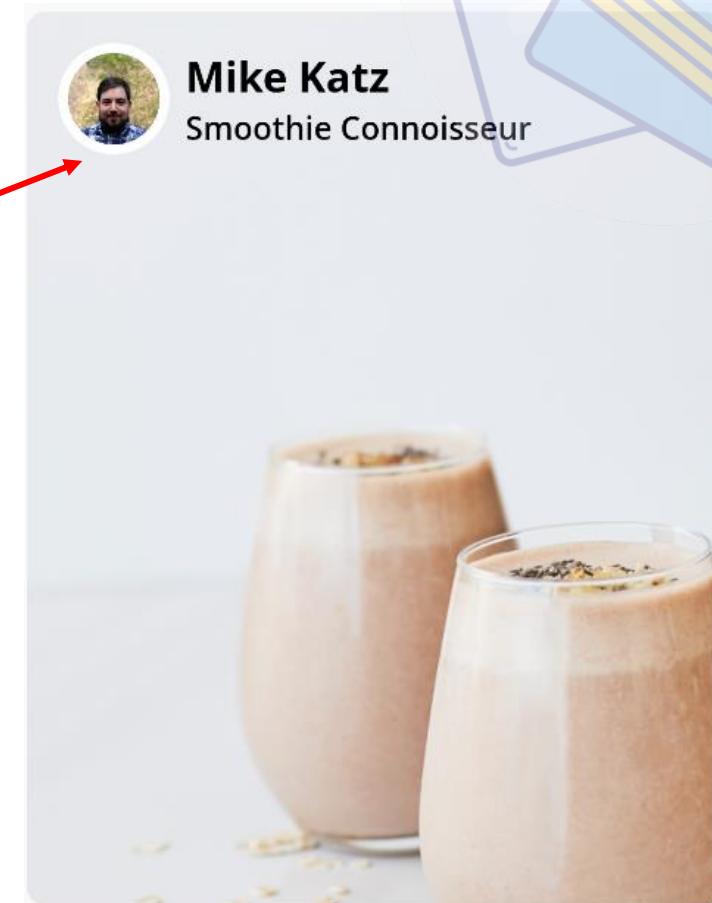
Các widget hiển thị và bố cục

Xây dựng Card2: thêm AuthorCard

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

class Card2 extends StatelessWidget {
  const Card2({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        ...
        child: Column(
          children: const <Widget>[
            AuthorCard(
              authorName: 'Mike Katz',
              title: 'Smoothie Connoisseur',
              imageProvider: AssetImage('assets/author_katz.jpeg'),
            ),
          ],
        ),
      );
  }
}
```



Các widget hiển thị và bố cục

Xây dựng AuthorCard: thêm IconButton và Snackbar

```
class AuthorCard extends StatelessWidget {  
  ...  
  @override  
  Widget build(BuildContext context) {  
    return Container(  
      padding: const EdgeInsets.all(16),  
      child: Row(  
        children: [  
          ...  
          IconButton(  
            icon: const Icon(Icons.favorite_border),  
            iconSize: 30,  
            color: Colors.grey[400],  
            onPressed: () {  
              const snackBar = SnackBar(content: Text('Favorite Pressed'));  
              ScaffoldMessenger.of(context).showSnackBar(snackBar);  
            },  
          ),  
        ],  
      );  
  }  
}
```



Các widget hiển thị và bố cục

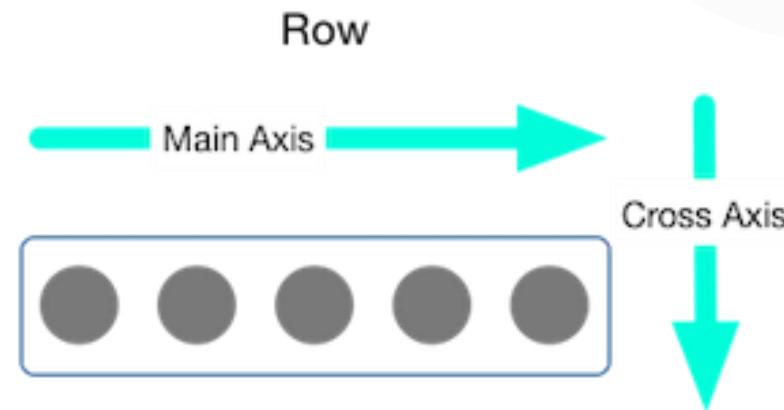
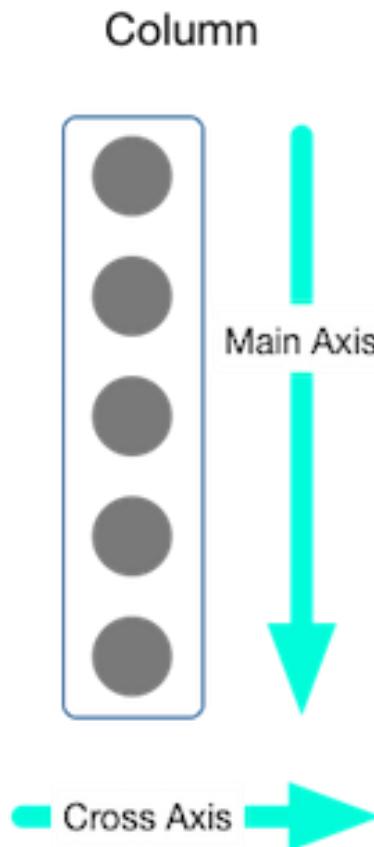
Xây dựng AuthorCard: điều chỉnh khoảng cách

```
return Container(  
  padding: const EdgeInsets.all(16),  
  child: Row(  
    mainAxisAlignment: MainAxisAlignment.spaceBetween,  
    children: [  
      ...  
      IconButton(  
        icon: const Icon(Icons.favorite_border),  
        iconSize: 30,  
        color: Colors.grey[400],  
        onPressed: () {  
          const snackBar = SnackBar(content: Text('Favorite Pressed'));  
          ScaffoldMessenger.of(context).showSnackBar(snackBar);  
        },  
      ),  
    ],  
  ),  
);
```



Các widget hiển thị và bố cục

Các trục của widget Column và Row



Các widget hiển thị và bố cục

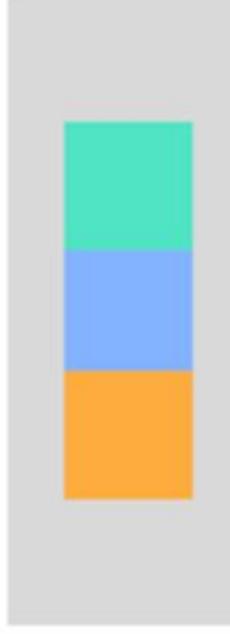
Sắp xếp các phần tử con trong Column: theo trục chính
(MainAxisAlignment)



.center	.start	.end	.spaceEvenly	.spaceAround	.spaceBetween
					

Các widget hiển thị và bố cục

Sắp xếp các phần tử con trong Column: theo trục cắt (CrossAxisAlignment)

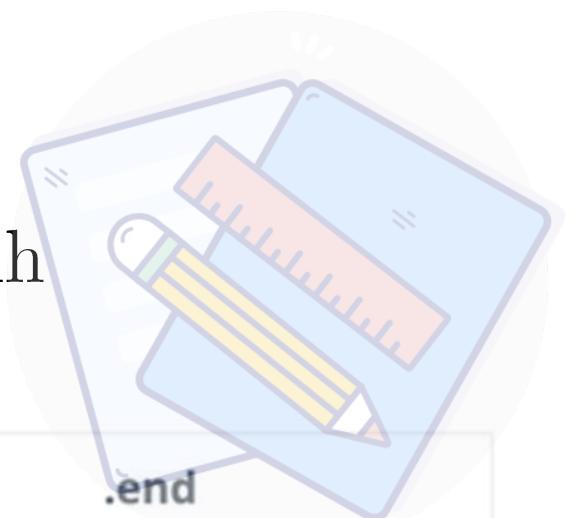
.center	.start	.end	.stretch
			



Các widget hiển thị và bố cục

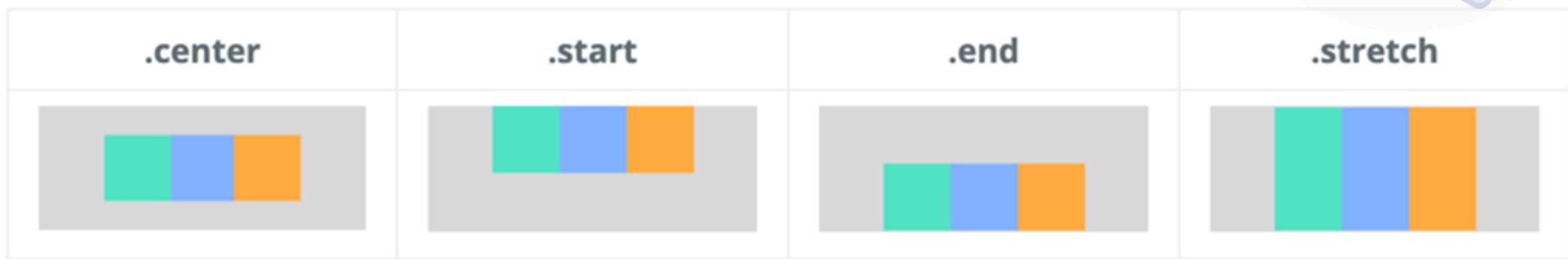
Sắp xếp các phần tử con trong Row: theo trục chính
(MainAxisAlignment)

<code>.center</code>	<code>.start</code>	<code>.end</code>
		
<code>.spaceEvenly</code>	<code>.spaceAround</code>	<code>.spaceBetween</code>
		



Các widget hiển thị và bố cục

Sắp xếp các phần tử con trong Row: theo trục cắt (CrossAxisAlignment)



Các widget hiển thị và bố cục

Xây dựng Card2: thêm text

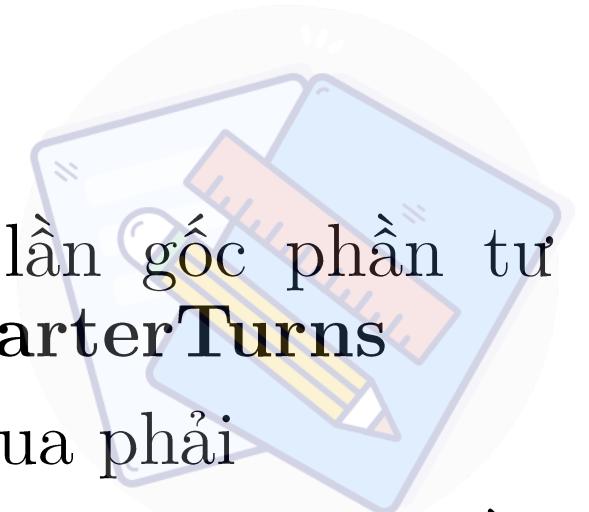
```
), // AuthorCard
Expanded(
  child: Stack(
    children: [
      Positioned(
        bottom: 16,
        right: 16,
        child: Text(
          'Recipe',
          style: FoderlichTheme.lightTextTheme.headline1,
        ), // Text
      ), // Positioned
      Positioned(
        bottom: 70,
        left: 16,
        child: RotatedBox(
          quarterTurns: 3,
          child: Text(
            'Smoothies',
            style: FoderlichTheme.lightTextTheme.headline1,
          ), // Text
        ), // RotatedBox
      ), // Positioned
    ],
  ), // Stack
), // Expanded
```



Các widget hiển thị và bố cục

RotatedBox: xoay widget con một số nguyên lần gốc phần tư vòng tròn (90 độ), chỉ định thông qua tham số **quarterTurns**

- Đối với Text, vị trí ban đầu nằm ngang từ trái qua phải
- Giá trị **quarterTurns** là số nguyên dương thì xoay cùng chiều kim đồng hồ, là số nguyên âm thì xoay ngược chiều kim đồng hồ



Các widget hiển thị và bố cục

Card3 sử dụng các widget sau:

- Container, [SizedBox](#)
- Stack
- Column
- [Wrap](#)
- Icon, Image, Text
- [Chip](#)



Các widget hiển thị và bố cục

Xây dựng Card3

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

class Card3 extends StatelessWidget {
  const Card3({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        constraints: const BoxConstraints.expand(
          width: 350,
          height: 450,
        ), // BoxConstraints.expand
        decoration: const BoxDecoration(
          image: DecorationImage(
            image: AssetImage('assets/mag2.png'),
            fit: BoxFit.cover,
          ), // DecorationImage
          borderRadius: BorderRadius.all(Radius.circular(10.0)),
        ), // BoxDecoration
        child: Stack(
          children: const [],
        ), // Stack
      ), // Container
    ); // Center
  }
}
```



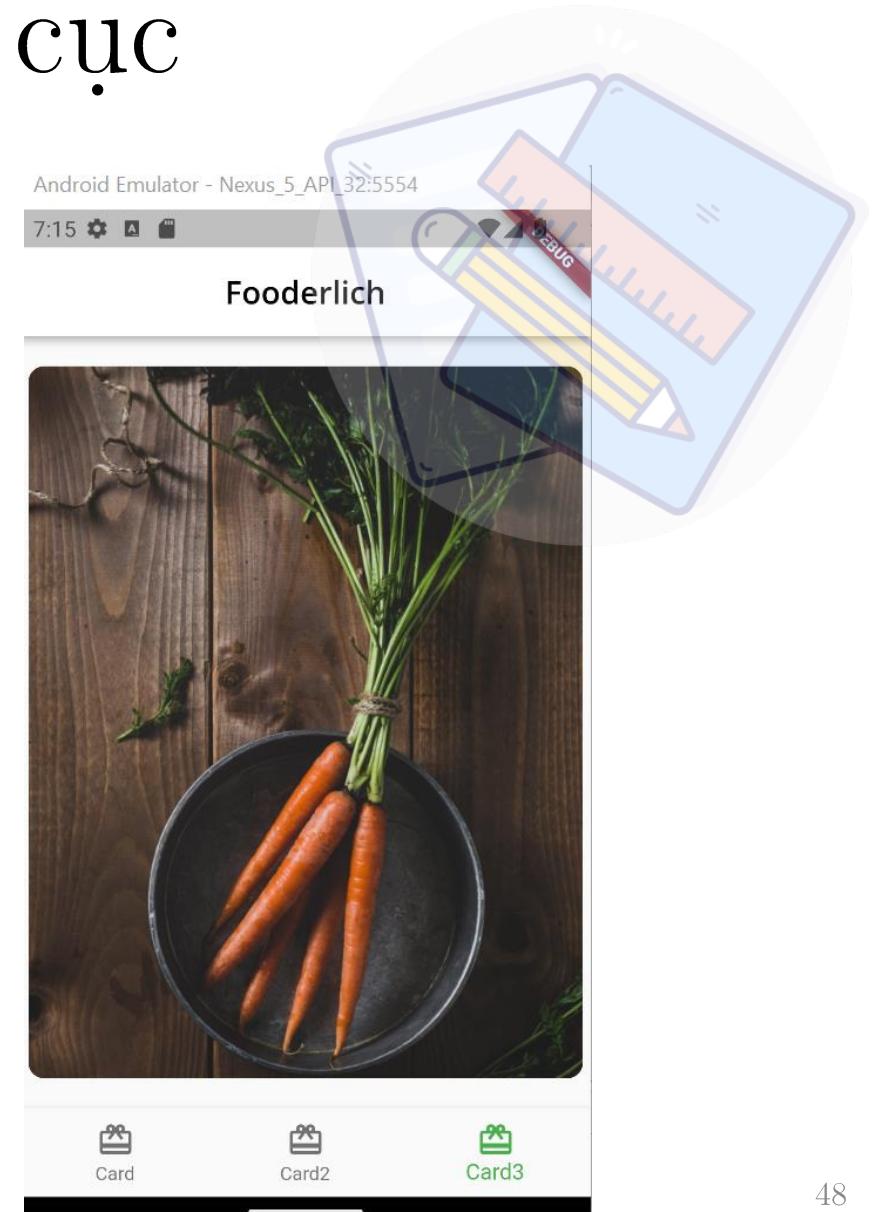
Các widget hiển thị và bố cục

Xây dựng Card3

```
import 'package:flutter/material.dart';
import 'card1.dart';
import 'card2.dart';
import 'card3.dart';

class _HomeState extends State<Home> {
  int _selectedIndex = 0;

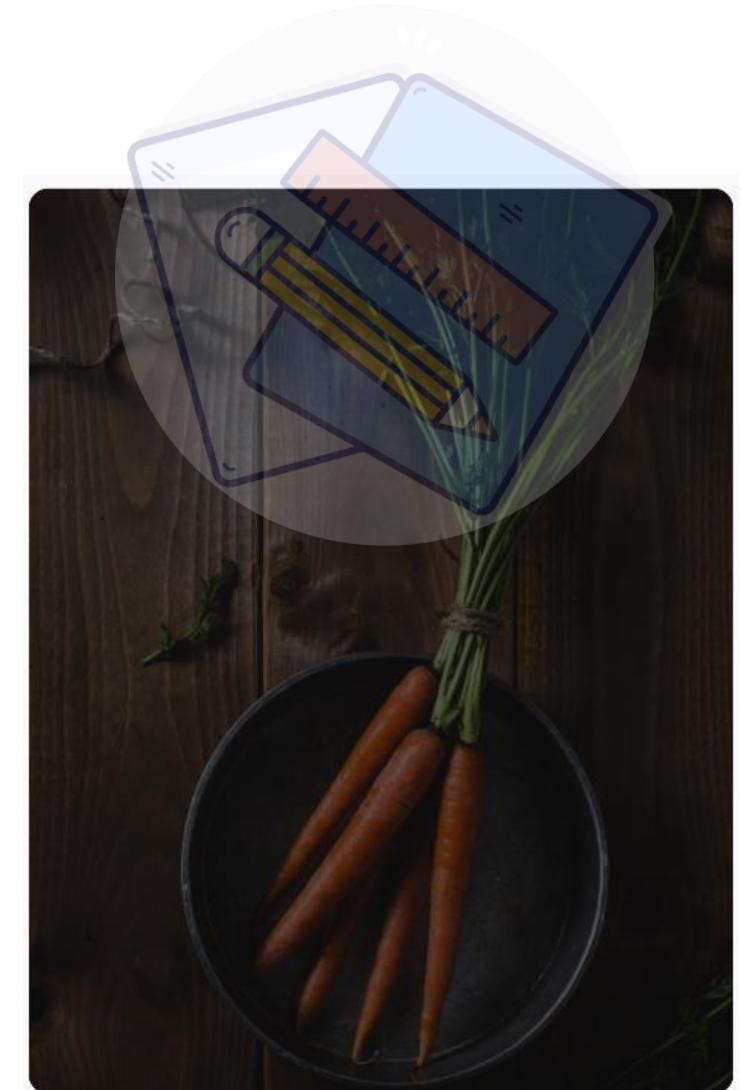
  static final List<Widget> _pages = <Widget>[
    const Card1(),
    const Card2(),
    const Card3(),
  ];
  ...
}
```



Các widget hiển thị và bố cục

Xây dựng Card3: tạo lớp phủ đen

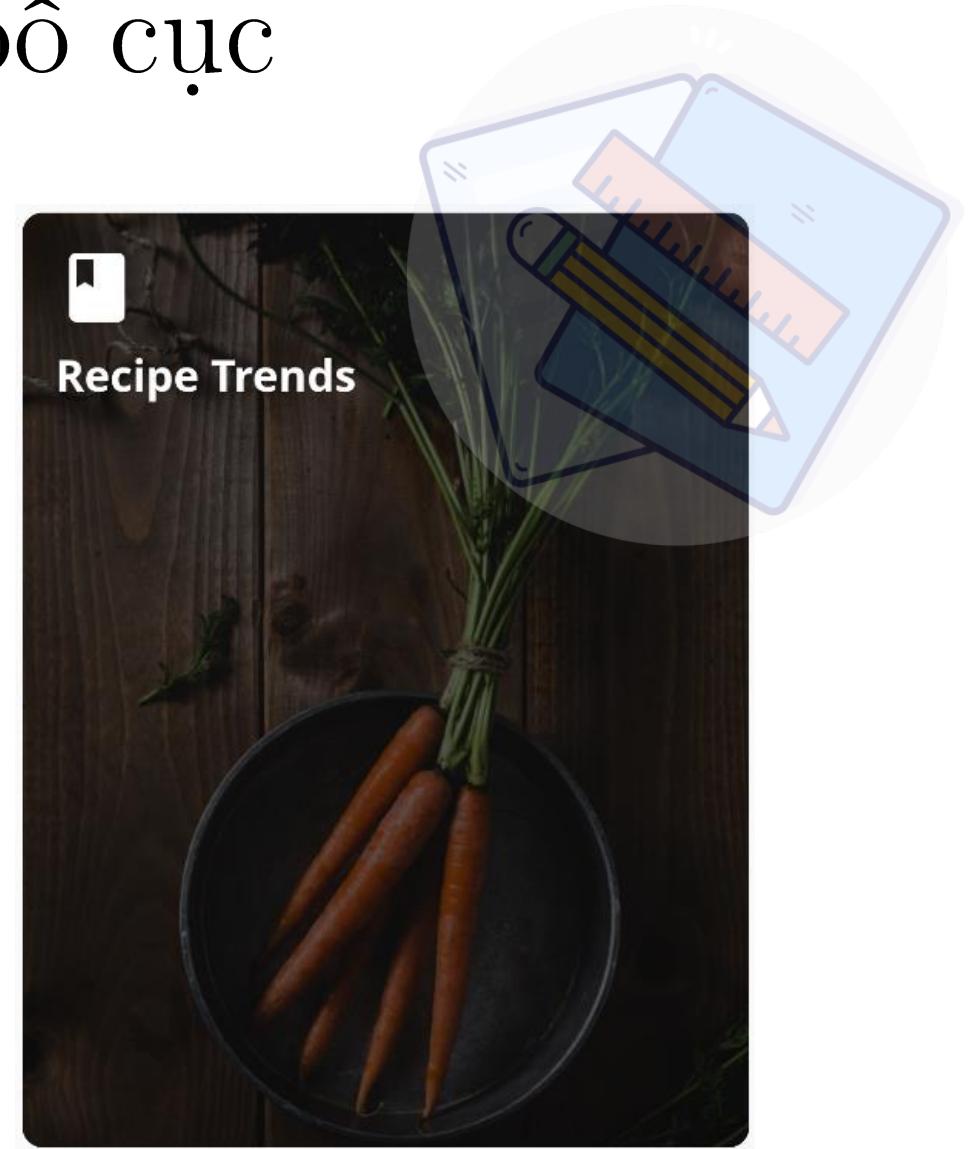
```
child: Stack(  
  children: <Widget>[  
    Container(  
      decoration: BoxDecoration(  
        color: Colors.black.withOpacity(0.6),  
        borderRadius: const BorderRadius.all(Radius.circular(10.0)),  
      ), // BoxDecoration  
    ), // Container  
  ], // <Widget>[]  
>, // Stack
```



Các widget hiển thị và bố cục

Xây dựng Card3: thêm tựa đề

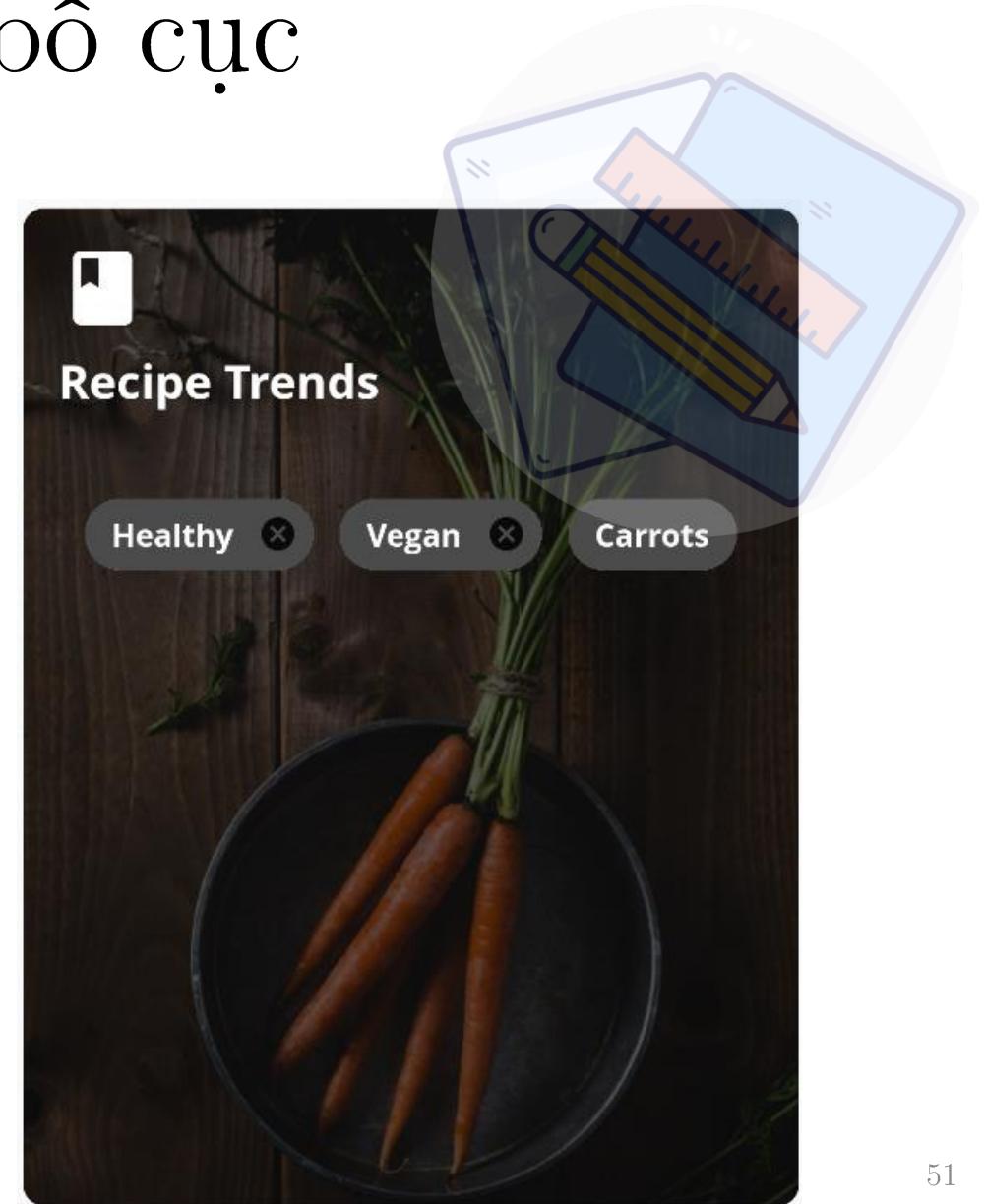
```
child: Stack(  
  children: <Widget>[  
    ...  
    Container(  
      padding: const EdgeInsets.all(16),  
      child: Column(  
        crossAxisAlignment: CrossAxisAlignment.start,  
        children: [  
          const Icon(Icons.book, color: Colors.white, size: 40),  
          const SizedBox(height: 8),  
          Text(  
            'Recipe Trends',  
            style: FoderlichTheme.darkTextTheme.headline2,  
          ),  
          const SizedBox(height: 30),  
        ],  
      ),  
    ),  
  ],  
,
```



Các widget hiển thị và bố cục

Xây dựng Card3: thêm các chip

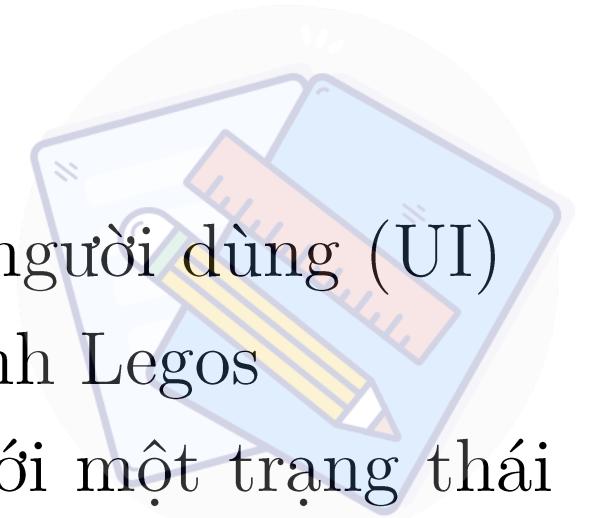
```
const SizedBox(height: 30),  
Center(  
  child: Wrap(  
    alignment: WrapAlignment.start,  
    spacing: 12,  
    children: [  
      Chip(  
        label: Text('Healthy',  
          style: FoderlichTheme.darkTextTheme.bodyText1), // Text  
        backgroundColor: Colors.black.withOpacity(0.7),  
        onDeleted: () {},  
      ), // Chip  
      Chip(  
        label: Text('Vegan',  
          style: FoderlichTheme.darkTextTheme.bodyText1), // Text  
        backgroundColor: Colors.black.withOpacity(0.7),  
        onDeleted: () {},  
      ), // Chip  
      Chip(  
        label: Text('Carrots',  
          style: FoderlichTheme.darkTextTheme.bodyText1), // Text  
        backgroundColor: Colors.black.withOpacity(0.7),  
      ), // Chip  
    ],  
  ), // Wrap  
, // Center
```



Hiểu về widget



Widget là gì?



- Một widget là một khối xây dựng của giao diện người dùng (UI)
- Xây dựng giao diện với widget giống như xếp hình Legos
- Có thể xem các widget như một hàm của UI. Với một trạng thái (state) của ứng dụng, phương thức build() của widget xây dựng nên UI cho widget

UI = f(state)

Screen Build

Xây dựng Card2

Column



Container

AuthorCard
Expanded

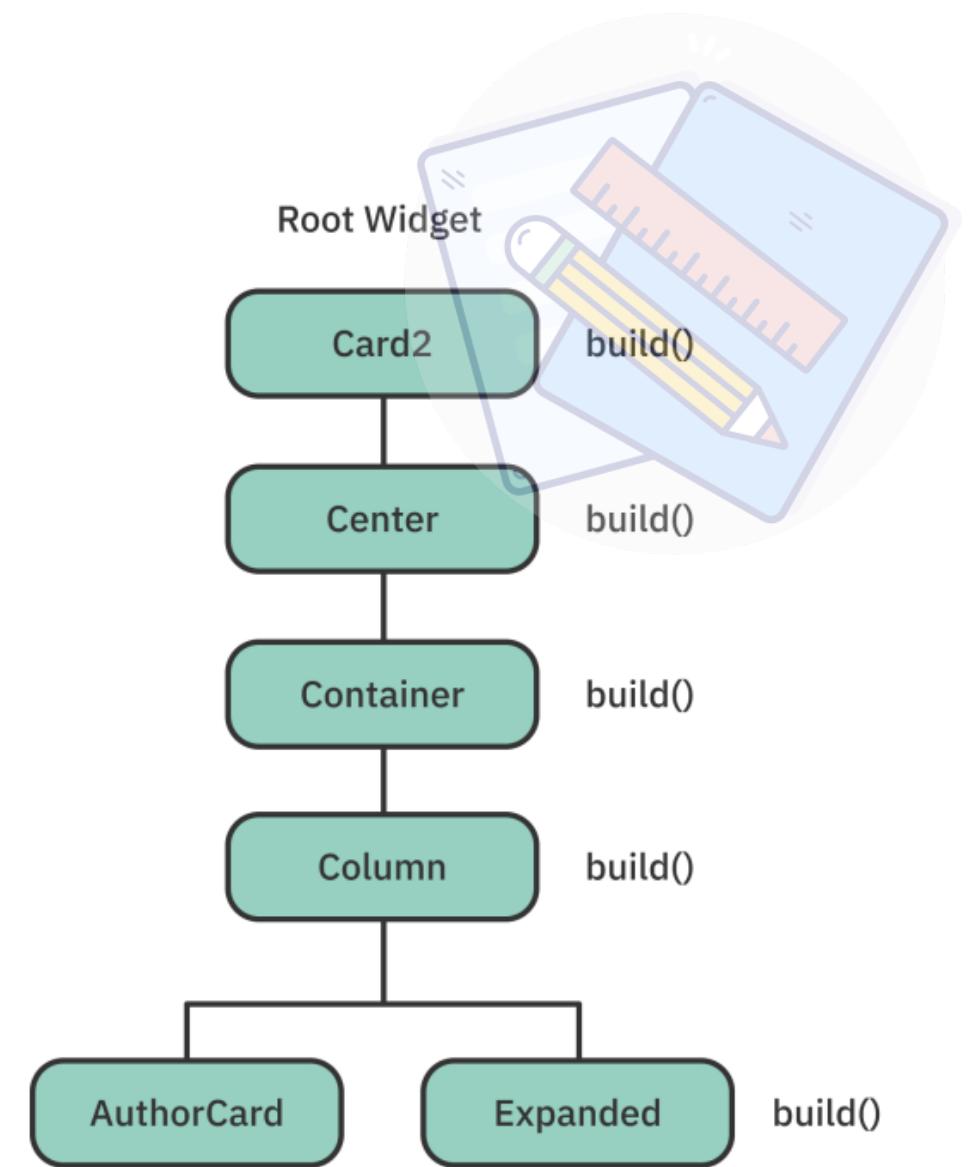


Cây widget

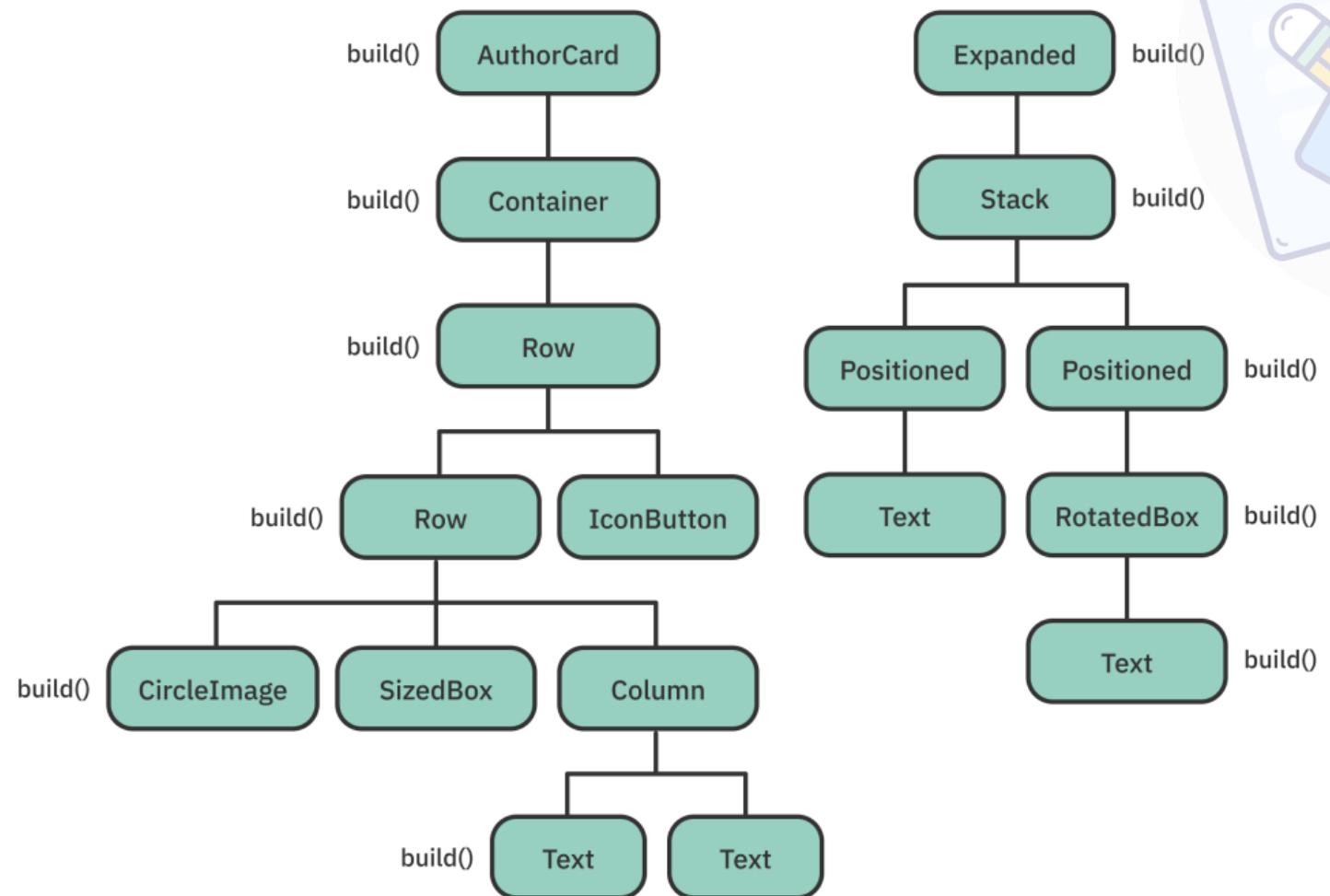
- Mỗi widget có một phương thức build()
- UI được tạo bằng cách lồng ghép widget này vào widget khác tạo thành cấu trúc dạng cây
 - Một widget có thể chứa các widget khác gọi là các widget con (children)
- Cây widget cung cấp một *bản vẽ thiết kế* miêu tả bố cục của UI. Flutter duyệt qua các nút trong cây widget và gọi từng phương thức build() để hợp thành UI hoàn chỉnh



Cây widget của Card2



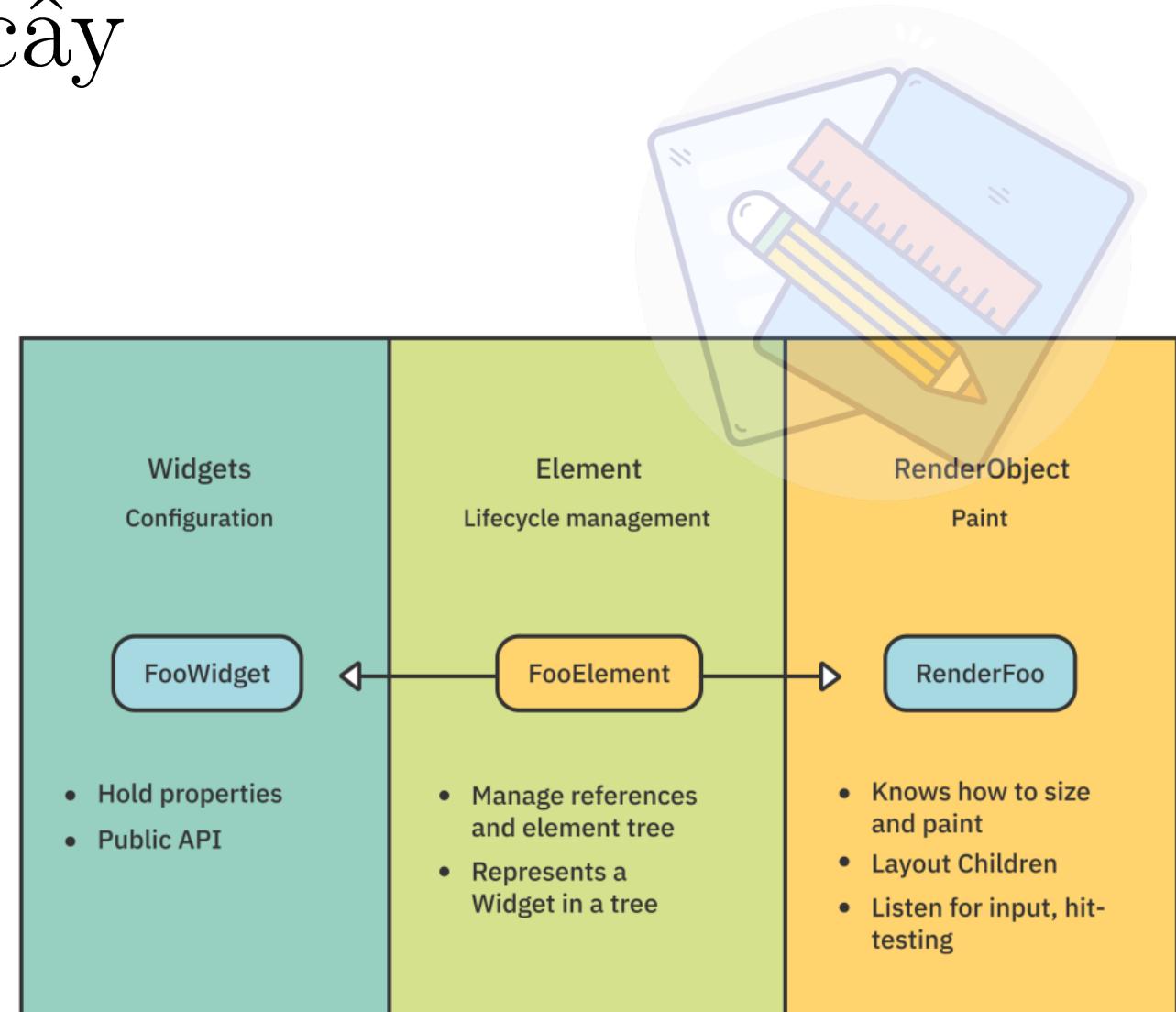
Cây widget của Card2



Ba cấu trúc dạng cây

Vì lý do hiệu năng, Flutter xây dựng giao diện nhờ vào không chỉ một mà là ba cây cùng lúc

- Cây Widget
- Cây Element
- Cây RenderObject

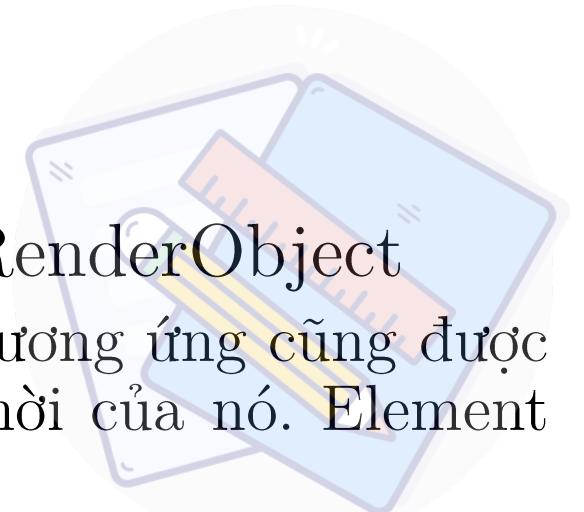


Ba cấu trúc dạng cây

- **Widget:** API công khai của Flutter, giúp tạo bản vẽ thiết kế / *cấu hình* cho giao diện người dùng (UI)
 - Chỉ chứa thông tin cấu hình, chi phí tạo hủy Widget là nhỏ
 - Bất biến (immutable), nếu có thay đổi thì phải tạo mới
- **RenderObject:** chịu trách nhiệm vẽ, định bố cục một Widget cụ thể, xử lý tương tác người dùng
 - Có thể được cập nhật cấu hình (mutable)
 - Chi phí khởi tạo lớn, RenderObject được tái sử dụng khi có thể



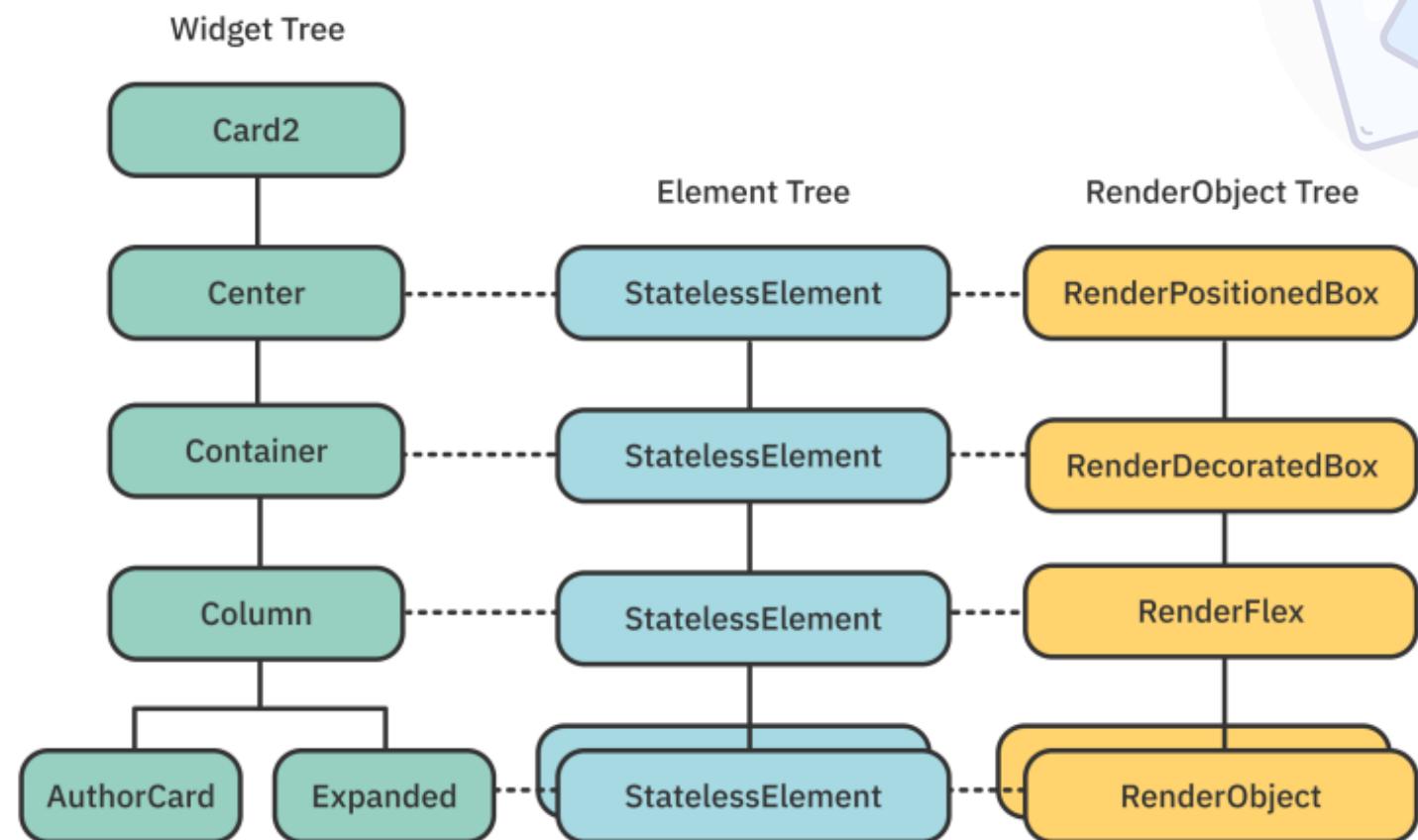
Ba cấu trúc dạng cây



- **Element:** đảm bảo sự đồng bộ giữa Widget và RenderObject
 - Lần đầu Widget được tạo và chèn vào cây, Element tương ứng cũng được tạo và tham chiếu đến Widget như cấu hình hiện thời của nó. Element cũng lưu giữ tham chiếu đến RenderObject
 - Khi đối tượng Widget được tái tạo lại (ví dụ như do trạng thái thay đổi), Flutter kiểm tra xem có thể cập nhật lại Element hay phải tạo mới
 - Nếu Widget mới có cùng kiểu (và cùng key) với Widget cũ thì Element được cập nhật tham chiếu Widget mới và RenderObject vẽ lại Widget
 - Nhà phát triển có thể tương tác với Element của Widget thông qua giao diện BuildContext: Widget build(BuildContext context) {...}

Ba cấu trúc dạng cây

Ba cấu trúc các cây của Card2



Các loại widget

Có ba loại widget chính:

- StatelessWidget
- StatefulWidget
- InheritedWidget



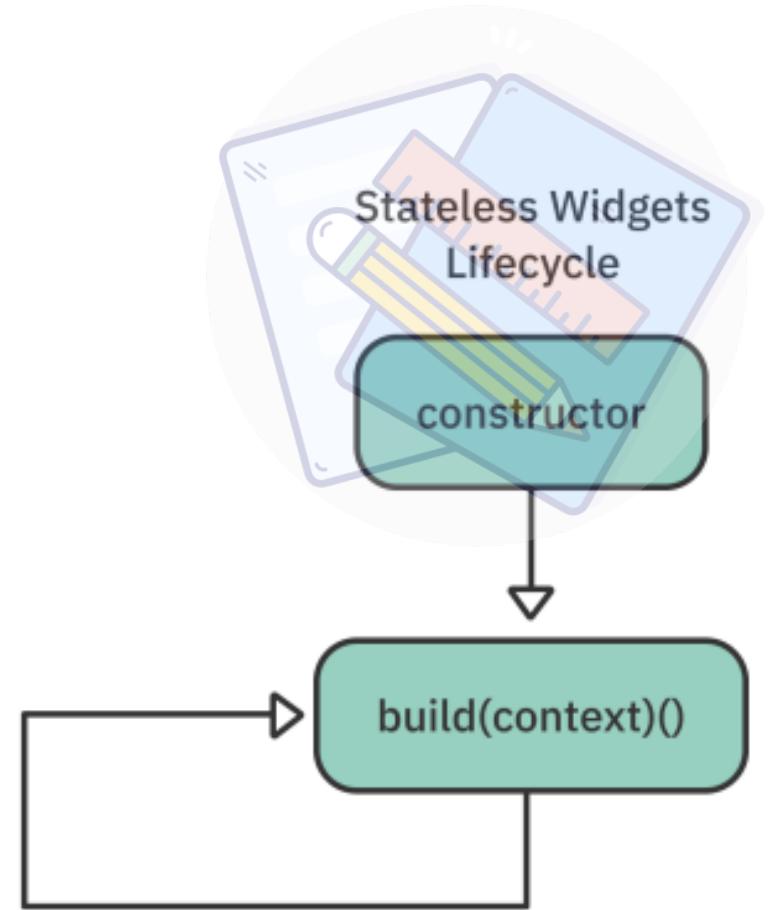
Tất cả các widget là bất biến (immutable) nhưng một số có trạng thái (state) đính kèm thông qua element của widget

StatelessWidget

Widget không yêu cầu trạng thái có thể thay đổi (thông thường thông tin cấu hình widget được widget cha cung cấp)

Phương thức build của dạng widget này chỉ được gọi khi:

- Widget được chèn vào cây widget lần đầu
- Widget cha thay đổi cấu hình của widget
- InheritedWidget mà widget phụ thuộc vào thay đổi
 - Theme.of(context), MediaQuery.of(context), ...

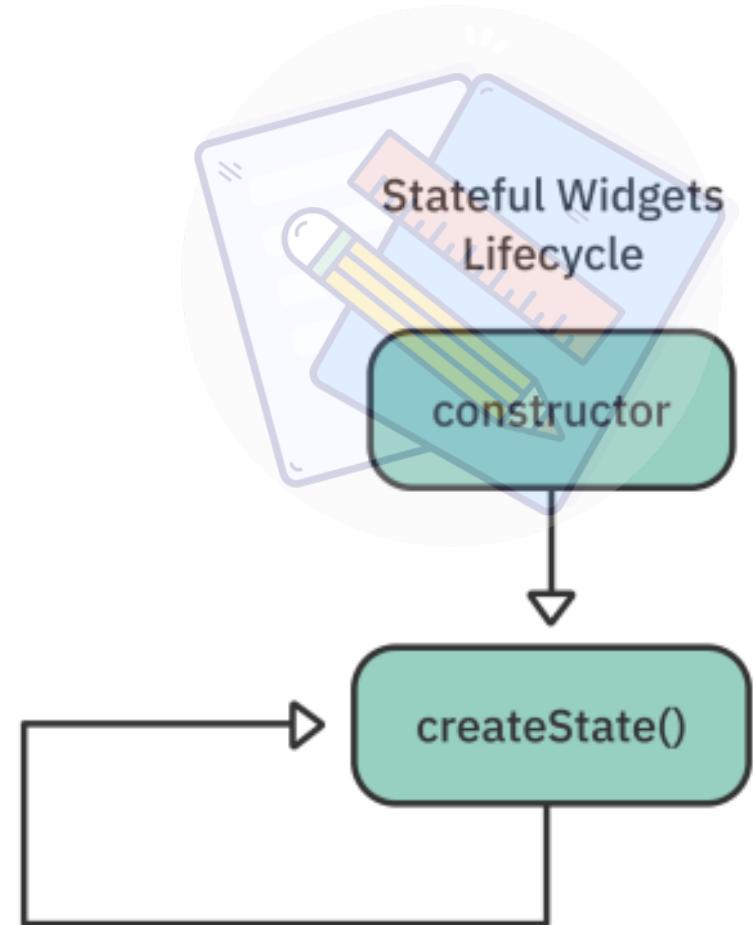


StatefulWidget

Widget có trạng thái có thể thay đổi được

StatefulWidget lưu giữ trạng thái trong một lớp trạng thái riêng

- Phải cài đặt createState()
- Trạng thái được quản lý bởi element (BuildContext) của widget
- Trạng thái được bảo tồn qua các lần widget được tái tạo

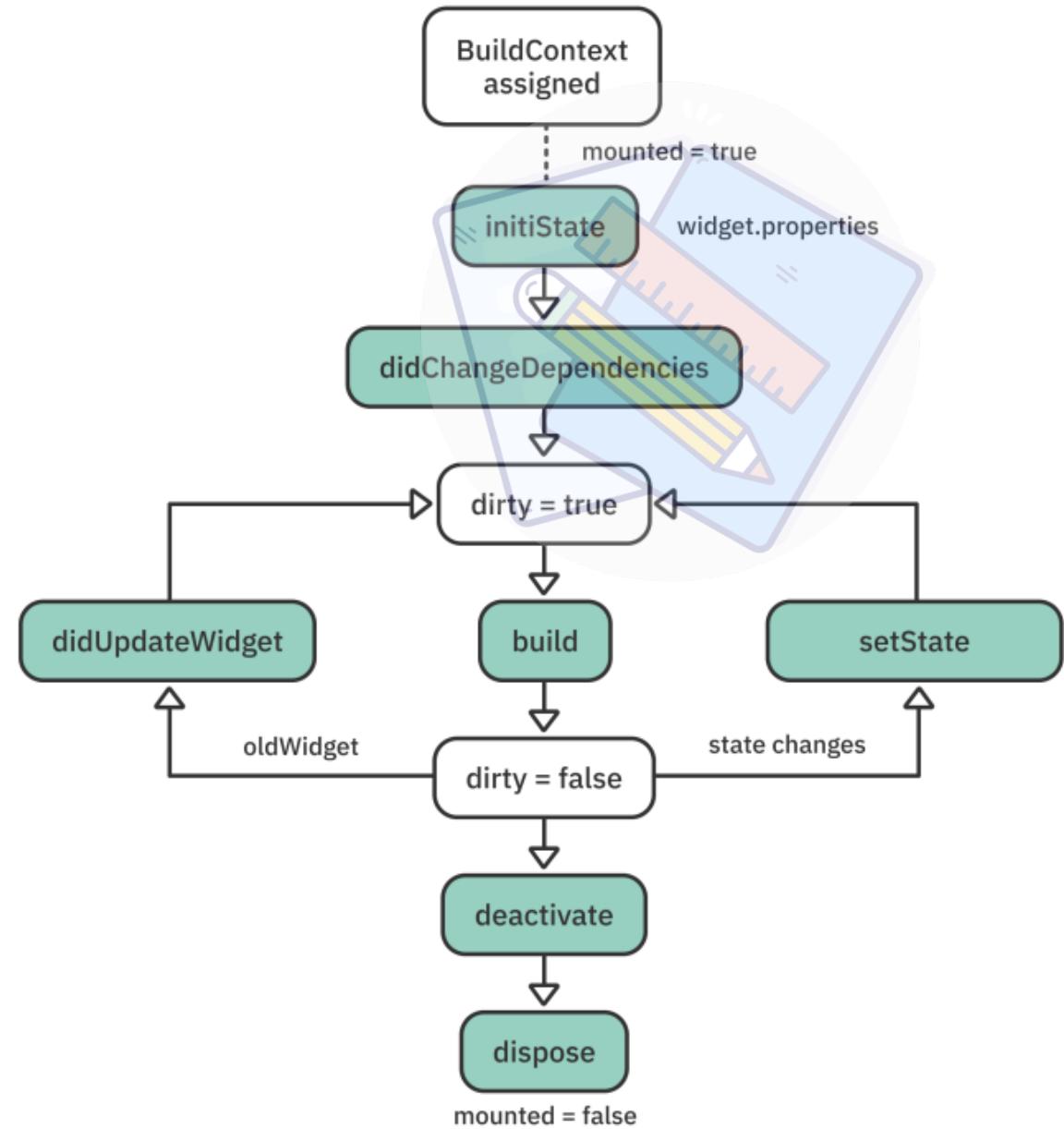


StatefulWidget

Widget có trạng thái có thể thay đổi được

StatefulWidget lưu giữ trạng thái trong một lớp trạng thái riêng

- Phải cài đặt createState()
- Trạng thái được quản lý bởi element (BuildContext) của widget
- Trạng thái được bảo tồn qua các lần widget được tái tạo



StatefulWidget

- **void initState():** được gọi *một lần duy nhất* khi widget được tạo, dùng để khởi tạo các trạng thái
 - BuildContext chưa sẵn dùng hoàn toàn

```
@override  
void initState() {  
    super.initState();  
    WidgetsBinding.instance.endOfFrame.then(  
        (_) {  
            if (mounted) {  
                // Có thể sử dụng context ở đây  
            }  
        },  
    );  
}
```



StatefulWidget

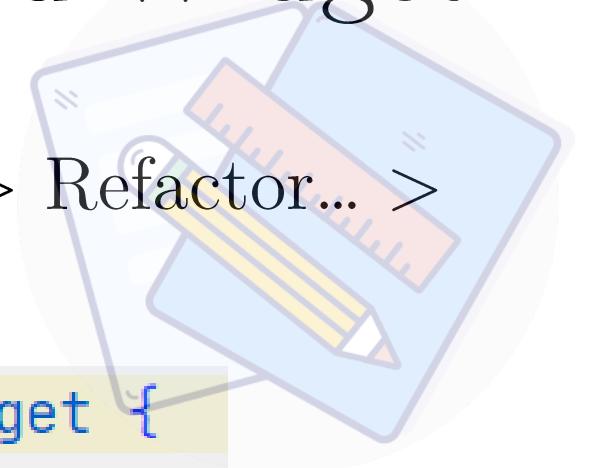


- **void didChangeDependencies():** được gọi ngay sau initState, dùng để cập nhật trạng thái khi *InheritedWidget* phụ thuộc thay đổi (`MediaQuery.of()`, `Theme.of()`, ...)
 - Có thể được gọi nhiều hơn lần
- **void didUpdateWidget(covariant T oldWidget):** dùng để cập nhật trạng thái khi *cấu hình widget* thay đổi (cấu hình cũ của widget được đưa vào tham số)
 - Có thể được gọi nhiều hơn một lần
- **void dispose():** được gọi khi widget được xóa bỏ khỏi cây widget, dùng để dọn dẹp trạng thái, giải phóng tài nguyên

Chuyển AuthorCard sang StatefulWidget

Chuột phải lên dòng định nghĩa lớp AuthorCard > Refactor... > Convert to StatefulWidget

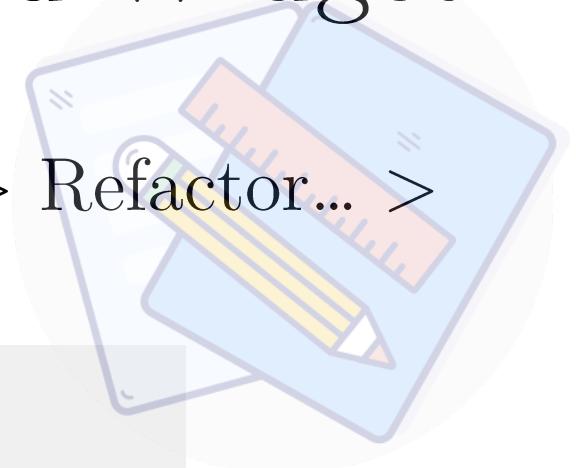
```
class AuthorCard extends StatelessWidget {  
    fina  
    fina  Convert to StatefulWidget  
    final ImageProvider? imageProvider;
```



Chuyển AuthorCard sang StatefulWidget

Chuột phải lên dòng định nghĩa lớp AuthorCard > Refactor... > Convert to StatefulWidget

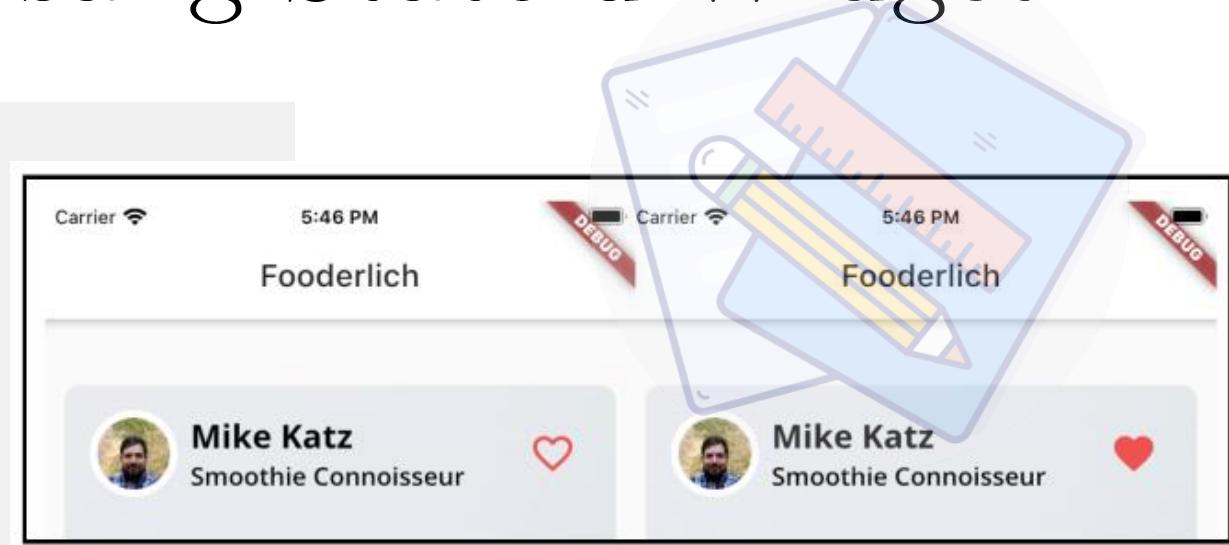
```
class AuthorCard extends StatefulWidget {  
    ...  
    @override  
    State<AuthorCard> createState() => _AuthorCardState();  
}  
  
class _AuthorCardState extends State<AuthorCard> {  
    @override  
    Widget build(BuildContext context) {  
        ...  
    }  
}
```



Chuyển AuthorCard sang StatefulWidget

```
class _AuthorCardState extends State<AuthorCard> {
  bool _isFavorited = false;

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: const EdgeInsets.all(16),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          ...
          IconButton(
            icon: Icon(_isFavorited ? Icons.favorite : Icons.favorite_border),
            iconSize: 30,
            color: Colors.red[400],
            onPressed: () {
              setState(() {
                _isFavorited = !_isFavorited;
              });
            },
          ),
        ],
      );
  }
}
```

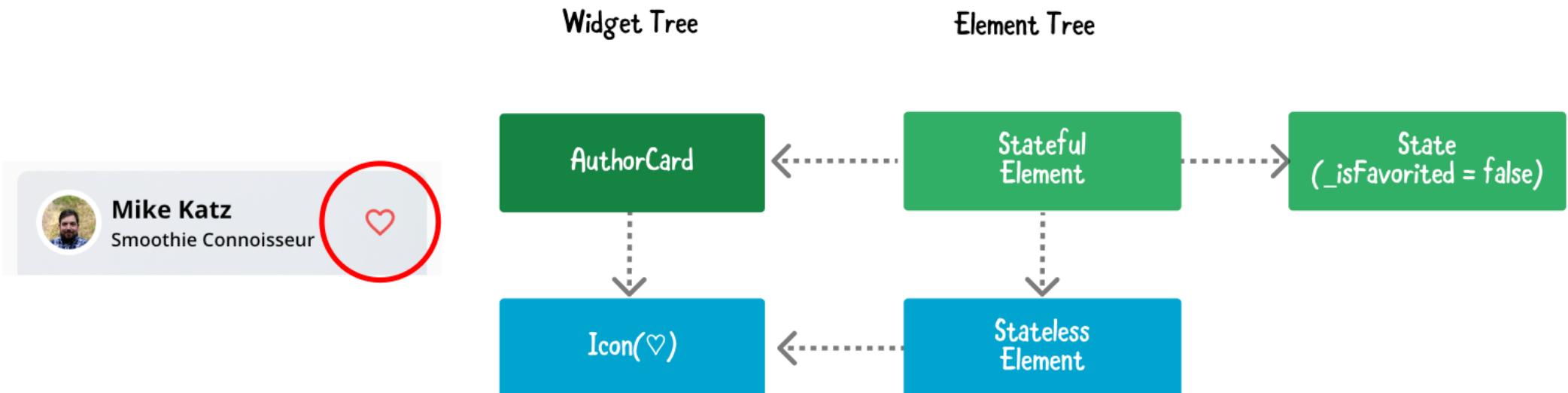


Có thể thay đổi trạng thái trước sau đó gọi setState:

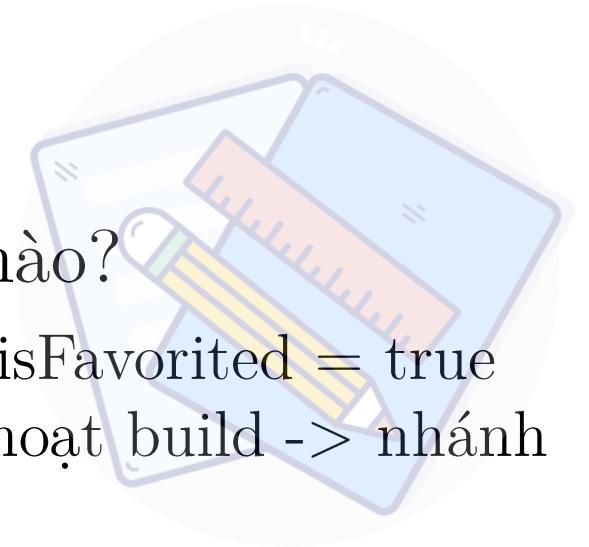
```
onPressed: () {
  _isFavorited = !_isFavorited;
  setState(() {});
},
```

StatefulWidget

Cây element quản lý thay đổi trạng thái như thế nào?

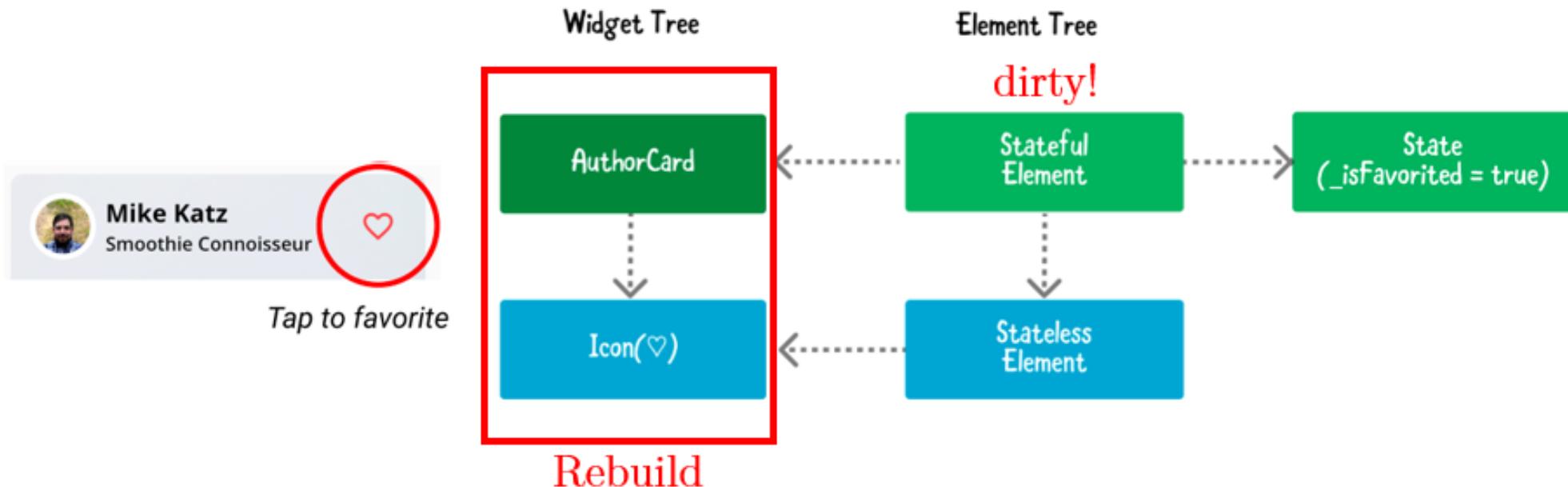


StatefulWidget

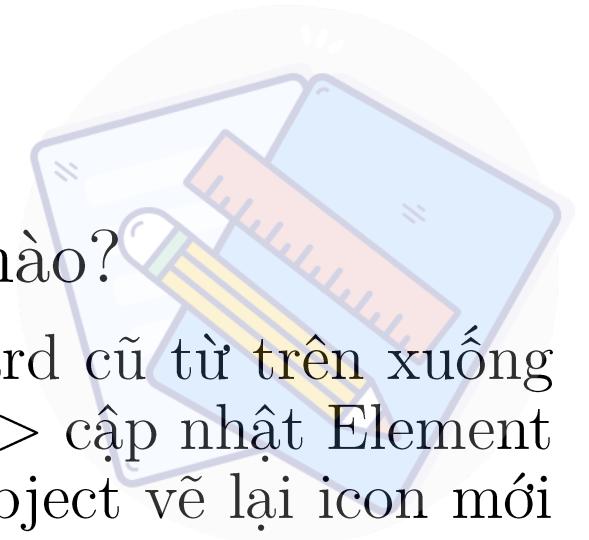


Cây element quản lý thay đổi trạng thái như thế nào?

Người dùng chạm vào yêu thích -> setState được gọi -> `_isFavorited = true` -> Element của AuthorCard bị đánh dấu là dirty -> kích hoạt build -> nhánh cây Widget AuthorCard được tái tạo từ trên xuống

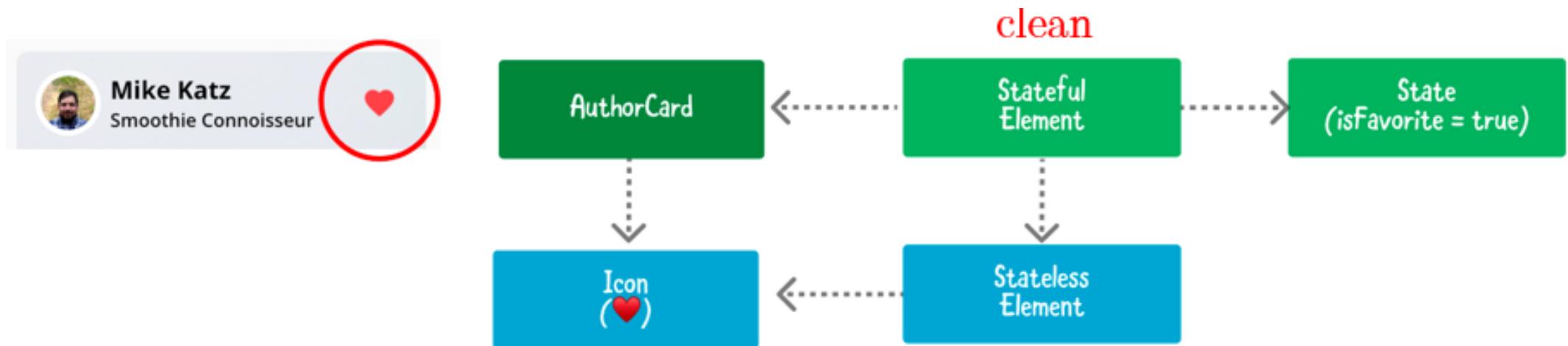


StatefulWidget

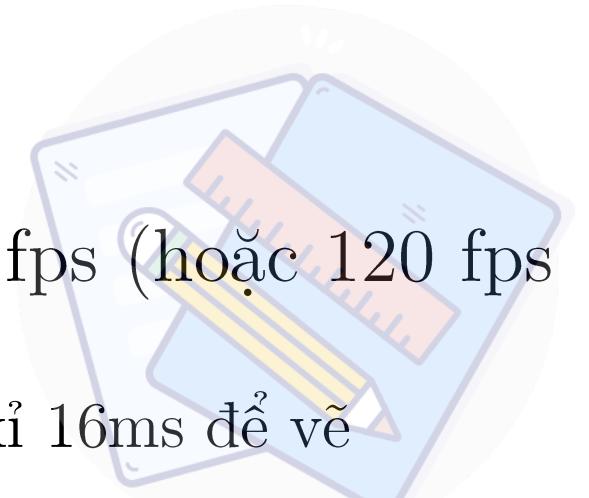


Cây element quản lý thay đổi trạng thái như thế nào?

Flutter so sánh cây Widget AuthorCard mới và AuthorCard cũ từ trên xuống từng Widget (theo kiểu và key) để cập nhật cây Element -> cập nhật Element của icon để tham chiếu đến Widget icon mới -> RenderObject vẽ lại icon mới trên màn hình

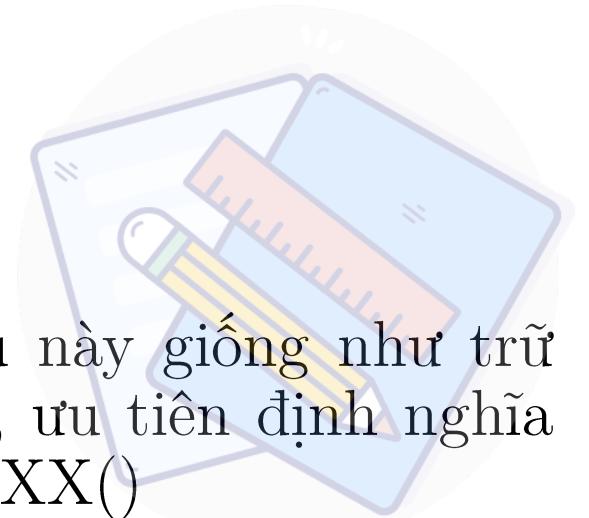


StatefulWidget



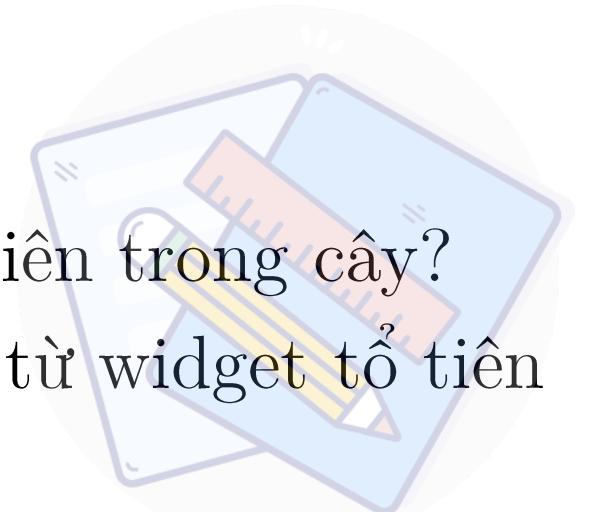
- Flutter cố gắng đáp ứng để ứng dụng chạy ở 60 fps (hoặc 120 fps trên các thiết bị có khả năng)
 - Màn hình được vẽ lại 60 lần/giây, mỗi frame có xấp xỉ 16ms để vẽ
- Lời gọi **setState** kích hoạt phương thức build của StatefulWidget đang xét, tái tạo lại widget cùng các widget hậu duệ

StatefulWidget



- Một số cách cải thiện hiệu năng:
 - Đánh dấu các widget là **const** khi nào có thể (điều này giống như trữ tạm lại widget và tái sử dụng khi cần thiết). Do đó, ưu tiên định nghĩa và sử dụng widget dạng lớp thay vì các hàm `_buildXXX()`
 - Có thể lưu giữ các widget không thay đổi trong các biến **final** và tái sử dụng trong build
 - Có thể tách nhỏ StatefulWidget để giới hạn việc phạm vi tái xây dựng hoặc tạo StatefulWidget nhận vào widget khác vào làm child
 - Sử dụng các widget `*Builder` (giới thiệu sau) thay vì `setState` để chỉ cập nhật lại những thành phần giao diện cần thiết

InheritedWidget

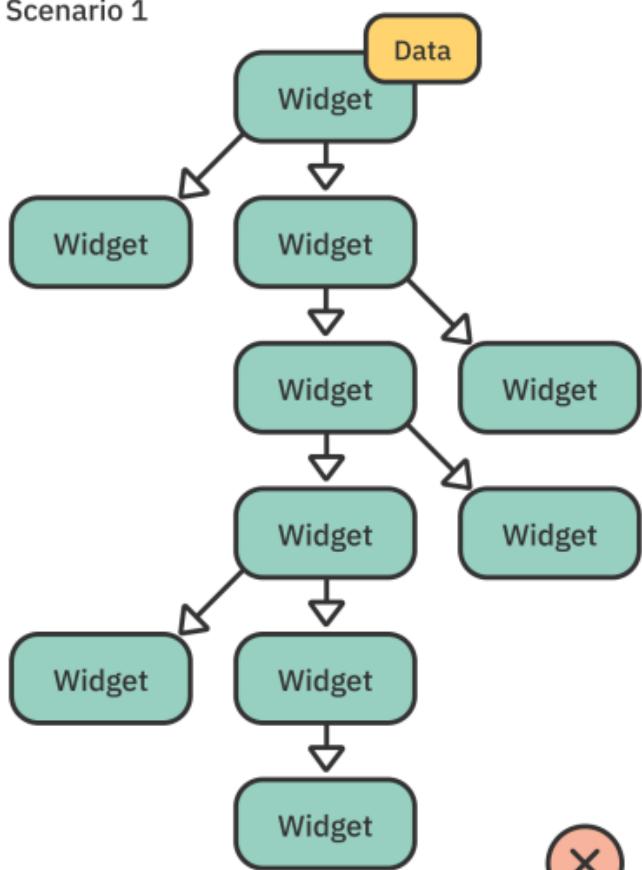


Làm thế nào truy xuất dữ liệu của các widget tổ tiên trong cây?

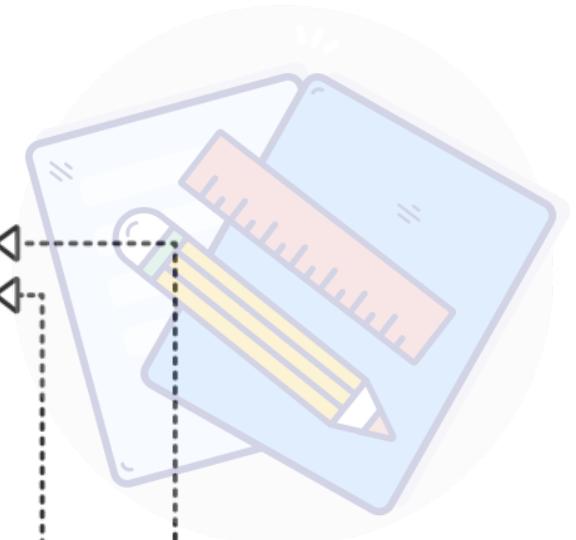
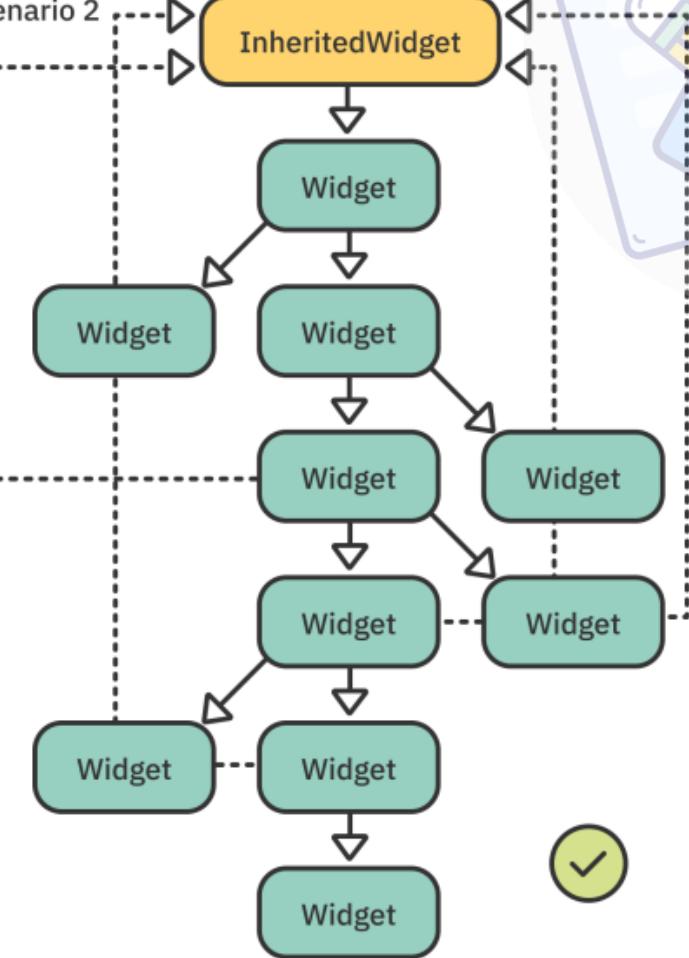
- **Kịch bản 1:** truyền dữ liệu như là tham số dần từ widget tổ tiên xuống widget con trong cây
- **Kịch bản 2:** thêm một InheritedWidget vào cây widget. Các widget hậu duệ bất kỳ có thể truy xuất đến InheritedWidget
 - Nhà phát triển thường không sử dụng trực tiếp InheritedWidget mà thông qua các gói wrapper ví dụ như gói Provider

InheritedWidget

Scenario 1



Scenario 2



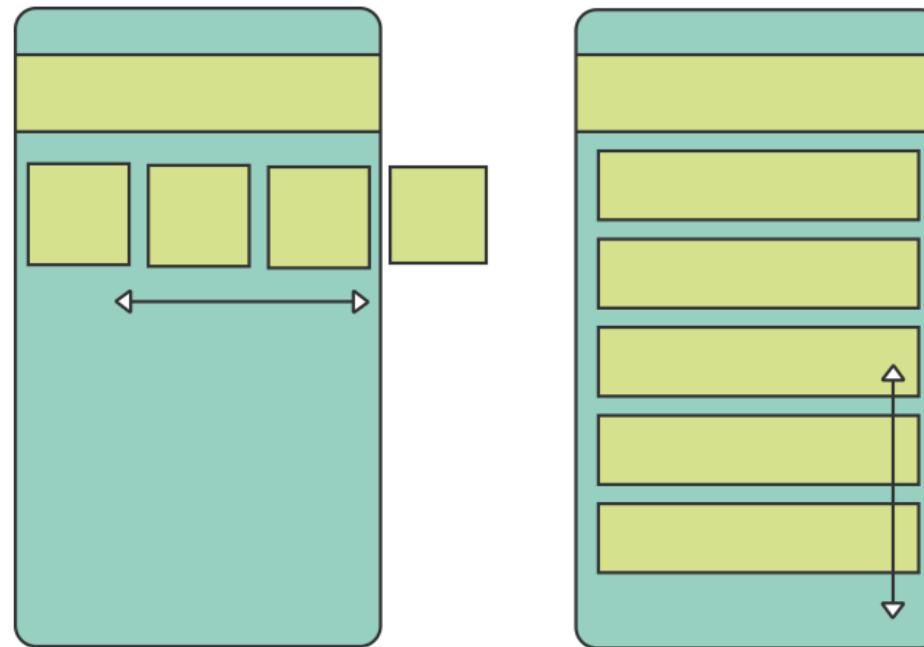
Các widget cuộn



ListView



- Một widget cuộn tuyến tính rất phổ biến, hỗ trợ cuộn theo chiều dọc và chiều ngang
 - Các widget Row và Column là tương tự với ListView nhưng không có tính năng cuộn



ListView



Các phương thức xây dựng thường dùng của [ListView](#):

- Phương thức xây dựng **mặc định** nhận vào một danh sách các widget con (children). Tất cả các phần tử con sẽ được tạo cho dù không đang ở trong khung nhìn
 - Chỉ nên dùng phương thức này khi số lượng phần tử là nhỏ
- **ListView.builder()**: tạo list theo yêu cầu (chỉ những phần tử trong khung nhìn mới được tạo)
- **ListView.separated()**: tạo list theo yêu cầu và đồng thời tạo ngăn cách giữa các phần tử

ListView

Một số tham số quan trọng:

- **scrollDirection:** điều khiển trục cuộn (mặc định là trục đứng)
- **itemCount:** số lượng phần tử trong ListView
- **itemBuilder:** hàm tạo widget con của ListView
- **separatorBuilder:** hàm tạo widget cách khoảng giữa các con



ListView

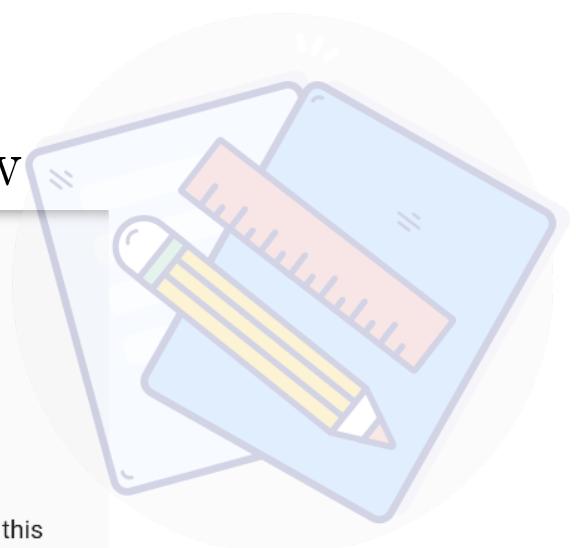
TodayRecipeListView



FriendPostListView

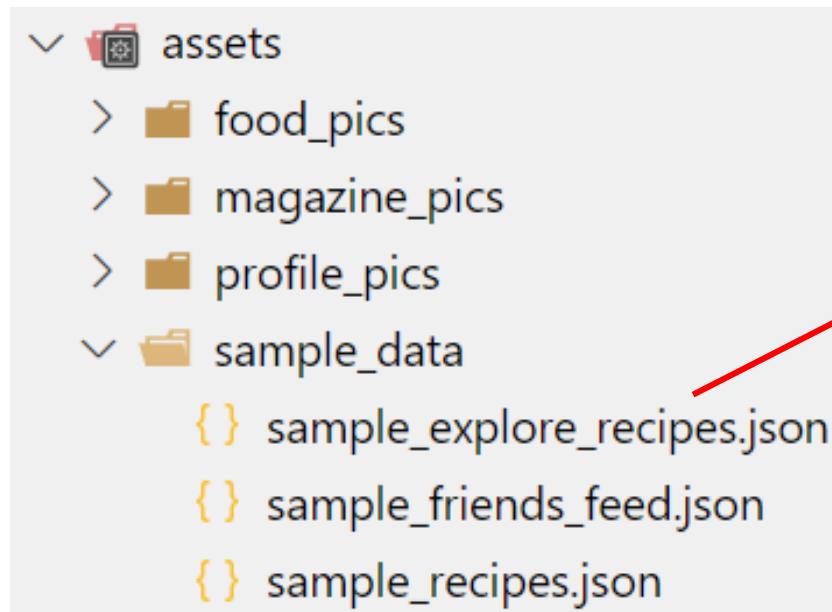
Social Chefs 🍴

- Made a delicious salad today!
20 mins ago
- Cooked a widget this morning.
30 mins ago
- Going to power through my day with this nutty smoothie to finish up Flutter apprentice for our readers!
40 mins ago
- Cooking up some steak 🥩 today, state should be rare, medium or medium well?
50 mins ago
- Kicking off the morning with mike's green smoothie recipe. Time to edit these chapters!
50 mins ago
- Not sure where I should travel to eat today.
60 mins ago
- Hot off the press, cooking up more books



TodayRecipeListView

Dữ liệu ví dụ được lưu trong tập tin JSON



```
{  
  "recipes": [  
    {  
      "id": "1",  
      "cardType": "card1",  
      "title": "The Art of Dough",  
      "subtitle": "Editor's Choice",  
      "backgroundImage": "assets/magazine_pics/card_bread.jpg",  
      "backgroundImageSource": "https://www.foodiesfeed.com/free-food-photo/baker-with-wheat-flour/",  
      "message": "Learn to make the perfect bread.",  
      "authorName": "Ray Wenderlich",  
      "role": "Founder of Raywenderlich",  
      "authorImage": "assets/profile_pics/person_ray.jpeg",  
      "durationInMinutes": 15,  
      "dietType": "Standard",  
      "calories": 136,  
      "tags": ["Baking", "Dough", "Bread"]  
    }  
  ]  
}
```



TodayRecipeListView

Lớp đọc dữ liệu từ tập tin JSON

```
// Mock recipe service that grabs sample json data to mock recipe request/response
class MockFooderlichService {
    // Batch request that gets both today recipes and friend's feed
    Future<ExploreData> getExploreData() async {
        final todayRecipes = await _getTodayRecipes();
        final friendPosts = await _getFriendFeed();

        return ExploreData(todayRecipes, friendPosts);
    }

    // Get sample explore recipes json to display in ui
    Future<List<ExploreRecipe>> _getTodayRecipes() async {
        // Simulate api request wait time
        await Future.delayed(const Duration(milliseconds: 1000));
        // Load json from file system
        final dataString =
            await _loadAsset('assets/sample_data/sample_explore_recipes.json');
        // Decode to json
        final Map<String, dynamic> json = jsonDecode(dataString);
    }
}
```



TodayRecipeListView

TodayRecipeListView nhận vào một danh sách các ExploreRecipe và sẽ xây dựng danh sách cuộn các Card tương ứng

```
class TodayRecipeListView extends StatelessWidget {  
    final List<ExploreRecipe> recipes;  
  
    const TodayRecipeListView({  
        Key? key,  
        required this.recipes,  
    }) : super(key: key);  
  
    @override  
    Widget build(BuildContext context) {
```



```
class RecipeCardType {  
    static const card1 = 'card1';  
    static const card2 = 'card2';  
    static const card3 = 'card3';  
}  
  
class ExploreRecipe {  
    String id;  
    String cardType;  
    String title;  
    String subtitle;  
    String backgroundImage;  
    String backgroundImageSource;  
    String message;  
    String authorName;  
    String role;  
    String profileImage;  
    int durationInMinutes;  
    String dietType;
```

TodayRecipeListView

TodayRecipeListView nhận vào một danh sách các ExploreRecipe và sẽ xây dựng danh sách cuộn các Card tương ứng

```
Widget build(BuildContext context) {
  return Padding(
    padding: const EdgeInsets.only(left: 16, right: 16, top: 16),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          'Recipes of the Day 🔍',
          style: Theme.of(context).textTheme.headline1,
        ), // Text
        const SizedBox(height: 16),
        Expanded(
          child: ListView.separated(
            scrollDirection: Axis.horizontal,
            itemCount: recipes.length,
            itemBuilder: (context, index) {
              final recipe = recipes[index];
              return buildCard(recipe);
            },
            separatorBuilder: (context, index) {
              return const SizedBox(width: 16);
            },
          ), // ListView.separated
        ),
      ],
    ),
  );
}
```

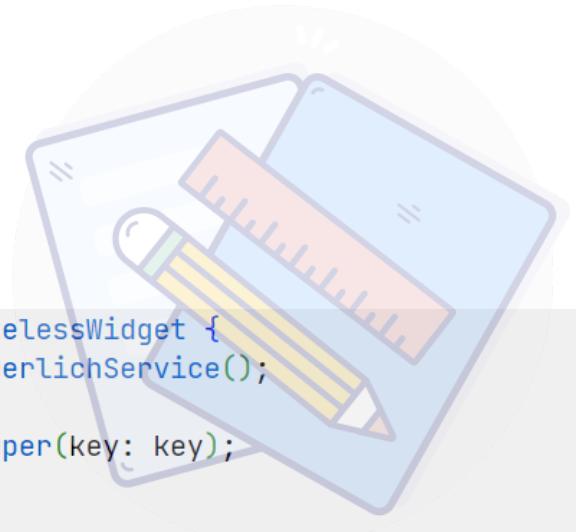
```
Widget buildCard(ExploreRecipe recipe) {
  if (recipe.cardType == RecipeCardType.card1) {
    return Card1(recipe: recipe);
  } else if (recipe.cardType == RecipeCardType.card2) {
    return Card2(recipe: recipe);
  } else if (recipe.cardType == RecipeCardType.card3) {
    return Card3(recipe: recipe);
  } else {
    throw Exception('This card doesn\'t exist yet');
  }
}
```



TodayRecipeListView

List1Screen đọc dữ liệu và tạo TodayRecipeListView

- Dữ liệu được đọc một cách bất đồng bộ: kết quả trả về là một **Future<T>**
 - Dùng **FutureBuilder** để tạo widget dựa trên kết quả T trả về từ một Future
- Trong khi quá trình đọc chưa xong, hiển thị tiến trình với **CircularProgressIndicator**

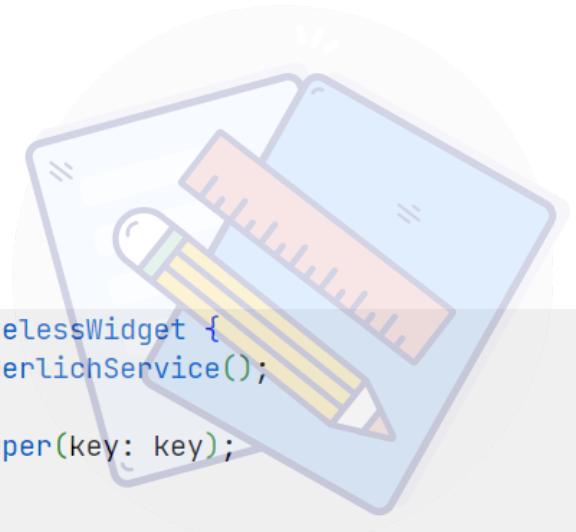


```
class List1Screen extends StatelessWidget {  
    final mockService = MockFooderlichService();  
  
    List1Screen({Key? key}) : super(key: key);  
  
    @override  
    Widget build(BuildContext context) {  
        return FutureBuilder(  
            future: mockService.getExploreData(),  
            builder: (context, AsyncSnapshot<ExploreData> snapshot) {  
                if (snapshot.connectionState == ConnectionState.done) {  
                    return TodayRecipeListView(  
                        recipes: snapshot.data?.todayRecipes ?? [],  
                    ); // TodayRecipeListView  
                } else {  
                    return const Center(child: CircularProgressIndicator());  
                }  
            },  
        ); // FutureBuilder  
    }  
}
```

TodayRecipeListView

List1Screen đọc dữ liệu và tạo TodayRecipeListView

- Phương thức xây dựng của FutureBuilder nhận vào
 - **future**: đối tượng future sẽ trả về một kết quả dữ liệu trong tương lai
 - **builder**: callback sẽ được gọi để tạo widget



```
class List1Screen extends StatelessWidget {
  final mockService = MockFooderlichService();

  List1Screen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return FutureBuilder(
      future: mockService.getExploreData(),
      builder: (context, AsyncSnapshot<ExploreData> snapshot) {
        if (snapshot.connectionState == ConnectionState.done) {
          return TodayRecipeListView(
            recipes: snapshot.data?.todayRecipes ?? [],
          ); // TodayRecipeListView
        } else {
          return const Center(child: CircularProgressIndicator());
        }
      },
    ); // FutureBuilder
  }
}
```

TodayRecipeListView

Hiển thị List1Screen

```
class Home extends StatefulWidget {
  const Home({Key? key}) : super(key: key);

  @override
  State<Home> createState() => _HomeState();
}

class _HomeState extends State<Home> {
  int _selectedIndex = 0;

  static final List<Widget> _pages = <Widget>[
    List1Screen(),
    Container(color: Colors.green),
    Container(color: Colors.blue),
  ]; // <Widget>[]

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }
}
```



TodayRecipeListView

Trở lại phương thức build của TodayRecipeListView

- ListView trong trường hợp này là một phần tử con của Column
- **Tại sao phải bao ListView trong widget Expanded?**

```
return Padding(  
  padding: const EdgeInsets.only(left: 16, right: 16, top: 16),  
  child: Column(  
    crossAxisAlignment: CrossAxisAlignment.start,  
    children: [  
      Text(  
        'Recipes of the Day 🔎',  
        style: Theme.of(context).textTheme.headline1,  
      ), // Text  
      const SizedBox(height: 16),  
      Expanded(  
        child: ListView.separated(  
          scrollDirection: Axis.horizontal,  
          itemCount: recipes.length,  
          itemBuilder: (context, index) {  
            final recipe = recipes[index];  
            return buildCard(recipe);  
          },  
          separatorBuilder: (context, index) {  
            return const SizedBox(width: 16);  
          },  
        ), // ListView.separated  
      ), // Expanded
```



TodayRecipeListView

```
return Padding(  
  padding: const EdgeInsets.only(left: 16, right: 16, top: 16),  
  child: Column(  
    crossAxisAlignment: CrossAxisAlignment.start,  
    children: [  
      Text(  
        'Recipes of the Day 🔎',  
        style: Theme.of(context).textTheme.headline1,  
      ), // Text  
      const SizedBox(height: 16),  
      ListView.separated(  
        scrollDirection: Axis.horizontal,  
        itemCount: recipes.length,  
        itemBuilder: (context, index) {  
          final recipe = recipes[index];  
          return buildCard(recipe);  
        },  
        separatorBuilder: (context, index) {  
          return const SizedBox(width: 16);  
        },  
      ), // ListView.separated  
    ],  
  ), // Column
```

===== Exception caught by rendering library =====
The following assertion was thrown during performResize():
Horizontal viewport was given unbounded height.

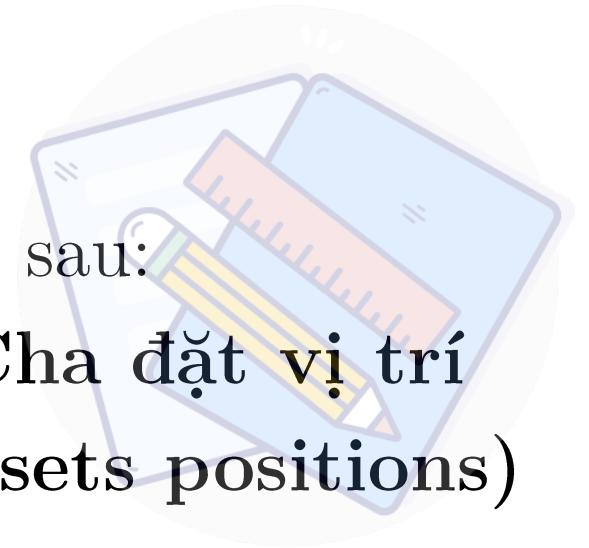


Tạo bố cục trong Flutter

Bố cục Flutter có thể được tóm tắt trong quy luật sau:

Ràng buộc đi xuống. Kích thước đi lên. Cha đặt vị trí
(Constraints go down. Sizes go up. Parent sets positions)

- Ràng buộc [BoxConstraints](#): một tập hợp 4 giá trị thực gồm chiều rộng tối thiểu, tối đa (minWidth, maxWidth), chiều cao tối thiểu, tối đa (minHeight, maxHeight)



Tạo bố cục trong Flutter

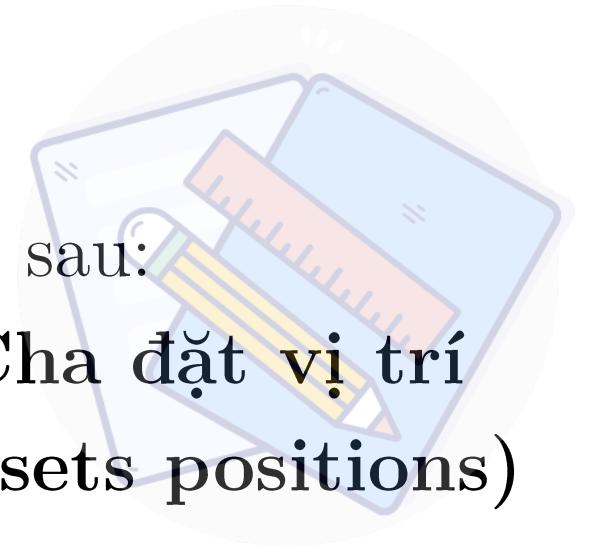
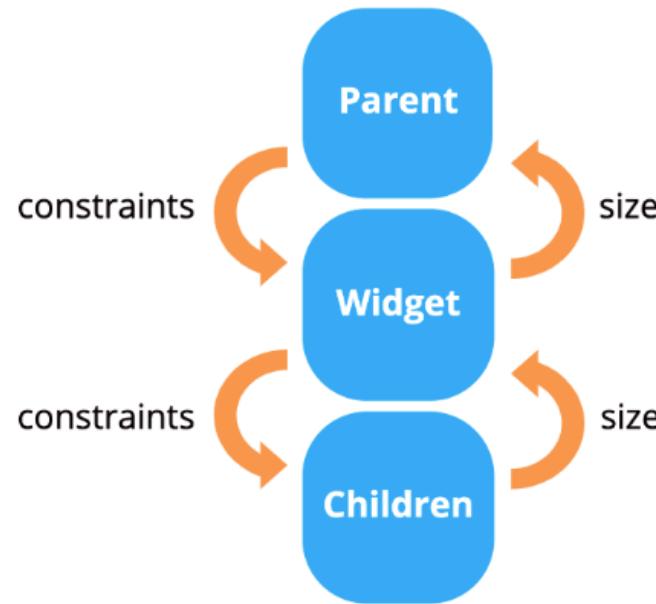


- Ràng buộc BoxConstraints: một tập hợp 4 giá trị thực gồm chiều rộng tối thiểu, tối đa (minWidth, maxWidth), chiều cao tối thiểu, tối đa (minHeight, maxHeight)
 - minWidth = maxWidth, minHeight=maxHeight: ràng buộc chặt (tight)
 - minWidth = 0, minHeight = 0: ràng buộc lỏng (loose)
 - maxWidth < double.infinity, maxHeight < double.infinity: ràng buộc có giới hạn (bounded)
 - maxWidth = double.infinity, maxHeight = double.infinity: ràng buộc không giới hạn (unbounded)
 - Ràng buộc chặt không giới hạn = mở rộng (expanding)

Tạo bố cục trong Flutter

Bố cục Flutter có thể được tóm tắt trong quy luật sau:

Ràng buộc đi xuống. Kích thước đi lên. Cha đặt vị trí
(Constraints go down. Sizes go up. Parent sets positions)

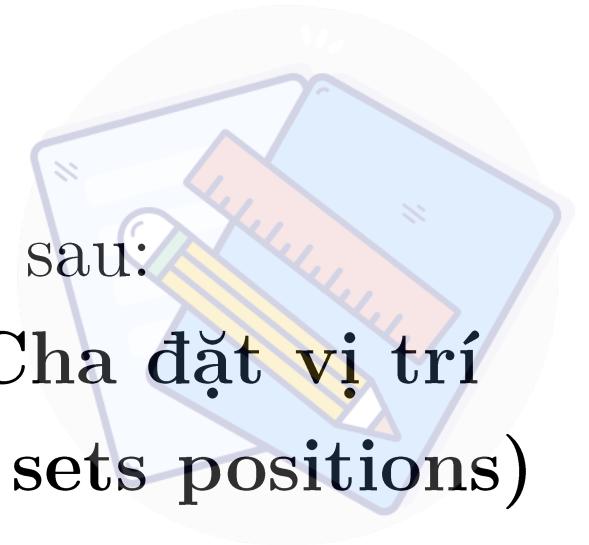


Tạo bố cục trong Flutter

Bố cục Flutter có thể được tóm tắt trong quy luật sau:

Ràng buộc đi xuống. Kích thước đi lên. Cha đặt vị trí
(Constraints go down. Sizes go up. Parent sets positions)

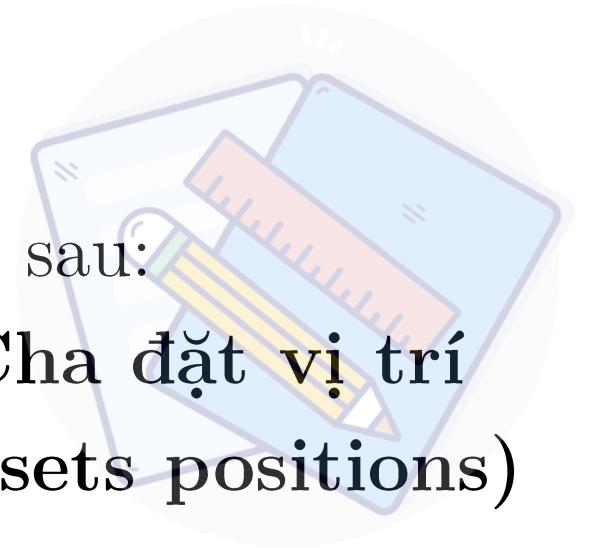
- Widget nhận ràng buộc kích thước từ widget cha
- Widget cho các widget con của nó biết các ràng buộc (có thể khác nhau cho từng con) và hỏi từng widget con kích thước mong muốn
- Widget lần lượt bố trí các widget con
- Widget báo lại cho widget cha kích thước của nó



Tạo bố cục trong Flutter

Bố cục Flutter có thể được tóm tắt trong quy luật sau:

Ràng buộc đi xuống. Kích thước đi lên. Cha đặt vị trí
(Constraints go down. Sizes go up. Parent sets positions)



Các hạn chế:

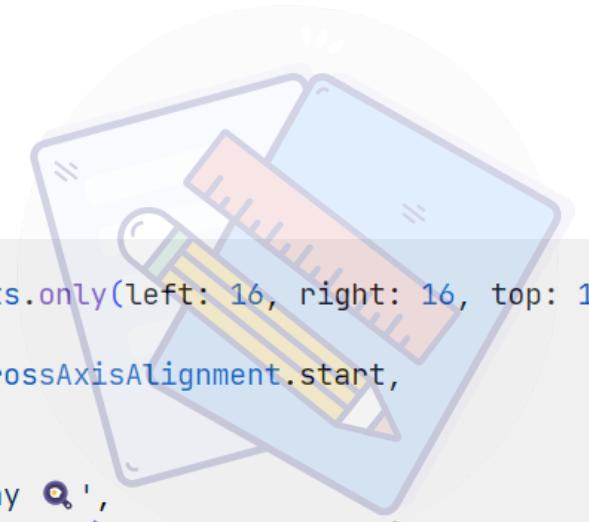
- Widget có thể có kích thước tùy ý nhưng chỉ trong ràng buộc được đưa ra bởi widget cha
- Widget không thể biết và không thể quyết định vị trí của nó trên màn hình

TodayRecipeListView

Trở lại phương thức build của TodayRecipeListView

- Column cho các widget con của nó ràng buộc chiều cao không giới hạn (unbounded constraints)
- ListView mong muốn chiều cao không giới hạn -> Lỗi
- **Giải pháp:** bao bằng widget Flexible/Expanded hoặc widget có kích thước xác định (e.g., SizedBox)

```
return Padding(  
  padding: const EdgeInsets.only(left: 16, right: 16, top: 16),  
  child: Column(  
    crossAxisAlignment: CrossAxisAlignment.start,  
    children: [  
      Text(  
        'Recipes of the Day 🔎',  
        style: Theme.of(context).textTheme.headline1,  
      ), // Text  
      const SizedBox(height: 16),  
      Expanded(  
        child: ListView.separated(  
          scrollDirection: Axis.horizontal,  
          itemCount: recipes.length,  
          itemBuilder: (context, index) {  
            final recipe = recipes[index];  
            return buildCard(recipe);  
          },  
          separatorBuilder: (context, index) {  
            return const SizedBox(width: 16);  
          },  
        ), // ListView.separated  
      ), // Expanded
```



ListView

FriendPostTile



FriendPostListView

Social Chefs



Made a delicious salad today!
20 mins ago

Cooked a widget this morning.
30 mins ago

Going to power through my day with this nutty smoothie to finish up Flutter apprentice for our readers!
40 mins ago

Cooking up some steak 🥩 today, state should be rare, medium or medium well?
50 mins ago

Kicking off the morning with mike's green smoothie recipe. Time to edit these chapters!
50 mins ago

Not sure where I should travel to eat today.
60 mins ago

Hot off the press, cooking up more books



List1



List2



Grid

FriendPostListView

FriendPostListView nhận vào một danh sách các Post và sẽ xây dựng danh sách cuộn các FriendPostTile tương ứng

```
class FriendPostListView extends StatelessWidget {  
    final List<Post> friendPosts;  
  
    const FriendPostListView({  
        Key? key,  
        required this.friendPosts,  
    }) : super(key: key);  
  
    @override  
    Widget build(BuildContext context) {
```



```
class Post {  
    String id;  
    String profileImageUrl;  
    String comment;  
    String foodPictureUrl;  
    String timestamp;  
  
    Post({  
        required this.id,  
        required this.profileImageUrl,  
        required this.comment,  
        required this.foodPictureUrl,  
        required this.timestamp,  
    });
```

FriendPostListView

FriendPostListView nhận vào một danh sách các Post và sẽ xây dựng danh sách cuộn các FriendPostTile tương ứng

```
child: Column(  
  mainAxisAlignment: MainAxisAlignment.start,  
  children: [  
    Text(  
      'Social Chefs 🍔',  
      style: Theme.of(context).textTheme.headline1,  
    ), // Text  
    const SizedBox(height: 16),  
    Expanded(  
      child: ListView.separated(  
        scrollDirection: Axis.vertical,  
        itemCount: friendPosts.length,  
        itemBuilder: (context, index) {  
          final post = friendPosts[index];  
          return FriendPostTile(post: post);  
        },  
        separatorBuilder: (context, index) {  
          return const SizedBox(height: 16);  
        },  
      ), // ListView.separated  
    ), // Expanded  
    const SizedBox(height: 16),  
  ],  
) // Column
```



FriendPostListView

List2Screen đọc dữ liệu và tạo FriendPostListView

```
class List2Screen extends StatelessWidget {
  final mockService = MockFooderlichService();

  List2Screen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return FutureBuilder(
      future: mockService.getExploreData(),
      builder: (context, AsyncSnapshot<ExploreData> snapshot) {
        if (snapshot.connectionState == ConnectionState.done) {
          return FriendPostListView(
            friendPosts: snapshot.data?.friendPosts ?? [],
          ); // FriendPostListView
        } else {
          return const Center(child: CircularProgressIndicator());
        }
      },
    ); // FutureBuilder
  }
}
```



FriendPostListView

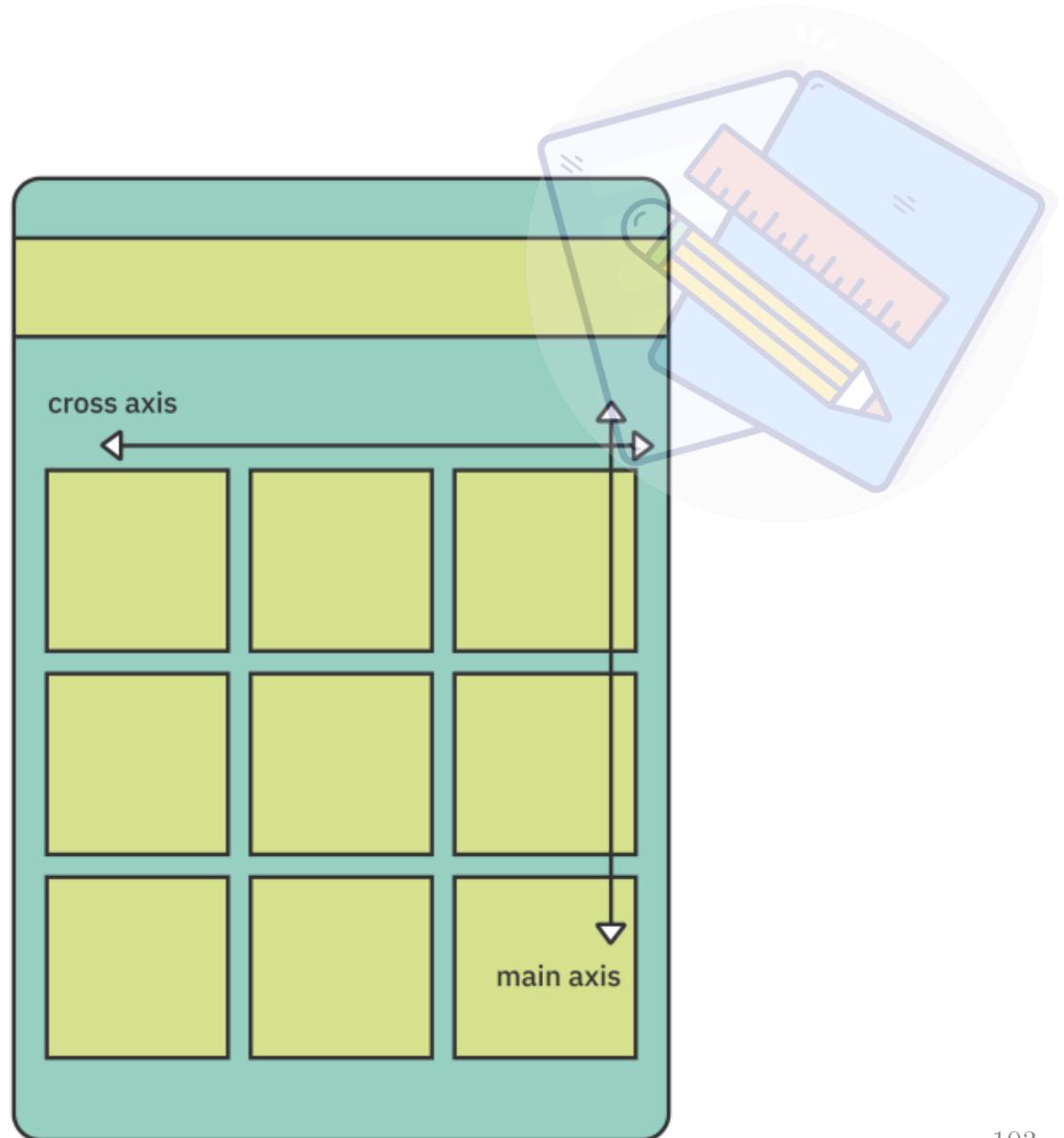
Hiển thị List2Screen

```
class Home extends StatefulWidget {  
  const Home({Key? key}) : super(key: key);  
  
  @override  
  State<Home> createState() => _HomeState();  
}  
  
class _HomeState extends State<Home> {  
  int _selectedIndex = 0;  
  
  static final List<Widget> _pages = <Widget>[  
    List1Screen(),  
    List2Screen(),  
    Container(color: Colors.blue),  
  ]; // <Widget>[]  
  
  void _onItemTapped(int index) {  
    setState(() {  
      _selectedIndex = index;  
    });  
  }  
}
```



GridView

- Mảng 2 chiều các widget, hỗ trợ cuộn theo chiều dọc và chiều ngang



GridView

Các phương thức xây dựng của [GridView](#):

- Phương thức xây dựng **mặc định** nhận vào một danh sách tường minh các widget con (children)
- **GridView.builder()**: tạo mảng cuộn 2 chiều các widget theo yêu cầu (on demand)
- **GridView.count()**: tạo mảng cuộn 2 chiều các widget với số phần tử xác định trên trục cắt
- **GridView.custom()**: tạo mảng cuộn 2 chiều các widget với SliverGridDelegate và SliverChildDelegate tùy biến
- **GridView.extent()**: tạo mảng cuộn 2 chiều các widget với các phần tử trên trục cắt có một khoảng rộng (extent) tối đa



GridView

scrollDirection: điều khiển trục cuộn (trục chính), mặc định là trục đứng

gridDelegate: giúp tạo bố cục các widget con trong GridView

- **SliverGridDelegateWithFixedCrossAxisCount**: tạo lưới với một số lượng con cỗ định trên trục cắt (cột)
- **SliverGridDelegateWithMaxCrossAxisExtent**: tạo lưới với một khoảng rộng tối đa trên trục cắt (cột) cho mỗi con



GridView

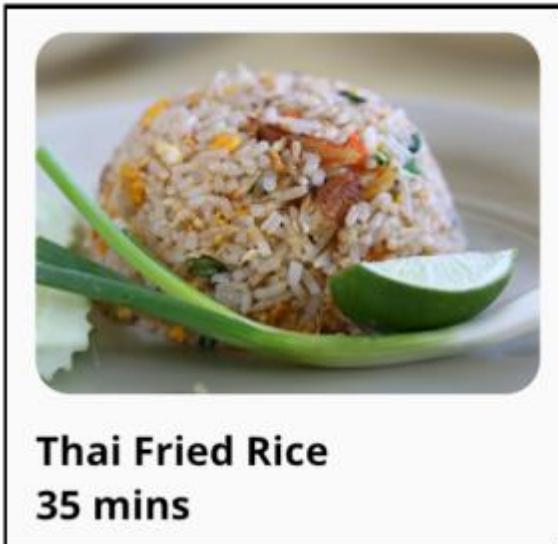
Một số tham số quan trọng:

- **crossAxisCount**: số lượng con trên trực cắt (mặc định có thể xem là số lượng cột trong lưới)
- **childAspectRatio**: tỉ số giữa khoảng rộng trên trực cắt và trực chính của mỗi con
- **crossAxisSpacing**: khoảng cách giữa các con trên trực cắt (mặc định là trực ngang)
- **mainAxisSpacing**: khoảng cách giữa các con trên trực chính (mặc định là trực đứng)

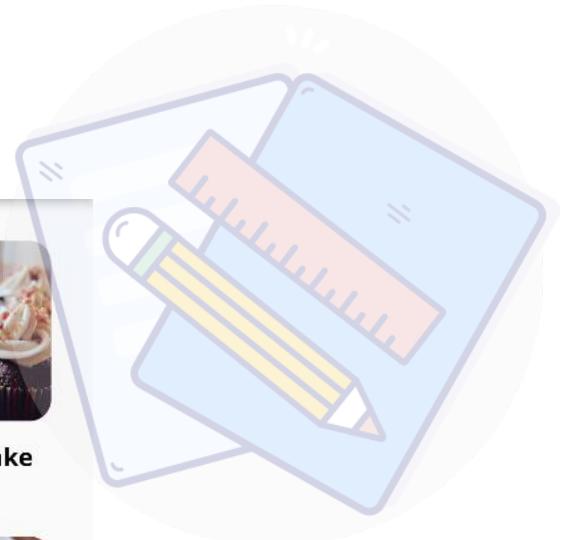
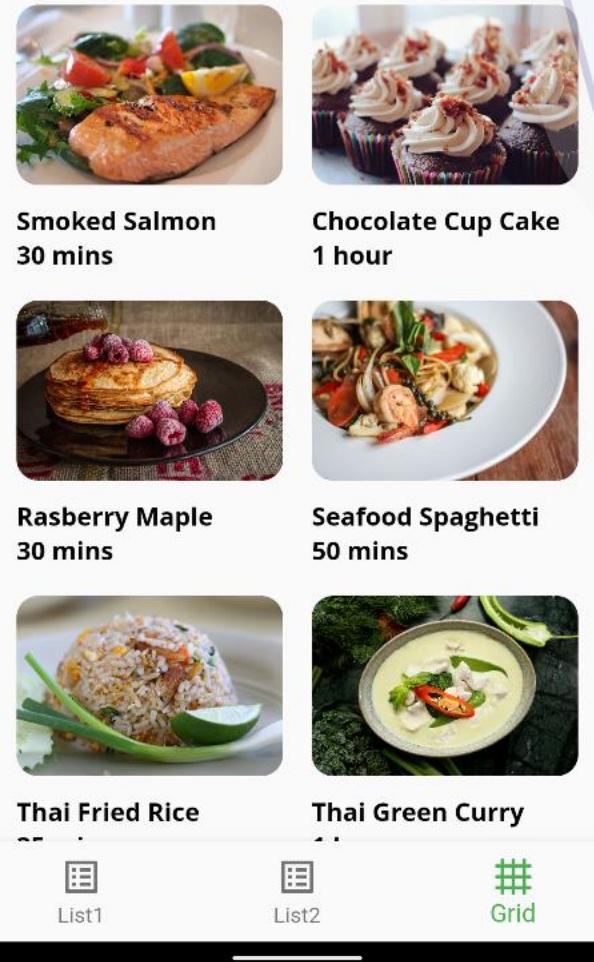


GridView

RecipeThumbnail



RecipesGridView



RecipesGridView

RecipesGridView nhận vào một danh sách các SimpleRecipe và sẽ xây dựng lưới các RecipeThumbnail tương ứng



```
class RecipesGridView extends StatelessWidget {  
    final List<SimpleRecipe> recipes;  
  
    const RecipesGridView({  
        Key? key,  
        required this.recipes,  
    }) : super(key: key);  
  
    @override  
    Widget build(BuildContext context) {  
        return ...  
    }  
}  
  
class SimpleRecipe {  
    String id;  
    String dishImage;  
    String title;  
    String duration;  
    String source;  
    List<String> information;  
  
    SimpleRecipe({  
        required this.id,  
        required this.dishImage,  
        required this.title,  
        required this.duration,  
        required this.source,  
        required this.information,  
    });  
}
```

RecipesGridView

RecipesGridView nhận vào một danh sách các SimpleRecipe và sẽ xây dựng lưới các RecipeThumbnail tương ứng

```
@override  
Widget build(BuildContext context) {  
  return Padding(  
    padding: const EdgeInsets.only(left: 16, right: 16, top: 16),  
    child: GridView.builder(  
      itemCount: recipes.length,  
      gridDelegate:  
        const SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 2),  
      itemBuilder: (context, index) {  
        final simpleRecipe = recipes[index];  
        return RecipeThumbnail(recipe: simpleRecipe);  
      },  
    ), // GridView.builder  
  ); // Padding  
}
```



RecipesGridView

GridScreen đọc dữ liệu và tạo RecipesGridView

```
class GridScreen extends StatelessWidget {
    final exploreService = MockFooderlichService();

    GridScreen({Key? key}) : super(key: key);

    @override
    Widget build(BuildContext context) {
        return FutureBuilder(
            future: exploreService.getRecipes(),
            builder: (context, AsyncSnapshot<List<SimpleRecipe>> snapshot) {
                if (snapshot.connectionState == ConnectionState.done) {
                    return RecipesGridView(recipes: snapshot.data ?? []);
                } else {
                    return const Center(child: CircularProgressIndicator());
                }
            },
        ); // FutureBuilder
    }
}
```



RecipesGridView

Hiển thị GridScreen

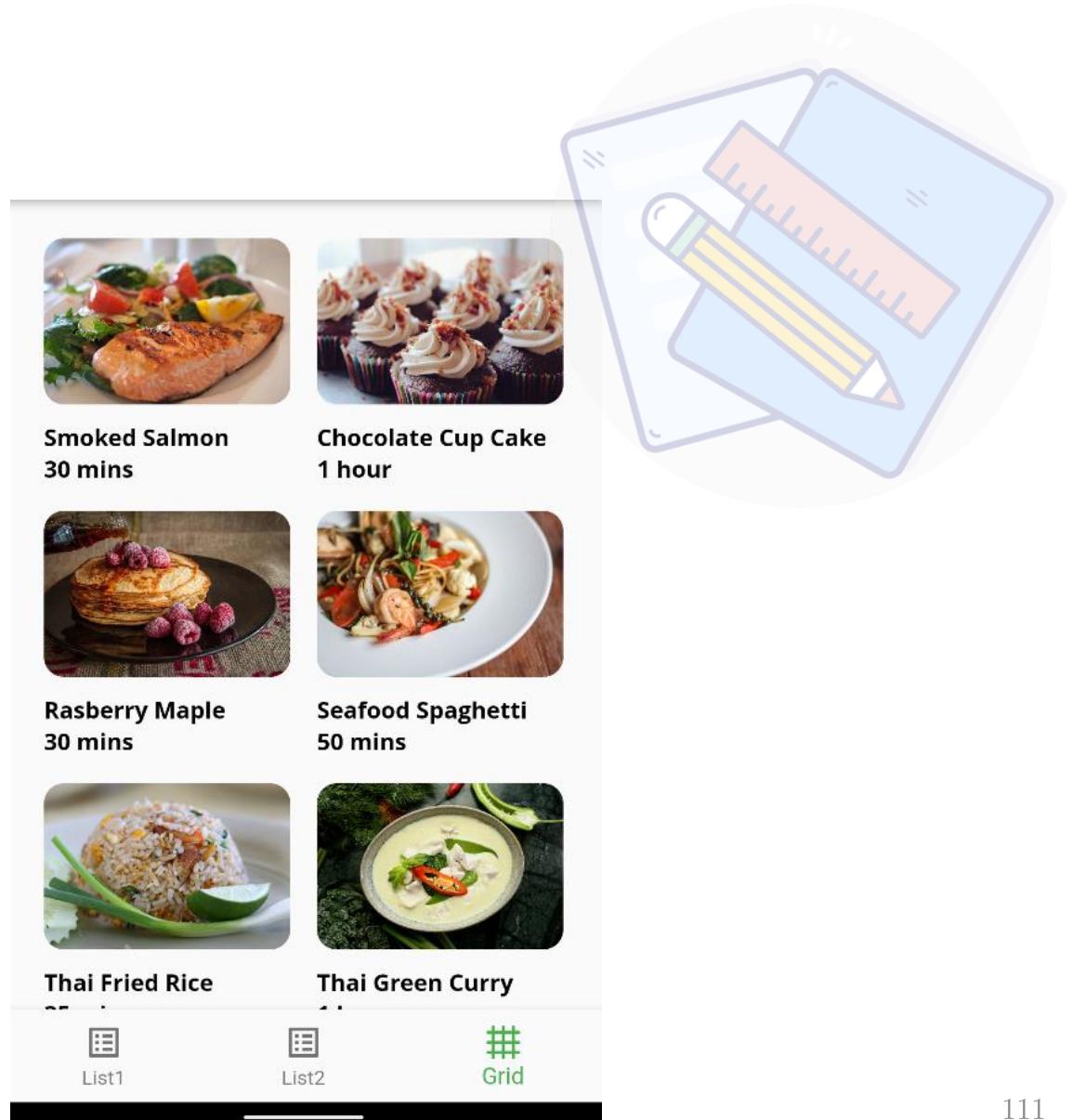
```
class Home extends StatefulWidget {
  const Home({Key? key}) : super(key: key);

  @override
  State<Home> createState() => _HomeState();
}

class _HomeState extends State<Home> {
  int _selectedIndex = 0;

  static final List<Widget> _pages = <Widget>[
    List1Screen(),
    List2Screen(),
    GridScreen(),
  ]; // <Widget>[]

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }
}
```



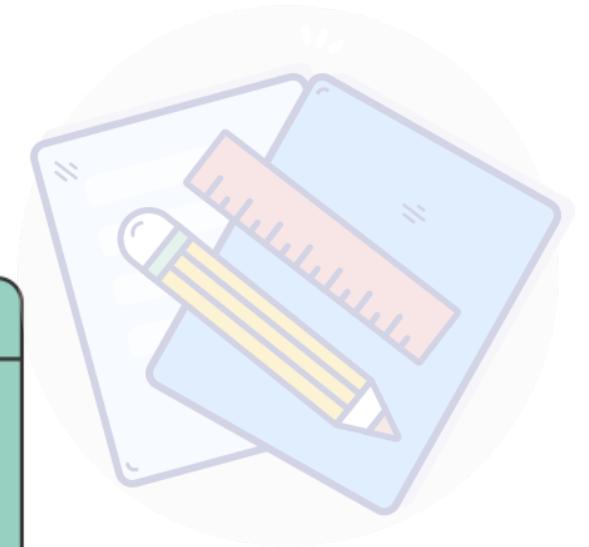
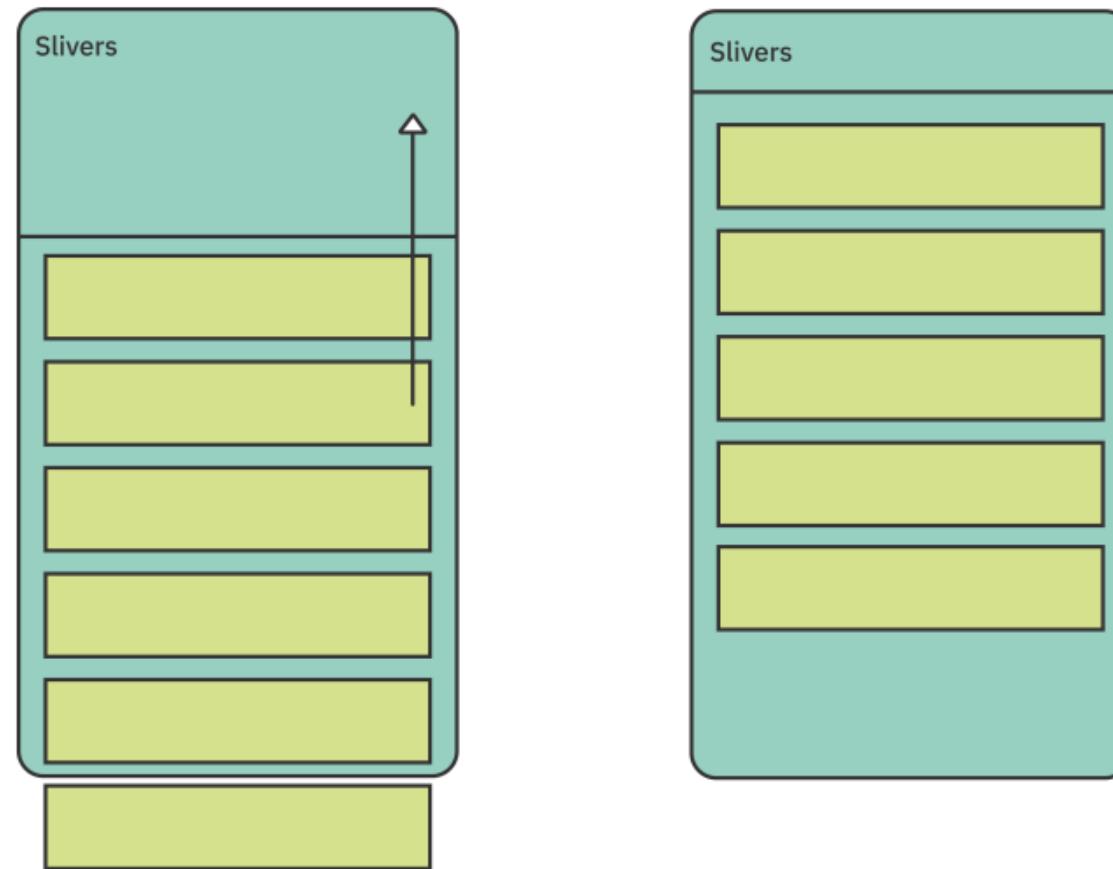
Các widget cuộn khác

- [SingleChildScrollView](#)
- [CustomScrollView](#)
- [PageView](#)
- [StaggeredGridView](#)



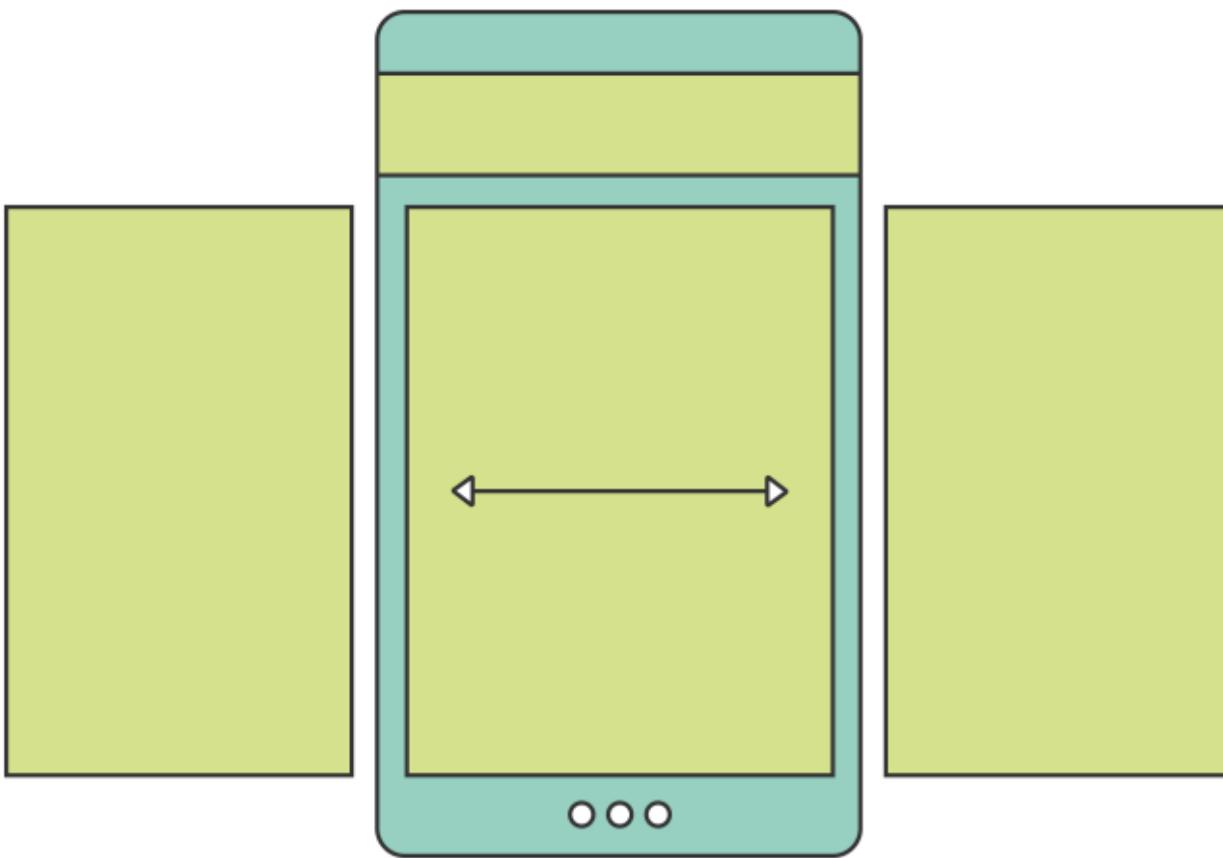
Các widget cuộn khác

CustomScrollView

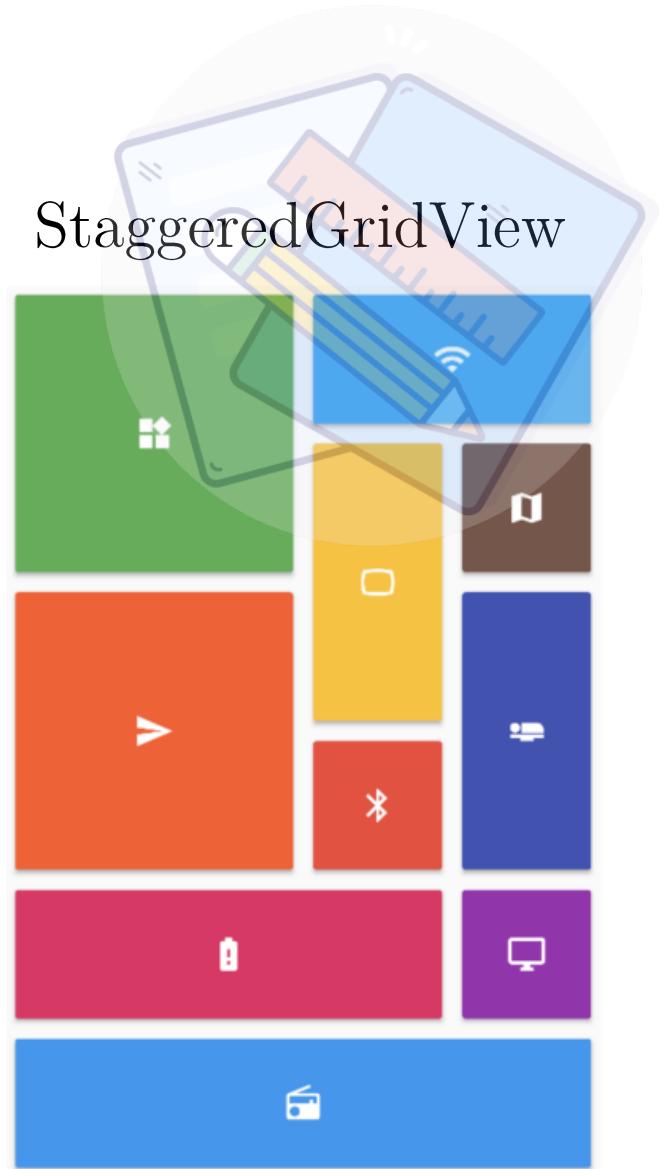


Các widget cuộn khác

PageView

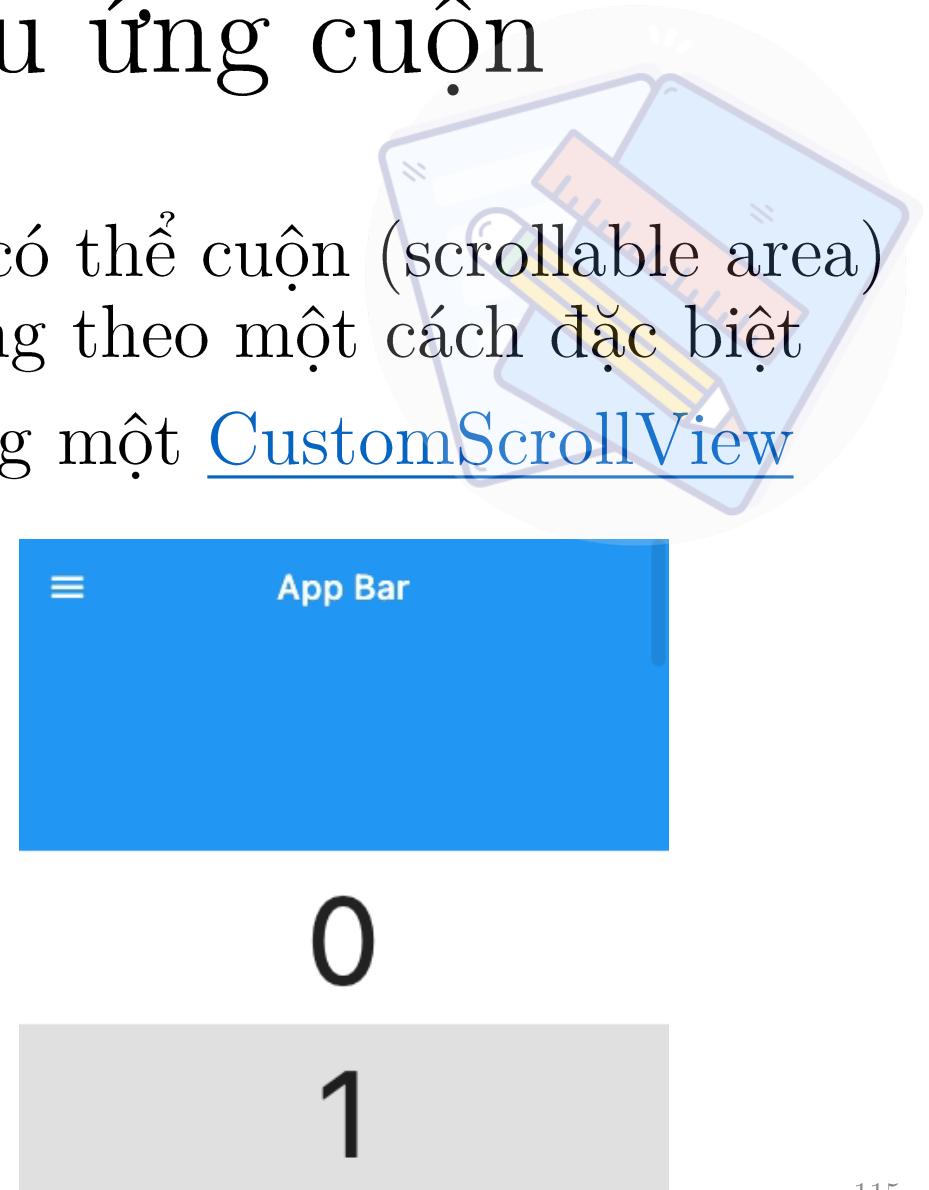


StaggeredGridView



Sử dụng sliver để tạo hiệu ứng cuộn

- Một **sliver** là một phần của khu vực có thể cuộn (scrollable area) mà bạn có thể định nghĩa để hoạt động theo một cách đặc biệt
- Các sliver thường được dùng bên trong một CustomScrollView
 - SliverAppBar
 - SliverGrid
 - SliverList



Câu hỏi?

