**FLUTTER**

# **RoadMap**

# **Install**

## **Installation the latest stable release of the Flutter SDK**

* Download Flutter SDK mới nhất từ homepage flutter
* Extract the zip file and place the contained flutter in the desired installation location for the Flutter SDK (for example, C:\src\flutter).

## **Update your path**

If you wish to run Flutter commands in the regular Windows console, take these steps to add Flutter to the PATH environment variable:

* From the Start search bar, enter ‘env’ and select **Edit environment variables for your account**.
* Under **User variables** check if there is an entry called **Path**:
  + If the entry exists, append the full path to flutter\bin using ; as a separator from existing values.
  + If the entry doesn’t exist, create a new user variable named Path with the full path to flutter\bin as its value.

## **Tại CMD or PowerShell**

Run command: flutter doctor để check các thành phần cần thiết, nếu thành phần nào báo X or ! thì cần xử lý thêm (install, update, run command…) cho đến khi nào tất check xanh là ok.

## **For Windows desktop development, you need the following in addition to the Flutter SDK:**

* [Visual Studio 2022](https://visualstudio.microsoft.com/downloads/) or [Visual Studio Build Tools 2022](https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2022) When installing Visual Studio or only the Build Tools, you need the “Desktop development with C++” workload installed for building windows, including all of its default components.

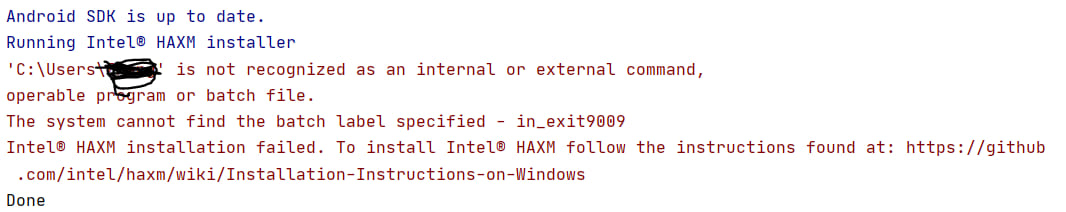
## **Agree to Android Licenses**

Before you can use Flutter, you must agree to the licenses of the Android SDK platform. This step should be done after you have installed the tools listed above.

$ flutter doctor --android-licenses

## **Install Android Studio**

* Sau khi install xong, vào SDK Manager – Tab SDK Tools
  + Cài Android SDK Command-line Tools
  + Cài Google Play Service
  + Intel x86 Emulator Acceletator (HAXM Installation)



Lỗi như hình trên – có thể do tên folder (user name) trên pc vi phạm quy tắc ko no space, no specials charactor

Flutter app can run on Android and IOS emulators with Android Studio, but ONLY on Apple machines (not Windows). So if you have Android Studio installed on Windows machine you won't have IOS emulator.

This info isn't obvious in Android Studio documentation, but you can find more details in Flutter installation guide:

macOS install (Flutter, Android Studio, emulators for Android and IOS)

Windows install (Flutter, Android Studio, emulator for Android)

So in your case the best way is to develop flutter apps on your Mac.

Setup the iOS simulator???

https://www.makeuseof.com/windows-10-install-ios-emulator/

* Tạo Virtual Device
  + Add Skin Samsung
    - Tải skin tại <https://developer.samsung.com/galaxy-emulator-skin>
    - Unzip skin file vào folder chỉ định
    - Trong màn hình Decive Manager click Create Device, New Hardwave Profile, Nhập các giá trị Screen Size và Resolution y như mô tả của skin từ <https://developer.samsung.com/galaxy-emulator-skin/galaxy-s.html> (tùy version device).

Kéo xuống dưới cùng có phần Default Skin, click … browser tới folder chứa skin

* Open AVD: Open Android Studio (khi đã xóa hết app, ko quản lý app nào), click More Action, chọn Virtual Device Manager, chọn máy ảo, click run máy ảo. Lúc này có thể close Android Studio, chỉ để lại AVD

## **Setup AVD không install Android Studio – ko khả thi**

You do not need Android Studio to create or run a virtual device. Just use sdkmanager and avdmanager from the Android SDK tools.

* ProcedureTo Install the JDK Software and Set Environment Variables JAVA\_HOME on a Windows System: <https://docs.oracle.com/cd/E19182-01/820-7851/inst_cli_jdk_javahome_t/index.html>

avdmanager create avd --name AndroidDevice01 --package "system-images;android-21;default;x86\_64"

Nơi lưu AVD: Đường dẫn mặc định: **C:\Users\<tên người dùng>\.android\avd**

## Run AVD Emulator without Android Studio

* Show list AVD trên hệ thống:

Run

cd ~/Android/Sdk/tools/bin && ./avdmanager list avd

OR

cd ~/Android/Sdk/tools && ./emulator -list-avds

Demo: C:\Users\Thinkpad T480s\./avdmanager list avd AppData\Local\Android\Sdk\tools>emulator -list-avds

PS C:\Users\Thinkpad T480s\AppData\Local\Android\Sdk\emulator> .\emulator -list-avds // cú pháp trên PS khác: phải thêm '.\'

* Copy name of the device you want to run and then: cd ~/Android/Sdk/tools && ./emulator -avd NAME\_OF\_YOUR\_DEVICE, nếu AVD nào báo lỗi 'PANIC: Missing emulator engine program for 'x86' CPU.' thì vào path ~/Android/Sdk/emulator

Demo: C:\Users\Thinkpad T480s\AppData\Local\Android\Sdk\emulator>.\emulator.exe -avd Pixel\_5\_API\_30

## **Kết nối Physical Device chưa được**

## **Smartface IDE - Develop iOS and Android apps on Windows, macOS or Linux. No more OS or hardware lock-in.**

**Install**

https://smartface.io/

**Using**

https://docs.smartface.io/smartface-getting-started/create-a-new-project

## **Install VS Code (if using VS Code)**

VS Code is a lightweight editor with complete Flutter app execution and debug support.

* Install the Flutter and Dart plugins

## **Code demo trên web**

<https://dart.dev/#try-dart>

https://dartpad.dev/?

# **VS Code**

Hot Reload – Run without Debugging – Ctrl + F5

Lệnh in debug vo Debug console: ngoài print() ra nên dùng debugPrint()

## **Setup trong Setting.json (tại path C:\Users\Thinkpad T480s\AppData\Roaming\Code\User\profiles\130c8c0a\settings.json)**

Bấm tổ hợp Ctrl +Shift + P, nhập setting.json, chọn dòng Open User Setting

* Auto fix code khi save:
* "editor.codeActionsOnSave": {"source.fixAll": true},
* Show link cấp các widget:

    "dart.previewFlutterUiGuides": true,

## **Định dạng code**

* Dấu ',' sau phần tử cuối cùng trong câu lệnh: nếu bỏ thì code phía sau nó sẽ dồn lên cùng dòng; nếu nhập thì sẽ xuống dòng tại vị trí đó

## **Xử Lý Lỗi**

* Run flutter doctor –vv hoặc các lệnh flutter từ cmd báo Error: Unable to find git in your PATH.

Chạy cmd: git config --global --add safe.directory '\*'

## **Using the Flutter inspector**

<https://docs.flutter.dev/tools/devtools/inspector#get-started>

### **Lỗi không show object lên giao diện:**

Một số nguyên nhân sau:

* Dò giá trị Height/width thấy bằng 0 (có thể do setup default của các widget) - setup lại cho phù hợp

## **Flutter Project Structure**

**Android – Nơi chứa code Android – viết Android Native code**

**Ios – Nơi chứa code Ios – viết Ios Native code**

**Lib – Nơi chứa toàn bộ code flutter (file .dart)**

**Test – Nơi viết test code**

**Pubspec.yaml**

Mỗi dự án Flutter bao gồm file pubspec.yaml. Nó nằm ở đầu cây dự án và chứa siêu dữ liệu về dự án mà công cụ Dart và Flutter cần biết. Pubspec được viết bằng YAML , mà con người có thể đọc được, nhưng hãy lưu ý rằng khoảng trắng (tab v dấu cách) mới quan trọng .

Tệp pubspec chỉ định các phụ thuộc mà dự án yêu cầu, chẳng hạn như các gói cụ thể (và phiên bản của chúng), phông chữ hoặc tệp hình ảnh. Nó cũng chỉ định các yêu cầu khác, chẳng hạn như sự phụ thuộc vào gói dành cho nhà phát triển (như gói thử nghiệm hoặc mô phỏng) hoặc các ràng buộc cụ thể đối với phiên bản SDK Flutter.

**Pubspec.lock**

Lần đầu tiên bạn xây dựng dự án của mình, nó sẽ tạo file pubspec.lock chứa phiên bản cụ thể của các gói được bao gồm. Điều này đảm bảo rằng bạn sẽ nhận được cùng một phiên bản vào lần tiếp theo dự án được xây dựng.

**analysis\_options.yaml**

**YAML is sensitive to whitespace.** Don’t use tabs in a YAML file, and use 2 spaces to denote each level of indentation.

Phân tích tĩnh cho phép tìm ra các vấn đề trước khi thực thi một dòng mã duy nhất. Đó là một công cụ mạnh mẽ được sử dụng để ngăn chặn lỗi và đảm bảo mã tuân thủ các hướng dẫn về kiểu dáng.Với sự trợ giúp của bộ phân tích, bạn có thể tìm ra những lỗi đơn giản như viết sai chính tả. Ví dụ, có thể đã có một dấu chấm phẩy không cần thiết bị đưa vào câu lệnh if một cách tình cờ

Rules được gán trong file này

## Shortcut

* Show Widget Tree (gồm các function, class nhưng ko bao gồm các properties): Ctrl + Shift + P 🡪 View: Show Flutter

Với các widget có icon  ta có thể click chọn, right click để thực hiện 1 số thao tác trên nó

* Tạo StatelessWidget: type 'stl' or 'stless'
* Tạo StatefulWidget: type 'stf' or 'stful'

## Check Version & Upgrade

Open Terminal / Command Prompt (Windows),

Check version: flutter --version

Upgrade: flutter upgrade

# **DART**

<https://dart.dev/tools/dart-devtools#1-start-the-target-app>

<https://dart.dev/tools/linter-rules/prefer_interpolation_to_compose_strings>

## Package trong Dart

Theo định nghĩa, một package trong Dart là một bộ sưu tập các tập tin Dart liên quan và được tổ chức theo một cấu trúc cụ thể¹. Một package có thể chứa các hàm, lớp, biến, kiểu dữ liệu, widget, plugin, hoặc bất kỳ đoạn code Dart nào có thể được tái sử dụng và chia sẻ. Một package cũng có thể phụ thuộc vào các package khác để sử dụng các chức năng của chúng.

Một package trong Dart có thể được phân loại thành ba loại chính²:

- Package Dart: là những package có thể được sử dụng ở cả môi trường web lẫn thiết bị di động. Ví dụ như [english\_words] là một package chứa khoảng 5000 từ tiếng Anh và có các chức năng cơ bản như danh từ, âm tiết, v.v.

- Package Flutter: là những package phụ thuộc vào framework Flutter và chỉ có thể được sử dụng trong môi trường thiết bị di động. Ví dụ như [fluro] là một custom router cho Flutter.

- Plugin Flutter: là những package phụ thuộc vào framework Flutter cũng như nền tảng cơ bản (Android SDK hoặc iOS SDK). Ví dụ như [camera] là một plugin dùng để tương tác với thiết bị camera.

Để sử dụng một package trong Dart, bạn cần phải thực hiện các bước sau³:

- Tìm kiếm và chọn một package phù hợp với nhu cầu của bạn trên [pub.dev](^4^), một máy chủ trực tuyến để lưu trữ và xuất bản các package Dart.

- Nhập tên và phiên bản của package vào file pubspec.yaml của dự án của bạn, trong phần dependencies. Ví dụ:

dependencies:

english\_words: ^3.1.5

- Chạy lệnh `flutter pub get` hoặc `dart pub get` để cài đặt package vào dự án của bạn.

- Nhập file Dart chính của package vào file code của bạn bằng lệnh `import`. Ví dụ:

import 'package:english\_words/english\_words.dart';

- Sử dụng các hàm, lớp, biến, hoặc widget của package theo ý muốn.

# **FLUTTER**

## Widget

Flutter là một framework phát triển ứng dụng di động và web sử dụng ngôn ngữ lập trình Dart. Trong Flutter, giao diện người dùng được xây dựng bằng cách kết hợp các widget. Widget là các thành phần cơ bản trong Flutter, và chúng được sắp xếp lại để tạo thành giao diện người dùng. Dưới đây là một số widget quan trọng trong Flutter:

* StatelessWidget: Đây là một widget không thay đổi trạng thái sau khi được xây dựng ban đầu. Nó được sử dụng cho các thành phần tĩnh của giao diện người dùng.
* StatefulWidget: Đây là một widget có thể thay đổi trạng thái sau khi được xây dựng. Nó được sử dụng cho các thành phần động của giao diện người dùng.
* MaterialApp là widget kế thừa các kiến trúc theo chuẩn design marterial (xem như html tag trong trang web)
* Scaffold: Widget này cung cấp cấu trúc cơ bản cho một màn hình, bao gồm AppBar, body và các thành phần khác. Scaffold là widget bao chứa toàn bộ code chúng ta vào trong nó (phủ đầy giao diện chúng ta, như thẻ body của html vậy thôi)
* AppBar: Widget này tạo một thanh đầu của ứng dụng, thường chứa tiêu đề, các biểu tượng thao tác và nút điều hướng.
* Container: Widget này tạo một hình chữ nhật có thể chứa các widget con và được sử dụng để tạo khoảng trống, định dạng và cài đặt kiểu cho các widget con.
* Text: Widget này hiển thị văn bản trên giao diện người dùng. Bạn có thể định dạng văn bản bằng cách sử dụng thuộc tính style.
* Image: Widget này hiển thị hình ảnh từ các nguồn như asset hoặc mạng.
* Column và Row: Đây là các widget chứa các widget con theo chiều dọc (Column) hoặc chiều ngang (Row). Chúng giúp bạn sắp xếp các widget con một cách linh hoạt.
* ListView: Widget này cho phép bạn tạo danh sách cuộn được tạo bằng cách xếp các widget con dọc theo trục chính.
* FlatButton, RaisedButton, IconButton: Các widget này tạo các nút tương tác, cho phép người dùng thực hiện các hành động khi nhấp vào.
* TextField: Widget này cho phép người dùng nhập dữ liệu văn bản.
* InkWell: Widget này tạo ra một vùng có hiệu ứng mực khi người dùng chạm vào, giúp tạo ra giao diện tương tác thú vị.
* GestureDetector: Widget này cho phép bạn bắt các sự kiện tương tác như chạm, vuốt, nhấn giữ và nhiều thứ khác.

Đây chỉ là một số ví dụ cơ bản về các widget trong Flutter. Flutter cung cấp một loạt các widget khác nhau để bạn có thể xây dựng các giao diện người dùng phong phú và linh hoạt.

## StatelessWidget & StatefulWidget

Trong Flutter, tất cả các widget đều thuộc một trong hai loại: StatelessWidget hoặc StatefulWidget.

**StatelessWidget:** Đây là loại widget mà trạng thái của nó không thay đổi sau khi được xây dựng ban đầu. Điều này có nghĩa là nếu bạn muốn thay đổi giao diện của widget này, bạn phải xây dựng một phiên bản mới của nó với dữ liệu mới. StatelessWidget thường được sử dụng cho các thành phần giao diện người dùng tĩnh và không thay đổi, như văn bản hiển thị, hình ảnh, biểu đồ, vv.

Hiểu nôm na là Widget tĩnh và nó không thể tự thay đổi được những gì mà nó hiển thị sau khi đã được Render xong.

Widget này cần 1 hàm Widget build(BuildContext context) để render dữ liệu lên màn hình. Hàm build chỉ được gọi 1 lần khi ứng dụng đang hoạt động, như vậy thì dữ liệu chỉ được render 1 và không thay đổi suốt quá trình sử dụng ứng dụng. Dữ liệu hiển thị bạn có thể hard code hoặc truyền thông qua hàm Constructors của class và dữ liệu này sẽ không thay đổi suốt quá trình hiển thị trên màn hình.

*Vậy đặt ra vấn đề Tôi muốn thay đổi text đang hiển thị trên màn hình thì làm thế nào?*

Tuy StatelessWidget không thể tự thay đổi được chính nó, nhưng khi Widget cha thay đổi thì StatelessWidget sẽ được khởi tạo lại.

Example:

class MyWidget extends StatelessWidget {

  final bool load\_state;

  const MyWidget(this.load\_state);

  @override

  Widget build(BuildContext context) { // render data output

    return load\_state

        ? Text('loading state $load\_state')

        : const CircularProgressIndicator();

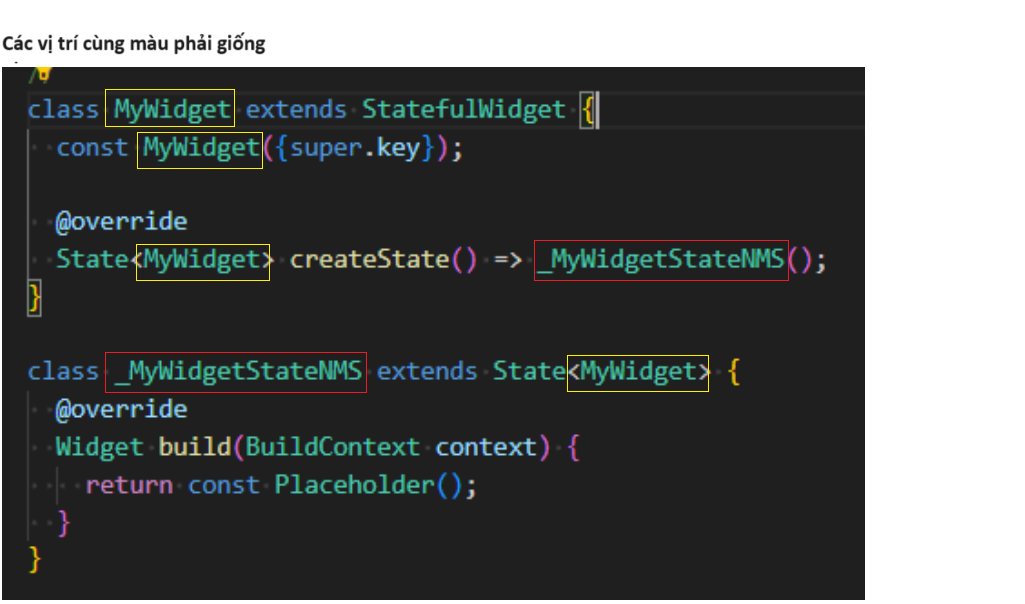
  }

}

**StatefulWidget:** Đây là loại widget có trạng thái có thể thay đổi sau khi được xây dựng ban đầu. StatefulWidget được sử dụng cho các thành phần giao diện người dùng có thể tương tác và thay đổi dựa trên tương tác người dùng, thời gian hoặc dữ liệu thay đổi.

Khác với StatelessWidget thì StatefulWidget là 1 Widget động và nó có thể thay đổi những gì đang hiển thị bằng cách thay đổi State của chính nó.

Cấu trúc StatefulWidget



* MyWidget là một widget con của StatefulWidget. Bằng cách kế thừa StatefulWidget, MyWidget cho phép bạn tạo một widget có khả năng thay đổi trạng thái.
* State<MyWidget> createState() => \_MyWidgetState(); - Tạo trạng thái cho MyWidget:

Hàm createState được ghi đè để tạo ra một đối tượng trạng thái tương ứng cho MyWidget. Trong trường hợp này, đối tượng trạng thái được tạo là \_MyWidgetState.

* \_MyWidgetState là một lớp con của State và chịu trách nhiệm quản lý trạng thái của MyWidget. Class \_ MyWidgetState sẽ overrides phương thức Widget build(BuildContext context) hàm này trả về Widget. Đây là nơi bạn định nghĩa UI mà class hiển thị. StatefulWidget quản lý trạng thái UI thông qua State, khi State thay đổi thì StatefulWidget sẽ render lại UI mà nó đang hiển thị.

Vậy câu hỏi đặt ra là Làm thế nào tôi có thể thay đổi được State của Statefull hay đơn giản là update dữ liệu đang hiển thị trên màn hình?

Câu trả lời này khá đơn giản. StatefulWidget cung cấp phương thức setState() để bạn có thể thay đổi State của class (chỉ có StatefulWidget) mới có setState()). Hiểu đơn giản là khi bạn muốn Update UI của StatefulWidget thì cần gọi phương thức setState() để thông báo cho StatefullWidget là tôi muốn bạn UpdateUI. Tất nhiên bạn có thể gọi setState() nhiều lần mỗi khi cần thay đổi UI trong vòng đời của ứng dụng.

\*Bản thân câu lệnh

setState(() {}); là thực hiện việc update UI.

Các lệnh được gọi trong thân nó là tác vụ tuỳ chọn theo các mục đích khác.

Ví dụ:

setState(() { // update value on UI and value of \_inputValue

                    \_inputValue = value;

                  });

Và đặc biệt, nên sử dụng StatefulWidget trong Widget con của ListView để các item của chúng có thể tự động cập nhật trạng thái mà không cần cả ListView thay đổi trạng thái.

Nhìn như này mình đoán sẽ nhiều bạn đặt câu hỏi Thế tại sao không dùng StatefulWidget vừa có thể hiển thị vừa có thể thay đổi dc UI việc gì phải dùng StatelessWidget. Thì đúng là như thế, nhưng các bạn lưu ý do StatelessWidget không có State nên việc render UI của nó nhẹ hơn và nhanh hơn rất nhiều. Các bạn cần tìm hiểu rõ về State để lựa chọn sử dụng giữa StatelessWidget và StatefulWidget để có thể dễ dàng trong việc viết code và tối ưu hiệu năng.

Example: 01

// MyWidgetFul extends StatefulWidget call method createState() return kieu State<StatefulWidget> cua MyWidgetFul\_state

class MyWidgetFul extends StatefulWidget {

  final bool loading; // Fields in a Widget subclass are always marked "final".

  const MyWidgetFul(this.loading);

  @override

  State<StatefulWidget> createState() {

    return MyWidgetFul\_state();

  }

}

// MyWidgetFul\_state lai extends nguoc ve State<MyWidgetFul>, call method build – render data output

class MyWidgetFul\_state extends State<MyWidgetFul> {

  late bool \_local\_load\_state;

  @override

  void initState() {

    // khoi tao gia tri ban dau, run truoc build function

    \_local\_load\_state = widget.loading;

  }

  @override

  Widget build(BuildContext context) { // This method is rerun every time setState method is called

    return \_local\_load\_state

        ? Text('StatefulWidget')

        : CircularProgressIndicator();

  }

}

Example: 02

class MyHomePage extends StatefulWidget {

  const MyHomePage({super.key, required this.title});

  final String title; // Fields in a Widget subclass are always marked "final".

  @override

  State<MyHomePage> createState() => \_MyHomePageState();

}

class \_MyHomePageState extends State<MyHomePage> {

  int \_counter = 0;

  void \_incrementCounter() {

    setState(() { // setState thông báo cho Flutter rằng có một thay đổi đã xảy ra trong State này, dẫn đến run lại phương thức build ở dưới để cập nhật hiển thị dựa trên các giá trị đã được cập nhật. Nếu chúng ta thay đổi giá trị của \_counter mà không gọi setState(), thì phương thức build sẽ ko run lại, do đó không có gì thay đổi trên giao diện hiển thị.

      \_counter++;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        backgroundColor: Theme.of(context).colorScheme.inversePrimary,

        title: Text(widget.title),

      ),

      body: Center(

        child: Column(

          children: <Widget>[

            const Text(

              'You have pushed the button this many times:',

            ),

            Text(

              '$\_counter',

              style: Theme.of(context).textTheme.headlineMedium,

            ),

          ],

        ),

      ),

      floatingActionButton: FloatingActionButton(

        onPressed: \_incrementCounter,

        tooltip: 'Increment',

        child: const Icon(Icons.add),

      ), // This trailing comma makes auto-formatting nicer for build methods.

    );

  }

## Truyền tham số cho widget

1. Khai báo variable làm tham số, định nghĩa hàm constructor với tham số

class PaddingAndMargin extends StatelessWidget {

// khai báo variable làm tham số, định nghĩa hàm constructor với tham số

  double pixel = 0;

  PaddingAndMargin(this.pixel);

  @override

  Widget build(BuildContext context) {

    return Card(

      margin: EdgeInsets.all(20.0),

      color: Colors.amber,

      child: Padding(

padding: EdgeInsets.symmetric(horizontal: pixel), // sử dụng tham số được truyền vào

child: Text('Flutter'),

      ),

    );

  }

}

//-----------------------------

1. Trong hàm main(): gọi hàm constructor PaddingAndMargin đã định nghĩa ở bước 1

 body: PaddingAndMargin(20),

## Quy tắc cơ bản khi dùng widget

* Chia nhỏ các tác vụ vào widget
* Tận dụng tối đa các widget build-in như Container, Text, Row, Column,…
* Tránh lồng nhiều lớp widget
* Dùng const khi có thể cho các object
* Áp dụng Material Design và Cupertino
* Tránh logic phức tạp trong build:

Hạn chế việc thực hiện các phép tính phức tạp, truy vấn cơ sở dữ liệu, gọi API mạng hay các xử lý lớn khác trong hàm build của widget. Hàm build được gọi lại nhiều lần trong quá trình render giao diện và có thể gây lag nếu chứa các thao tác tốn nhiều thời gian.

Việc tính toán phức tạp trong hàm build có thể làm chậm quá trình render giao diện, làm cho ứng dụng trở nên không mượt và không đáp ứng tốt, dẫn đến trải nghiệm người dùng kém.

Thay vì thực hiện logic phức tạp trong build, bạn nên thực hiện những việc như tính toán, truy vấn dữ liệu, gọi API mạng, và các xử lý phức tạp khác trước khi đến phần xây dựng giao diện. Kết quả của các xử lý này nên được lưu trong trạng thái của widget hoặc các biến tạm thời, và sau đó bạn chỉ cần sử dụng các giá trị đã tính toán trong hàm build.

Ví dụ, nếu bạn cần lấy dữ liệu từ cơ sở dữ liệu, hãy thực hiện việc này trong phương thức initState của StatefulWidget và lưu kết quả trong trạng thái. Sau đó, trong hàm build, bạn chỉ cần sử dụng dữ liệu đã lấy từ trạng thái mà không cần phải thực hiện truy vấn cơ sở dữ liệu trong mỗi lần render.

class MyWidget extends StatefulWidget {

  @override

  \_MyWidgetState createState() => \_MyWidgetState();

}

class \_MyWidgetState extends State<MyWidget> {

  List<Data> fetchedData = []; // lưu data lấy về từ method initState()

  @override

  void initState() {

    super.initState();

    fetchDataFromDatabase();

  }

  void fetchDataFromDatabase() async {

    // Logic lấy dữ liệu từ cơ sở dữ liệu

    // và lưu vào fetchedData

  }

  @override

  Widget build(BuildContext context) {

    return ListView.builder(

      itemCount: fetchedData.length,

      itemBuilder: (context, index) {

        // Sử dụng fetchedData[index] để hiển thị dữ liệu

      },

    );

  }

}

* Thuộc tính child: nên đặt sau cùng trong list thuộc tính
* Định nghĩa các widget thành phần, gán vào biến bên ngoài, truyền vào widget cha bằng các biến này.

Định nghĩa, gán widget thành phần vào biến trước

* var stars = Row(
* mainAxisSize: MainAxisSize.min,
* children: [
* Icon(Icons.star, color: Colors.green[500]),
* Icon(Icons.star, color: Colors.green[500]),
* Icon(Icons.star, color: Colors.green[500]),
* const Icon(Icons.star, color: Colors.black),
* const Icon(Icons.star, color: Colors.black),
* ],
* );
* final ratings = Container(
* padding: const EdgeInsets.all(20),
* child: Row(
* mainAxisAlignment: MainAxisAlignment.spaceEvenly,
* children: [
* stars, // truyền biến starts ở trên vào đây
* const Text(
* '170 Reviews',
* style: TextStyle(…..
* ---------------------------------
* final leftColumn = Container(
* padding: const EdgeInsets.fromLTRB(20, 30, 20, 20),
* child: Column(
* children: [
* titleText,
* subTitle,
* ratings, // truyền biến ratings ở trên vào đây
* iconList,
* ],
* ),
* );

………………

## Quy tắc thiết kế layout – Rất quan trọng

https://docs.flutter.dev/ui/layout/constraints

"Constraints go down. Sizes go up. Parent sets position." là một nguyên tắc quan trọng trong Flutter khi bạn đang thiết kế giao diện. Nó có nghĩa:

"Constraints go down": Điều này đề cập đến việc các ràng buộc (constraints) về kích thước và vị trí của các widget con thường được chuyển từ widget cha xuống. Điều này có thể bao gồm giới hạn về kích thước tối đa, tối thiểu và các quy tắc khác liên quan đến vị trí.

"Sizes go up": Khi widget con được xác định về kích thước và vị trí dựa trên các ràng buộc từ widget cha, chúng thường cố gắng tăng kích thước của chính mình để phù hợp với ràng buộc đó. Điều này đảm bảo rằng các widget con có đủ không gian để hiển thị nội dung của họ.

"Parent sets position": Widget cha thường định vị và quyết định vị trí của các widget con trong giao diện. Widget con không tự do quyết định vị trí của mình mà phải tuân theo chỉ đạo của widget cha.

Tóm lại, nguyên tắc này giúp duy trì sự sắp xếp và kích thước chính xác của các phần tử trong giao diện Flutter bằng cách đảm bảo rằng các widget con hiểu và tuân theo các ràng buộc và chỉ đạo từ widget cha.

## MaterialApp class

MediaQuery

In Flutter, MediaQuery is a utility class that provides information about the current media or screen properties. It allows you to retrieve information such as the device's screen size, pixel density, orientation, and more. The primary purpose of MediaQuery is to help you create responsive and adaptive layouts by allowing your widgets to adapt their behavior based on the characteristics of the device.

Here are some common use cases and purposes of MediaQuery in Flutter:

* Responsive Layouts: You can use MediaQuery to obtain the device's screen size and adjust your widget layout accordingly. For example, you might want to display a different number of columns in a grid or change the font size based on the screen size.
* Orientation Handling: You can determine the current device orientation (portrait or landscape) using MediaQuery. This information is useful for adjusting the layout or behavior of your widgets based on the orientation.
* Text Scaling: You can use the MediaQuery.textScaleFactor property to adjust text size based on the user's system-wide text scaling settings.
* Platform-specific Adaptations: MediaQuery allows you to check the current platform (iOS or Android) and apply platform-specific adaptations to your UI.

## LayoutBuilder

Builds a widget tree that can depend on the parent widget's size.

Similar to the [Builder](https://api.flutter.dev/flutter/widgets/Builder-class.html) widget except that the framework calls the [builder](https://api.flutter.dev/flutter/widgets/ConstrainedLayoutBuilder/builder.html) function at layout time and provides the parent widget's constraints. This is useful when the parent constrains the child's size and doesn't depend on the child's intrinsic size. The [LayoutBuilder](https://api.flutter.dev/flutter/widgets/LayoutBuilder-class.html)'s final size will match its child's size.

If the child should be smaller than the parent, consider wrapping the child in an [Align](https://api.flutter.dev/flutter/widgets/Align-class.html) widget. If the child might want to be bigger, consider wrapping it in a [SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html) or [OverflowBox](https://api.flutter.dev/flutter/widgets/OverflowBox-class.html).

 It has two parameters. build context and Boxconstrant. BuildContext refers to a widget. But box constraint is more important, it gives the width to the parent widget which is used to manage the child according to the size of the parent.

Note: The main difference between **Media Query** and **LayoutBuilder** is that Media Query uses the full context of the screen instead of the size of your particular widget. While the layout builder can determine the maximum width and height of any widget.

[LayoutBuilder](https://api.flutter.dev/flutter/widgets/LayoutBuilder/LayoutBuilder.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) builder([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [BoxConstraints](https://api.flutter.dev/flutter/rendering/BoxConstraints-class.html) constraints)})

Example

//---------------------------

class LayoutBuilder01 extends StatelessWidget {

  const LayoutBuilder01({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: const Text('LayoutBuilder01 Example')),

      body: LayoutBuilder(

        builder: (BuildContext context, BoxConstraints constraints) {

          if (constraints.maxWidth > 600) {

            return \_buildWideContainers();

          } else {

            return \_buildNormalContainer();

          }

        },

      ),

    );

  }

  Widget \_buildNormalContainer() {

    return Center(

      child: Container(

        height: 100.0,

        width: 100.0,

        color: Colors.red,

        child: const Text('Normal Container - constraints.maxWidth <= 600'),

      ),

    );

  }

  Widget \_buildWideContainers() {

    return Center(

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceEvenly,

        children: <Widget>[

          const Text('Wide Containers - constraints.maxWidth > 600'),

          Container(

            height: 100.0,

            width: 100.0,

            color: Colors.red,

          ),

          Container(

            height: 100.0,

            width: 100.0,

            color: Colors.yellow,

          ),

        ],

      ),

    );

  }

}

// Demo using MediaQuery &

class LayoutBuilder02 extends StatefulWidget {

  const LayoutBuilder02({super.key});

  @override

  State<LayoutBuilder02> createState() => \_LayoutBuilder02State();

}

class \_LayoutBuilder02State extends State<LayoutBuilder02> {

  @override

  Widget build(BuildContext context) {

    return SafeArea(

      child: Scaffold(

        appBar: AppBar(

          title: const Text('LayoutBuilder02 Demo'),

          centerTitle: true,

        ),

        body: Container(

          /// Giving dimensions to parent Container

          /// using MediaQuery

          height: MediaQuery.of(context).size.height,

          width: MediaQuery.of(context).size.width,

          alignment: Alignment.center,

          child: LayoutBuilder(

            builder: (BuildContext ctx, BoxConstraints constraints) {

              // if the screen width >= 480 i.e Wide Screen

              if (constraints.maxWidth >= 480) {

                return Row(

                  mainAxisAlignment: MainAxisAlignment.spaceBetween,

                  children: <Widget>[

                    Container(

                      padding: const EdgeInsets.symmetric(horizontal: 8),

                      alignment: Alignment.center,

                      height: constraints.maxHeight \* 0.3,

                      margin: const EdgeInsets.fromLTRB(10.0, 0.0, 10.0, 0.0),

                      decoration: const BoxDecoration(

                        color: Colors.red,

                        borderRadius: BorderRadius.all(Radius.circular(10)),

                      ),

                      child: const Text(

                        'Left Wide Screen',

                        style: TextStyle(

                          fontSize: 14,

                          fontWeight: FontWeight.bold,

                          color: Colors.white,

                        ),

                      ),

                    ),

                    Container(

                      padding: const EdgeInsets.symmetric(horizontal: 8),

                      alignment: Alignment.center,

                      height: constraints.maxHeight \* 0.3,

                      margin: const EdgeInsets.fromLTRB(10.0, 0.0, 10.0, 0.0),

                      decoration: const BoxDecoration(

                        color: Colors.orange,

                        borderRadius: BorderRadius.all(Radius.circular(10)),

                      ),

                      child: const Text(

                        'Right Wide Screen',

                        style: TextStyle(

                          fontSize: 14,

                          fontWeight: FontWeight.bold,

                          color: Colors.white,

                        ),

                      ),

                    ),

                  ],

                );

                // If screen size is < 480

              } else {

                return Container(

                  alignment: Alignment.center,

                  height: constraints.maxHeight \* 0.3,

                  margin: const EdgeInsets.fromLTRB(10.0, 0.0, 10.0, 0.0),

                  decoration: const BoxDecoration(

                    color: Colors.cyan,

                    borderRadius: BorderRadius.all(Radius.circular(10)),

                  ),

                  child: const Text(

                    'Normal Screen',

                    style: TextStyle(

                      fontSize: 18,

                      fontWeight: FontWeight.bold,

                      color: Colors.white,

                    ),

                  ),

                );

              }

            },

          ),

        ),

      ),

    );

  }

}

// LayoutBuilder03 - constraints.maxHeight/maxWidth

class LayoutBuilder03 extends StatelessWidget {

  const LayoutBuilder03({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Scaffold(body: LayoutBuilder(builder: (context, constraints) {

      // Large screens (tablet on landscape mode, desktop, TV)

      if (constraints.maxWidth > 600) {

        return Row(

          children: [

            Column(

              children: [

                Container(

                  height: constraints.maxHeight \* 0.2,

                  width: constraints.maxWidth \* 0.75,

                  color: Colors.blueAccent,

                  child: const Center(

                    child: Text('Header'),

                  ),

                ),

                Container(

                  height: constraints.maxHeight \* 0.6,

                  width: constraints.maxWidth \* 0.75,

                  color: Colors.amber,

                  child: const Center(

                    child: Text('Main Content'),

                  ),

                ),

                Container(

                  height: constraints.maxHeight \* 0.2,

                  width: constraints.maxWidth \* 0.75,

                  color: Colors.lightGreen,

                  child: const Center(

                    child: Text('Footer'),

                  ),

                ),

              ],

            ),

            Container(

              width: constraints.maxWidth \* 0.25,

              height: constraints.maxHeight,

              color: Colors.pink,

              child: const Center(

                child: Text('Right Sidebar'),

              ),

            ),

          ],

        );

      }

      // Samll screens

      return Column(

        children: [

          Container(

            height: constraints.maxHeight \* 0.2,

            color: Colors.blue,

            child: const Center(

              child: Text('Header'),

            ),

          ),

          Container(

            height: constraints.maxHeight \* 0.6,

            color: Colors.amber,

            child: const Center(

              child: Text('Main Content'),

            ),

          ),

          Container(

            height: constraints.maxHeight \* 0.2,

            color: Colors.lightGreen,

            child: const Center(

              child: Text('Footer'),

            ),

          )

        ],

      );

    }));

  }

}

## Scaffold class

[Scaffold](https://api.flutter.dev/flutter/material/Scaffold/Scaffold.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [PreferredSizeWidget](https://api.flutter.dev/flutter/widgets/PreferredSizeWidget-class.html)? appBar, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? body, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? floatingActionButton, [FloatingActionButtonLocation](https://api.flutter.dev/flutter/material/FloatingActionButtonLocation-class.html)? floatingActionButtonLocation, [FloatingActionButtonAnimator](https://api.flutter.dev/flutter/material/FloatingActionButtonAnimator-class.html)? floatingActionButtonAnimator, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)>? persistentFooterButtons, [AlignmentDirectional](https://api.flutter.dev/flutter/painting/AlignmentDirectional-class.html) persistentFooterAlignment = AlignmentDirectional.centerEnd, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? drawer, [DrawerCallback](https://api.flutter.dev/flutter/material/DrawerCallback.html)? onDrawerChanged, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? endDrawer, [DrawerCallback](https://api.flutter.dev/flutter/material/DrawerCallback.html)? onEndDrawerChanged, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? bottomNavigationBar, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? bottomSheet, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? resizeToAvoidBottomInset, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) primary = true, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) drawerDragStartBehavior = DragStartBehavior.start, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) extendBody = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) extendBodyBehindAppBar = false, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? drawerScrimColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? drawerEdgeDragWidth, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) drawerEnableOpenDragGesture = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) endDrawerEnableOpenDragGesture = true, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId})

Creates a visual scaffold for Material Design widgets.

*Const*

### Some Properties Of Scaffold

* floatingActionButton: A floating action button is a circular icon button that hovers over content to promote a primary action in the application. Floating action buttons are most commonly used in the [Scaffold.floatingActionButton](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButton.html) field.
* floatingActionButton: FloatingActionButton(
* onPressed: () {
* Navigator.push(context,
* MaterialPageRoute(builder: (context) => const Screen2()));
* },
* tooltip: 'FloatingActionButton',
* shape: const RoundedRectangleBorder(), // set hình dạng
* child: const Icon(Icons.add),
* ),
* // set vị trí của FloatingActionButton
* floatingActionButtonLocation: FloatingActionButtonLocation.centerFloat,
* );
* bottomNavigationBar: một thanh điều hướng dưới cùng (bottom navigation bar) cho màn hình của bạn. Thanh điều hướng dưới cùng thường được sử dụng để chuyển giữa các phần của ứng dụng hoặc màn hình khác nhau.

class Home2 extends StatefulWidget {

  const Home2({super.key});

  @override

  State<Home2> createState() => \_Home2State();

}

class \_Home2State extends State<Home2> {

  int \_selectedIndex = 0; // set index default khi load layout

  static const TextStyle optionStyle =

      TextStyle(fontSize: 30, fontWeight: FontWeight.bold);

  // List các option trên BottomNavigationBar

  static const List<Widget> \_widgetOptions = <Widget>[

    Text(

      'Index 0: Home',

      style: optionStyle,

    ),

    Text(

      'Index 1: Business',

      style: optionStyle,

    ),

    Text(

      'Index 2: School',

      style: optionStyle,

    ),

  ];

  void \_onItemTapped(int index) {

    setState(() {

      \_selectedIndex = index;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('BottomNavigationBar Sample 2'),

      ),

      body: Center(

// show object trong list \_widgetOptions ở trên. Khi onTap:

        // \_onItemTapped sẽ auto reload widget theo \_selectedIndex

        child: \_widgetOptions.elementAt(\_selectedIndex),

      ),

      bottomNavigationBar: BottomNavigationBar(

        items: const <BottomNavigationBarItem>[

          BottomNavigationBarItem(

              icon: Icon(Icons.home),

              label: 'Home',

              backgroundColor: Colors.red),

          BottomNavigationBarItem(

              icon: Icon(Icons.business),

              label: 'Business',

              backgroundColor: Colors.green),

          BottomNavigationBarItem(

              icon: Icon(Icons.school),

              label: 'School',

              backgroundColor: Colors.blue),

        ],

        currentIndex: \_selectedIndex, // set default selected index

        selectedItemColor: Colors.amber[800], // set color for selected item

        onTap:

            \_onItemTapped, // call function \_onItemTapped update \_selectedIndex

      ),

    );

  }

}

* Thuộc tính type: BottomNavigationBarType.shifting của BottomNavigationBar.

Thuộc tính type of BottomNavigationBarType có 2 value is fixed or shifting. Default is fixed (các items luôn fix tại đáy màn hình), if items > 3 auto switch to shifting - ko show label các items ko chọn, chỉ show lớn item đang focus. Nếu muốn dùng shifting thì phải set backgroundColor cho 1 or mỗi item

Nếu ko muốn BottomNavigationBarType auto switch thì set type: BottomNavigationBarType thủ công

(type: BottomNavigationBarType.shifting or fixed).

Nếu muốn show label của all items trong chế độ shifting thì set

showUnselectedLabels: true,

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('BottomNavigationBar Sample 2A'),

      ),

      body: Center(

        child: \_widgetOptions.elementAt(\_selectedIndex),

      ),

      bottomNavigationBar: BottomNavigationBar(

        /// set type of BottomNavigationBarType - fixed or shifting. Default is fixed, if item > 3 auto switch to shifting. Nếu muốn dùng shifting thì phải set backgroundColor cho 1 or mỗi item

        /// Nếu ko muốn BottomNavigationBarType auto switch thì set type: BottomNavigationBarType thủ công

        type: BottomNavigationBarType.shifting,

        showUnselectedLabels:

            true, // luôn show label của item ko được chọn trong chế độ shifting

        items: const <BottomNavigationBarItem>[

          BottomNavigationBarItem(

            icon: Icon(Icons.home),

            label: 'Home',

            backgroundColor: Colors.red,

          ),

          BottomNavigationBarItem(

            icon: Icon(Icons.business),

            label: 'Business',

            backgroundColor: Colors.green,

          ),

          BottomNavigationBarItem(

            icon: Icon(Icons.school),

            label: 'School',

            backgroundColor: Colors.purple,

          ),

          BottomNavigationBarItem(

            icon: Icon(Icons.settings),

            label: 'Settings',

            backgroundColor: Colors.pink,

          ),

        ],

        currentIndex: \_selectedIndex,

        selectedItemColor: Colors.amber[800],

        onTap: \_onItemTapped,

      ),

    );

  }

}

bottomNavigationBar: BottomNavigationBar - show page theo index - ko cần dùng Navigator

// Home3 - bottomNavigationBar: BottomNavigationBar - show page theo index - ko cần dùng Navigator

class Home3 extends StatefulWidget {

  const Home3({super.key});

  @override

  \_Home3State createState() => \_Home3State();

}

class \_Home3State extends State<Home3> {

  int \_selectedIndex = 0;

  static const List<Widget> \_pages = <Widget>[

    FirstPage(),

    SecondPage(),

    ThirdPage(),

  ];

  void \_onItemTapped(int index) {

    setState(() {

      \_selectedIndex = index;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('BottomNavigationBar Demo'),

      ),

      body: \_pages[

          \_selectedIndex], // show page theo index - ko cần dùng Navigator

      bottomNavigationBar: BottomNavigationBar(

        items: const <BottomNavigationBarItem>[

          // list các item trên BottomNavigationBar

          BottomNavigationBarItem(

            icon: Icon(Icons.home),

            label: 'Home',

          ),

          BottomNavigationBarItem(

            icon: Icon(Icons.business),

            label: 'Business',

          ),

          BottomNavigationBarItem(

            icon: Icon(Icons.school),

            label: 'School',

          ),

        ],

        currentIndex: \_selectedIndex, // set default selected index

        selectedItemColor: Colors.blue,

        onTap: \_onItemTapped,

      ),

    );

  }

}

class FirstPage extends StatelessWidget {

  const FirstPage({super.key});

  @override

  Widget build(BuildContext context) {

    return const Center(

      child: Text('Home Page'),

    );

  }

}

class SecondPage extends StatelessWidget {

  const SecondPage({super.key});

  @override

  Widget build(BuildContext context) {

    return const Center(

      child: Text('Business Page'),

    );

  }

}

class ThirdPage extends StatelessWidget {

  const ThirdPage({super.key});

  @override

  Widget build(BuildContext context) {

    return const Center(

      child: Text('School Page'),

    );

  }

}

bottomNavigationBar: NavigationBar

Home4 - bottomNavigationBar: NavigationBar

class Home4 extends StatefulWidget {

  const Home4({super.key});

  @override

  State<Home4> createState() => \_Home4State();

}

class \_Home4State extends State<Home4> {

  int currentPageIndex = 0;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      bottomNavigationBar: NavigationBar(

        onDestinationSelected: (int index) {

          // call function onDestinationSelected update currentPageIndex, render lại widget

          setState(() {

            currentPageIndex = index;

          });

        },

        indicatorColor: const Color.fromARGB(255, 249, 207, 153),

        selectedIndex: currentPageIndex, // set default selected index

        destinations: const <Widget>[

          // list các item trên NavigationBar

          NavigationDestination(

            selectedIcon: Icon(Icons.home),

            icon: Icon(Icons.home\_outlined),

            label: 'Home',

          ),

          NavigationDestination(

            icon: Icon(Icons.business),

            label: 'Business',

          ),

          NavigationDestination(

            selectedIcon: Icon(Icons.school),

            icon: Icon(Icons.school\_outlined),

            label: 'School',

          ),

        ],

      ),

      body: <Widget>[

        // body của Scaffold là list các page được định nghĩa bằng các Container

        Container(

          color: Colors.red,

          alignment: Alignment.center,

          child: const Text('Page 1'),

        ),

        Container(

          color: Colors.green,

          alignment: Alignment.center,

          child: const Text('Page 2'),

        ),

        Container(

          color: Colors.blue,

          alignment: Alignment.center,

          child: const Text('Page 3'),

        ),

      ][currentPageIndex],

    );

  }

}

* Thuộc tính persistentFooterButtons - list button fixed ở đáy màn hình

// Home5 - persistentFooterButtons

List<String> titles = <String>[

  'Cloud',

  'Beach',

  'Sunny',

];

class Home5 extends StatelessWidget {

  const Home5({super.key});

  @override

  Widget build(BuildContext context) {

    final ColorScheme colorScheme = Theme.of(context).colorScheme;

    final Color oddItemColor = colorScheme.primary.withOpacity(0.05);

    final Color evenItemColor = colorScheme.primary.withOpacity(0.15);

    const int tabsCount = 3;

    return DefaultTabController(

      // DefaultTabController - để tạo TabBar

      initialIndex: 0, // initialIndex - để chọn tab mặc định

      length: tabsCount, // length - số lượng tab

      child: Scaffold(

        persistentFooterButtons: [

          // persistentFooterButtons - tạo list các button fixed ở cuối màn hình

          ElevatedButton(

            onPressed: () {},

            child: const Text('Button 1'),

          ),

          ElevatedButton(

            onPressed: () {},

            child: const Text('Button 2'),

          ),

        ],

        persistentFooterAlignment: AlignmentDirectional

            .center, // set vị trí của persistentFooterButtons

        appBar: AppBar(

          title: const Text('AppBar Sample'),

          // This check specifies which nested Scrollable's scroll notification

          // should be listened to.

          //

          // When `ThemeData.useMaterial3` is true and scroll view has

          // scrolled underneath the app bar, this updates the app bar

          // background color and elevation.

          //

          // This sets `notification.depth == 1` to listen to the scroll

          // notification from the nested `ListView.builder`.

          notificationPredicate: (ScrollNotification notification) {

            return notification.depth == 1;

          },

          // The elevation value of the app bar when scroll view has

          // scrolled underneath the app bar.

          scrolledUnderElevation:

              10.0, // scrolledUnderElevation - độ nâng bóng của appbar khi scroll

          shadowColor: Theme.of(context).shadowColor,

          bottom: TabBar(

            // bottom của AppBar - tạo TabBar dưới AppBar

            tabs: <Widget>[

              Tab(

                icon: const Icon(Icons.cloud\_outlined),

                text: titles[0],

              ),

              Tab(

                icon: const Icon(Icons.beach\_access\_sharp),

                text: titles[1],

              ),

              Tab(

                icon: const Icon(Icons.brightness\_5\_sharp),

                text: titles[2],

              ),

            ],

          ),

        ),

        body: TabBarView(

          // body của Scaffold - tạo TabBarView cho TabBar

          children: <Widget>[

            ListView.builder(

              itemCount: 25,

              itemBuilder: (BuildContext context, int index) {

                return ListTile(

                  tileColor: index.isOdd ? oddItemColor : evenItemColor,

                  title: Text('${titles[0]} $index'),

                );

              },

            ),

            ListView.builder(

              itemCount: 25,

              itemBuilder: (BuildContext context, int index) {

                return ListTile(

                  tileColor: index.isOdd ? oddItemColor : evenItemColor,

                  title: Text('${titles[1]} $index'),

                );

              },

            ),

            ListView.builder(

              itemCount: 25,

              itemBuilder: (BuildContext context, int index) {

                return ListTile(

                  tileColor: index.isOdd ? oddItemColor : evenItemColor,

                  title: Text('${titles[2]} $index'),

                );

              },

            ),

          ],

        ),

      ),

    );

  }

}

* Thuộc tính bottomSheet - bottomSheet - cố định ở cuối màn hình

A persistent bottom sheet shows information that supplements the primary content of the app. A persistent bottom sheet remains visible even when the user interacts with other parts of the app.

// Home5A - bottomSheet - xem comment bên dưới

class Home5A extends StatelessWidget {

  const Home5A({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Home5A - bottomSheet'),

      ),

      body: const Center(

        child: Text('Your Screen Content'),

      ),

      bottomSheet: Container(

        // bottomSheet - cố định ở cuối màn hình

        padding: const EdgeInsets.all(16.0),

        color: Colors.blue,

        child: Row(

          mainAxisAlignment: MainAxisAlignment.spaceBetween,

          children: [

            const Text('This is a bottom sheet',

                style: TextStyle(color: Colors.white)),

            ElevatedButton(

              onPressed: () {

                // Xử lý khi nút trên bottom sheet được nhấn

              },

              child:

                  const Text('Action', style: TextStyle(color: Colors.white)),

            ),

          ],

        ),

      ),

    );

  }

}

- Hàm showModalBottomSheet - hiển thị bottom sheet, click ra ngoài bottom sheet để đóng

// Home6 - showModalBottomSheet -  - hiển thị bottom sheet, click ra ngoài bottom sheet để đóng bottom sheet

class Home6 extends StatelessWidget {

  const Home6({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Home6 - showModalBottomSheet'),

      ),

      body: const Center(

        child: Text('Your Screen Content'),

      ),

      floatingActionButton: FloatingActionButton(

        onPressed: () {

          // Hiển thị BottomSheet khi nhấn vào FloatingActionButton

          showModalBottomSheet(

            // showModalBottomSheet - hiển thị bottom sheet, click ra ngoài bottom sheet để đóng bottom sheet

            context: context,

            builder: (BuildContext context) {

              return const Padding(

                padding: EdgeInsets.all(16.0),

                child: Text(

                    'This is a BottomSheet - call function showModalBottomSheet'),

              );

            },

          );

        },

        child: const Icon(Icons.add),

      ),

    );

  }

}

- Hàm showBottomSheet - show bottom sheet trên 1 page mới, ko đóng được bottom sheet ở trang này

Trườn hợp này khi run sẽ xuất hiện lỗi FlutterError (No Scaffold widget found.

Home6A widgets require a Scaffold widget ancestor. Để xly this error phải đặt Scaffold ngoài widget này (widget cha của this widget)

// Home6A - showBottomSheet -  show bottom sheet trên 1 page mới, ko đóng được bottom sheet ở trang này

class Home6A extends StatelessWidget {

  const Home6A({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Bottom Sheet Demo'),

      ),

      body: Center(

        child: ElevatedButton(

          onPressed: () {

            // Khi nút được nhấn, hiển thị Bottom Sheet

            showBottomSheet(

              // show bottom sheet trên 1 page mới, ko đóng được bottom sheet ở trang này

              // lỗi FlutterError (No Scaffold widget found.

// Home6A widgets require a Scaffold widget ancestor. Để xly this error phải đặt Scaffold ngoài widget này (widget cha của this widget)

              context: context,

              builder: (context) {

                return Container(

                  height: 200,

                  color: Colors.blue,

                  child: const Center(

                    child: Text(

                      'This is a Bottom Sheet!',

                      style: TextStyle(color: Colors.white, fontSize: 20),

                    ),

                  ),

                );

              },

            );

          },

          child: const Text('Show Bottom Sheet'),

        ),

      ),

    );

  }

}

### Một số example bottomNavigationBar: NavigationDrawer

//----------------------------------------------------------------

// NavigationDrawer

// On top of Drawers, Navigation drawers offer a persistent and convenient way to switch between primary destinations in an app.

class NavigationDrawer01 extends StatefulWidget {

  const NavigationDrawer01({super.key});

  @override

  State<NavigationDrawer01> createState() => \_NavigationDrawer01State();

}

class \_NavigationDrawer01State extends State<NavigationDrawer01> {

  final PageController \_pageController = PageController();

  int \_currentIndex = 0;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text(

            'Bottom Navigation Drawer Demo - switch to corresponding page'),

      ),

      body: PageView(

        controller: \_pageController,

        onPageChanged: (index) {

          setState(() {

            \_currentIndex = index;

          });

        },

        children: const <Widget>[

          // children classes is defined in bottom

          HomePageContent(),

          SettingsPageContent(),

          AboutPageContent(),

        ],

      ),

      bottomNavigationBar: NavigationDrawer(

        // class NavigationDrawer is defined in bottom

        currentIndex: \_currentIndex,

        onTap: (index) {

          \_pageController.animateToPage(

            index,

            duration: const Duration(milliseconds: 300),

            curve: Curves.easeInOut,

          );

        },

      ),

    );

  }

}

class NavigationDrawer extends StatelessWidget {

  final int currentIndex;

  final Function(int) onTap;

  const NavigationDrawer(

      {super.key, required this.currentIndex, required this.onTap});

  @override

  Widget build(BuildContext context) {

    return BottomNavigationBar(

      currentIndex: currentIndex,

      onTap: onTap,

      items: const [

        BottomNavigationBarItem(

          icon: Icon(Icons.home),

          label: 'Home',

        ),

        BottomNavigationBarItem(

          icon: Icon(Icons.settings),

          label: 'Settings',

        ),

        BottomNavigationBarItem(

          icon: Icon(Icons.info),

          label: 'About',

        ),

      ],

    );

  }

}

class HomePageContent extends StatelessWidget {

  const HomePageContent({super.key});

  @override

  Widget build(BuildContext context) {

    return const Center(

      child: Text('Home Page Content'),

    );

  }

}

class SettingsPageContent extends StatelessWidget {

  const SettingsPageContent({super.key});

  @override

  Widget build(BuildContext context) {

    return const Center(

      child: Text('Settings Page Content'),

    );

  }

}

class AboutPageContent extends StatelessWidget {

  const AboutPageContent({super.key});

  @override

  Widget build(BuildContext context) {

    return const Center(

      child: Text('About Page Content'),

    );

  }

}

//

// Update contains in 1 page

class NavigationDrawer02 extends StatelessWidget {

  final GlobalKey<\_HomePageContentState> \_homePageKey = GlobalKey();

  NavigationDrawer02({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Bottom Navigation Drawer Demo - update only 1 page'),

      ),

      body: HomePageContentA(

          key: \_homePageKey), // class HomePageContentA is defined in bottom

      bottomNavigationBar: NavigationDrawerA(

          homePageKey:

              \_homePageKey), // class NavigationDrawerA is defined in bottom

    );

  }

}

class NavigationDrawerA extends StatelessWidget {

  final GlobalKey<\_HomePageContentState> homePageKey;

  const NavigationDrawerA({super.key, required this.homePageKey});

  @override

  Widget build(BuildContext context) {

    return BottomNavigationBar(

      onTap: (index) {

        // You can perform actions based on the tapped item index if needed

        // For example, you can update a state variable in the HomePageContent

        // and trigger a rebuild.

        homePageKey.currentState?.updateContent(index);

      },

      items: const [

        BottomNavigationBarItem(

          icon: Icon(Icons.home),

          label: 'Home',

        ),

        BottomNavigationBarItem(

          icon: Icon(Icons.settings),

          label: 'Settings',

        ),

        BottomNavigationBarItem(

          icon: Icon(Icons.info),

          label: 'About',

        ),

      ],

    );

  }

}

class HomePageContentA extends StatefulWidget {

  const HomePageContentA({Key? key}) : super(key: key);

  @override

  State<HomePageContentA> createState() => \_HomePageContentState();

}

class \_HomePageContentState extends State<HomePageContentA> {

  String content = 'Home Page Content';

  void updateContent(int index) {

    setState(() {

      // Update the content based on the tapped item index

      if (index == 1) {

        content = 'Settings Page Content';

      } else if (index == 2) {

        content = 'About Page Content';

      } else {

        content = 'Home Page Content';

      }

    });

  }

  @override

  Widget build(BuildContext context) {

    return Center(

      child: Text(content),

    );

  }

}

### Một số example về properties drawer và endDrawer

//---------------------------

// // endDrawer - dựng list items on right side; drawer - dựng list items on left side

class Drawer01 extends StatelessWidget {

  const Drawer01({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Scaffold with endDrawer'),

      ),

      body: const Center(

        child: Text('Main Content'),

      ),

      // endDrawer - dựng list items on right side; drawer - dựng list items on left side

      endDrawer: Drawer(

        // The content of the endDrawer

        child: ListView(

          padding: EdgeInsets.zero,

          children: <Widget>[

            const DrawerHeader(

              decoration: BoxDecoration(

                color: Colors.blue,

              ),

              child: Text(

                'End Drawer Header',

                style: TextStyle(

                  color: Colors.white,

                  fontSize: 24,

                ),

              ),

            ),

            ListTile(

              title: const Text('Item 1'),

              onTap: () {

                // Handle item tap

                Navigator.pop(context); // Close the drawer

              },

            ),

            ListTile(

              title: const Text('Item 2'),

              onTap: () {

                // Handle item tap

                Navigator.pop(context); // Close the drawer

              },

            ),

          ],

        ),

      ),

      drawer: Drawer(

        // The content of the endDrawer

        child: ListView(

          padding: EdgeInsets.zero,

          children: <Widget>[

            const DrawerHeader(

              decoration: BoxDecoration(

                color: Colors.blue,

              ),

              child: Text(

                'Drawer Header',

                style: TextStyle(

                  color: Colors.white,

                  fontSize: 24,

                ),

              ),

            ),

            ListTile(

              title: const Text('Item 1'),

              onTap: () {

                // Handle item tap

                Navigator.pop(context); // Close the drawer

              },

            ),

            ListTile(

              title: const Text('Item 2'),

              onTap: () {

                // Handle item tap

                Navigator.pop(context); // Close the drawer

              },

            ),

          ],

        ),

      ),

    );

  }

}

//----------------------------------------------------------------

// Using Scaffold.of(context).openEndDrawer/openDrawer();

class Drawer02 extends StatelessWidget {

  const Drawer02({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Scaffold with openEndDrawer'),

      ),

      body: Center(

        /// Scaffold.of(context) requires a BuildContext from the subtree of the widget tree that includes the Scaffold. If the ElevatedButton is not within the subtree of the Scaffold, it won't be able to find the scaffold using Scaffold.of(context).

        child: Builder(builder: (context) {

          return Row(

            children: [

              ElevatedButton(

                onPressed: () {

                  // Open the end drawer programmatically

                  Scaffold.of(context).openEndDrawer();

                },

                child: const Text('Using openEndDrawer'),

              ),

              ElevatedButton(

                onPressed: () {

                  // Open the end drawer programmatically

                  Scaffold.of(context).openDrawer();

                },

                child: const Text('Using openDrawer'),

              ),

            ],

          );

        }),

      ),

      endDrawer: Drawer(

        // The content of the endDrawer

        child: ListView(

          padding: EdgeInsets.zero,

          children: <Widget>[

            const DrawerHeader(

              decoration: BoxDecoration(

                color: Colors.blue,

              ),

              child: Text(

                'End Drawer Header',

                style: TextStyle(

                  color: Colors.white,

                  fontSize: 24,

                ),

              ),

            ),

            ListTile(

              title: const Text('Item 1'),

              onTap: () {

                // Handle item tap

                Navigator.pop(context); // Close the drawer

              },

            ),

            ListTile(

              title: const Text('Item 2'),

              onTap: () {

                // Handle item tap

                Navigator.pop(context); // Close the drawer

              },

            ),

          ],

        ),

      ),

      drawer: Drawer(

        // The content of the endDrawer

        child: ListView(

          padding: EdgeInsets.zero,

          children: <Widget>[

            const DrawerHeader(

              decoration: BoxDecoration(

                color: Colors.blue,

              ),

              child: Text(

                'Drawer Header',

                style: TextStyle(

                  color: Colors.white,

                  fontSize: 24,

                ),

              ),

            ),

            ListTile(

              title: const Text('Item 1'),

              onTap: () {

                // Handle item tap

                Navigator.pop(context); // Close the drawer

              },

            ),

            ListTile(

              title: const Text('Item 2'),

              onTap: () {

                // Handle item tap

                Navigator.pop(context); // Close the drawer

              },

            ),

          ],

        ),

      ),

    );

  }

}

//----------------------------------------------------------------

// Change content page corresponding item menu

class Drawer03 extends StatefulWidget {

  const Drawer03({super.key});

  @override

  State<Drawer03> createState() => \_Drawer03State();

}

class \_Drawer03State extends State<Drawer03> {

  String selectedPage = '';

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Drawer Example'),

      ),

      drawer: Drawer(

        child: ListView(

          padding: EdgeInsets.zero,

          children: <Widget>[

            const DrawerHeader(

              decoration: BoxDecoration(

                color: Colors.blue,

              ),

              child: Text(

                'Drawer Header',

                style: TextStyle(

                  color: Colors.white,

                  fontSize: 24,

                ),

              ),

            ),

            ListTile(

              leading: const Icon(Icons.message),

              title: const Text('Messages'),

              onTap: () {

                setState(() {

                  selectedPage = 'Messages';

                });

              },

            ),

            ListTile(

              leading: const Icon(Icons.account\_circle),

              title: const Text('Profile'),

              onTap: () {

                setState(() {

                  selectedPage = 'Profile';

                });

              },

            ),

            ListTile(

              leading: const Icon(Icons.settings),

              title: const Text('Settings'),

              onTap: () {

                setState(() {

                  selectedPage = 'Settings';

                });

              },

            ),

          ],

        ),

      ),

      body: Center(

        child: Text('Page: $selectedPage'),

      ),

    );

  }

}

### Some Properties Of AppBar

// AppBarExample1 - action: IconButton

class AppBarExample1 extends StatelessWidget {

  const AppBarExample1({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        leading: const Icon(Icons

            .chair), // widget hiển thị bên trái title (có thể là icon, button,...)

        title: const Text('AppBar Demo'),

        actions: <Widget>[

          IconButton(

            icon: const Icon(Icons.add\_alert),

            tooltip: 'Show Snackbar',

            onPressed: () {

              ScaffoldMessenger.of(context).showSnackBar(

                  const SnackBar(content: Text('This is a snackbar')));

            },

          ),

          Padding(

            // Padding - để tạo khoảng cách lề phải cho IconButton

            padding: const EdgeInsets.only(right: 30),

            child: IconButton(

              icon: const Icon(Icons.navigate\_next),

              tooltip: 'Go to the next page',

              onPressed: () {

                Navigator.push(context, MaterialPageRoute<void>(

                  builder: (BuildContext context) {

                    return Scaffold(

                      appBar: AppBar(

                        // automaticallyImplyLeading:false, // ẩn nút back trên appbar

                        leading: const Icon(Icons.cabin),

                        title: const Text('Next page'),

                      ),

                      body: const Center(

                        child: Text(

                          'This is the next page',

                          style: TextStyle(fontSize: 24),

                        ),

                      ),

                    );

                  },

                ));

              },

            ),

          ),

        ],

      ),

      body: const Center(

        child: Text(

          'This is the home page',

          style: TextStyle(fontSize: 24),

        ),

      ),

    );

  }

}

// AppBarExample2 -  bottom: TabBar

List<String> titles = <String>[

  'Cloud',

  'Beach',

  'Sunny',

];

class AppBarExample2 extends StatelessWidget {

  const AppBarExample2({super.key});

  @override

  Widget build(BuildContext context) {

    final ColorScheme colorScheme = Theme.of(context).colorScheme;

    final Color oddItemColor = colorScheme.primary.withOpacity(0.05);

    final Color evenItemColor = colorScheme.primary.withOpacity(0.15);

    const int tabsCount = 3;

    return DefaultTabController(

      // DefaultTabController - để tạo TabBar

      initialIndex: 0, // initialIndex - để chọn tab mặc định

      length: tabsCount, // length - số lượng tab

      child: Scaffold(

        appBar: AppBar(

          title: const Text('AppBar Sample'),

          // This check specifies which nested Scrollable's scroll notification

          // should be listened to.

          //

          // When `ThemeData.useMaterial3` is true and scroll view has

          // scrolled underneath the app bar, this updates the app bar

          // background color and elevation.

          //

          // This sets `notification.depth == 1` to listen to the scroll

          // notification from the nested `ListView.builder`.

          notificationPredicate: (ScrollNotification notification) {

            return notification.depth == 1;

          },

          // The elevation value of the app bar when scroll view has

          // scrolled underneath the app bar.

          scrolledUnderElevation:

              10.0, // scrolledUnderElevation - độ nâng bóng của appbar khi scroll

          shadowColor: Theme.of(context).shadowColor,

          bottom: TabBar(

            // bottom của AppBar - tạo TabBar dưới AppBar

            tabs: <Widget>[

              Tab(

                icon: const Icon(Icons.cloud\_outlined),

                text: titles[0],

              ),

              Tab(

                icon: const Icon(Icons.beach\_access\_sharp),

                text: titles[1],

              ),

              Tab(

                icon: const Icon(Icons.brightness\_5\_sharp),

                text: titles[2],

              ),

            ],

          ),

        ),

        body: TabBarView(

          // body của Scaffold - tạo TabBarView cho TabBar

          children: <Widget>[

            ListView.builder(

              itemCount: 25,

              itemBuilder: (BuildContext context, int index) {

                return ListTile(

                  tileColor: index.isOdd ? oddItemColor : evenItemColor,

                  title: Text('${titles[0]} $index'),

                );

              },

            ),

            ListView.builder(

              itemCount: 25,

              itemBuilder: (BuildContext context, int index) {

                return ListTile(

                  tileColor: index.isOdd ? oddItemColor : evenItemColor,

                  title: Text('${titles[1]} $index'),

                );

              },

            ),

            ListView.builder(

              itemCount: 25,

              itemBuilder: (BuildContext context, int index) {

                return ListTile(

                  tileColor: index.isOdd ? oddItemColor : evenItemColor,

                  title: Text('${titles[2]} $index'),

                );

              },

            ),

          ],

        ),

      ),

    );

  }

}

## [Text](https://api.flutter.dev/flutter/widgets/Text-class.html) widget

[Text](https://api.flutter.dev/flutter/widgets/Text/Text.html)([String](https://api.flutter.dev/flutter/dart-core/String-class.html) data, {[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? style, [StrutStyle](https://api.flutter.dev/flutter/painting/StrutStyle-class.html)? strutStyle, [TextAlign](https://api.flutter.dev/flutter/dart-ui/TextAlign.html)? textAlign, [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection, [Locale](https://api.flutter.dev/flutter/dart-ui/Locale-class.html)? locale, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? softWrap, [TextOverflow](https://api.flutter.dev/flutter/painting/TextOverflow.html)? overflow, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? textScaleFactor, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? maxLines, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticsLabel, [TextWidthBasis](https://api.flutter.dev/flutter/painting/TextWidthBasis.html)? textWidthBasis, [TextHeightBehavior](https://api.flutter.dev/flutter/dart-ui/TextHeightBehavior-class.html)? textHeightBehavior, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectionColor})

A run of text with a single style.

The Text widget displays a string of text with single style. The string might break across multiple lines or might all be displayed on the same line depending on the layout constraints.

### Một số properties

Widget build(BuildContext context) {

    return const Text(

      'Tiện ích Văn bản hiển thị một chuỗi văn bản với một kiểu duy nhất. Chuỗi có thể ngắt trên nhiều dòng hoặc tất cả có thể được hiển thị trên cùng một dòng tùy thuộc vào các ràng buộc về bố cục.',

      textAlign: TextAlign.left,

      maxLines: 2, // show tối đa bao nhiêu dòng

      overflow: TextOverflow.ellipsis, // kiểu báo hiệu khi nội dung text vượt quá khung container

      textScaleFactor: 2.0, // thay đổi kích thước font theo tỉ lệ so với fontsize gốc

      style: TextStyle(fontWeight: FontWeight.bold), // thay đổi text style

    );

  }

Widget build(BuildContext context) {

// Using the TextSpan Text.rich constructor, the Text widget can display a paragraph with differently styled TextSpans. The sample that follows displays "Hello beautiful world" with different styles for each word.

// TextSpan widget có tính chất đệ quy và nó có thể bao gồm các TextSpan khác

    return const Text.rich(

      style: TextStyle(fontSize: 20 \* 1.5),

      TextSpan(

        text: 'Hello', // default text style

        children: <TextSpan>[ // mỗi TextSpan áp dụng style riêng

          TextSpan(

              text: ' beautiful ',

              style: TextStyle(fontStyle: FontStyle.italic)),

          TextSpan(

              text: 'world', style: TextStyle(fontWeight: FontWeight.bold)),

        ],

      ),

    );

  }

### Selection

Text is not selectable by default. To make a Text selectable, one can wrap a subtree with a SelectionArea widget. To exclude a part of a subtree under SelectionArea from selection, once can also wrap that part of the subtree with SelectionContainer.disabled.

 Widget build(BuildContext context) {

    return MaterialApp(

      home: Scaffold(

        appBar: AppBar(title: const Text('SelectionContainer.disabled Sample')),

        body: const Center(

          child: SelectionArea( // SelectionArea – cho phép selection text

            child: Column(

              mainAxisAlignment: MainAxisAlignment.center,

              children: <Widget>[

                Text('Selectable text'),

                SelectionContainer.disabled(child: Text('Non-selectable text')), // SelectionContainer.disabled – ko cho phép selection object này

                Text('Selectable text'),

              ],

            ),

          ),

        ),

      ),

    );

  }

### style: TextStyle

 style: TextStyle(

fontFamily: 'Roboto' // nếu mặc định Flutter ko hổ trợ thì phải setup font theo hướng dẫn bên dưới

fontSize: 50,

          fontWeight: FontWeight.bold, // từ w100(normal) đến w900(bold)

          wordSpacing: 0.5, // space giữa các word

          letterSpacing: 1, // space giữa các letter

          decoration: TextDecoration.lineThrough, // gạch ngang dòng text

          decorationColor: Colors.red, // màu lằn kẻ

          fontStyle: FontStyle.italic, // italic or normal

// một số cách set color

decorationColor: Colors.red, // Colors.ten\_color

          color: Colors.greenAccent, // Colors.ten\_color\_biến\_thể

          backgroundColor: Color.fromARGB(255, 243, 103, 145), // nhập mã ARGB

          color: Color(0xFF3366FF)), // nhập mã ARGB theo hexa (0xAARRGGBB)

// Set độ trong suốt text - Using the RichText constructor, the Text widget can display a paragraph with differently styled TextSpans.

  Widget build(BuildContext context) {

    return RichText(

      text: TextSpan(

        style: const TextStyle(fontSize: 20),

        children: <TextSpan>[

          TextSpan(

            text: "You don't have the votes.\n",

            style: TextStyle(color: Colors.black.withOpacity(0.6)),

          ),

          TextSpan(

            text: "You don't have the votes!\n",

            style: TextStyle(color: Colors.black.withOpacity(0.8)),

          ),

          TextSpan(

            text:

                "You're gonna need congressional approval and you don't have the votes!\n",

            style: TextStyle(color: Colors.black.withOpacity(1.0)),

          ),

        ],

      ),

    );

  }

### Định nghĩa fontSize trong MartialApp widget qua TextTheme

* Định nghĩa trong MartialApp qua TextTheme
* void main() {
* runApp(MaterialApp( // MaterialApp là widget build-in kế thừa các kiến trúc theo chuẩn design marterial
* theme: ThemeData(
* //fontFamily: 'Playfair Display',
* textTheme: const TextTheme(
* bodyLarge: TextStyle(fontSize: 18.0),
* bodyMedium: TextStyle(fontSize: 26.0),
* bodySmall: TextStyle(fontSize: 26.0),
* // ...  // and so on for every text style
* ),
* Áp dụng fontSize cho object
* TextSpan(
* text: "You don't have the votes!\n",
* style: TextStyle(
* color: Colors.black.withOpacity(0.8),
* fontSize: Theme.of(context).textTheme.bodyLarge?.fontSize), // gọi lại định nghĩa ở trên MaterialApp
* ),

### Dùng TextTheme thiết lập TextStyle cho cả ứng dụng

 Widget build(BuildContext context) {

    return MaterialApp(

      theme: ThemeData(

        textTheme: const TextTheme(

          bodyMedium: TextStyle( // văn bản thông thường dùn thuộc tính bodyMedium

              fontFamily: 'Roboto',

              fontSize: 30,

              color: Colors.green,

              fontWeight: FontWeight.bold), // Font chữ mặc định

        ),

      ),

      home: HomeScreen(),

    );

  }

### DefaultTextStyle

DefaultTextStyle trong Flutter được sử dụng để thiết lập một TextStyle mặc định cho một nhóm widget con trong cây giao diện. Khi bạn bọc một hoặc nhiều widget con bên trong DefaultTextStyle, các widget đó sẽ thừa hưởng các thuộc tính của TextStyle được thiết lập trong DefaultTextStyle, trừ khi bạn ghi đè thuộc tính trong widget con.

DefaultTextStyle có thể hữu ích khi bạn muốn áp dụng cùng một TextStyle cho nhiều widget con, mà không cần phải đặt lại TextStyle cho từng widget. Thay vì chỉnh sửa từng widget riêng lẻ, bạn chỉ cần thiết lập DefaultTextStyle bao ngoài chúng.

 body: const Center(

          child: DefaultTextStyle( // áp dụng TextStyle cho các object trong child: Column phía dưới

            style: TextStyle(

              fontSize: 30,

              fontWeight: FontWeight.w500,

              color: Color.fromARGB(255, 183, 12, 225),

            ),

            child: Column(

              mainAxisAlignment: MainAxisAlignment.center,

              children: [

                Text('Hello'),

                Text('World'),

                Text('Flutter'),

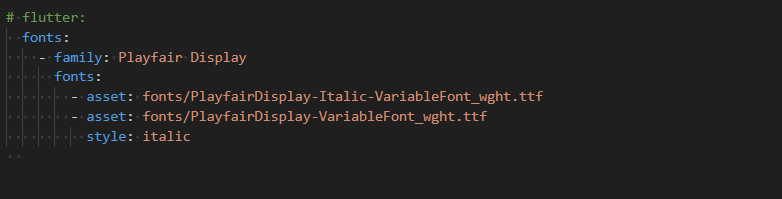
              ],

            ),

          ),

### Use a custom font

* Import the font files: tạo folder fonts trong project
* Declare the font in the pubspec.yaml – chú ý: khoảng cách thụt dòng phải chính xác



* Set a font as the default trong MaterialApp – family chính là fontFamily



* Use a font in a specific widget.

### Kiểm tra properties TextStyle với các method getParagraphStyle() và getTextStyle()

void printText\_Para\_Style({required BuildContext context}) {

  TextStyle textStyle = DefaultTextStyle.of(context).style;

  List<String> listA = textStyle.getParagraphStyle().toString().split(',');

  List<String> listB = textStyle.getTextStyle().toString().split(',');

  print('getParagraphStyle:\n');

  for (var style in listA) {

    // Kiểm tra các thuộc tính của style để xem nó có chứa từ 'unspecified' hay không

    if (!style.contains('unspecified')) {

      print(style.toString());

    }

  }

  print('----------------');

  print('getTextStyle:\n');

  for (var style in listB) {

    // Kiểm tra các thuộc tính của style để xem nó có chứa từ 'unspecified' hay không

    if (!style.contains('unspecified')) {

      print(style.toString());

    }

  }

}

### Padding & Margin

 Widget build(BuildContext context) {

    return Card(

      margin: EdgeInsets.all(20.0),

      margin: EdgeInsets.fromLTRB(50, 50, 30, 50),

      margin: EdgeInsets.symmetric(horizontal: 50),

      color: Colors.amber,

      child: Padding(

        padding: EdgeInsets.all(10.0), // set cùng value cho 4 cạnh

        padding: EdgeInsets.fromLTRB(10, 10, 20,20), // set value cho từng cạnh: Left - Top - Right - Bottom

        padding: EdgeInsets.only(left: 30), // set value cho các cạnh chỉ định

        padding: EdgeInsets.only(left: 30, bottom: 30),

        padding: EdgeInsets.symmetric(horizontal: pixel), // set value 2 cạnh đối xứng (symmetric)

        padding: EdgeInsets.symmetric(horizontal: 10, vertical: 40),

        // child: Text('Flutter self-taught'),

        child: Text('Flutter'),

      ),

    );

  }

}

Một số ngữ cảnh sử dụng padding

1. Trong widget cha để căn lề nội dung của các widget con:

Container(

            padding: EdgeInsets.all(16.0), // Khoảng cách xung quanh nội dung

            child: Text('Hello, world!'),

          )),

 ListView(

          padding: EdgeInsets.symmetric(vertical: 16.0), // Lề ở cả hai phía dọc

          children: [

            ListTile(title: Text('Item 1')),

            ListTile(title: Text('Item 2')),

            // ...

          ],

        )

1. Trong widget Card để tạo lề cho các chi tiết bên trong:

 Card(

          margin: EdgeInsets.all(16.0), // Khoảng cách xung quanh thẻ

          child: Column(

            children: [

              ListTile(title: Text('Title')),

              Padding(

                padding: EdgeInsets.all(16.0), // Lề cho nội dung bên trong

                child: Text('Content'),

              ),

            ],

          ),

        )

## Image

[Image](https://api.flutter.dev/flutter/widgets/Image/Image.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)> image, [ImageFrameBuilder](https://api.flutter.dev/flutter/widgets/ImageFrameBuilder.html)? frameBuilder, [ImageLoadingBuilder](https://api.flutter.dev/flutter/widgets/ImageLoadingBuilder.html)? loadingBuilder, [ImageErrorWidgetBuilder](https://api.flutter.dev/flutter/widgets/ImageErrorWidgetBuilder.html)? errorBuilder, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticLabel, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) excludeFromSemantics = false, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? height, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [Animation](https://api.flutter.dev/flutter/animation/Animation-class.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>? opacity, [BlendMode](https://api.flutter.dev/flutter/dart-ui/BlendMode.html)? colorBlendMode, [BoxFit](https://api.flutter.dev/flutter/painting/BoxFit.html)? fit, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html) alignment = Alignment.center, [ImageRepeat](https://api.flutter.dev/flutter/painting/ImageRepeat.html) repeat = ImageRepeat.noRepeat, [Rect](https://api.flutter.dev/flutter/dart-ui/Rect-class.html)? centerSlice, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) matchTextDirection = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) gaplessPlayback = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isAntiAlias = false, [FilterQuality](https://api.flutter.dev/flutter/dart-ui/FilterQuality.html) filterQuality = FilterQuality.low})

A widget that displays an image.

Example

 Widget build(BuildContext context) {

    return Column(

      children: [

        Image(

          image: FileImage(File('assets/Lion.jpg')),

          height: 200,

        ),

        const Image(

          image: AssetImage('assets/image\_a.webp'),

          height: 200,

        ),

        const Image(

          image: NetworkImage(

              'https://flutter.github.io/assets-for-api-docs/assets/widgets/owl.jpg'),

          height: 200,

        ),

      ],

    );

  }

Several constructors are provided for the various ways that an image can be specified:

Image.asset, for obtaining an image from an AssetBundle using a key.

* Tạo folder chứa image (example /assets tại root folder)
* Khai báo trong file pubspec.yaml
* assets:
* - assets/ # Khai báo path tổng quát cho all file
* - assets/image3.jpg # Khai báo riêng cho từng file
* - assets/image2.jpg
* - assets/image1.png
* Truy cập
* children: [
* Image.asset('assets/image1.png', width: 150, height: 150),
* Image.asset('assets/image2.jpg', width: 150, height: 150),
* Image.asset('assets/image3.jpg', width: 150, height: 150),
* Image.asset('assets/fish.png', width: 150, height: 150),
* ],

Image.network, for obtaining an image from a URL.

 Widget build(BuildContext context) {

    return Row(

      mainAxisAlignment: MainAxisAlignment.spaceEvenly,

      children: [

        const Image(

            width: 150,

            height: 150,

            image: NetworkImage(// Fetches the given URL from the network

                'https://flutter.github.io/assets-for-api-docs/assets/widgets/owl.jpg')),

        Image.network(

            // Image.network for a shorthand of an Image widget backed by NetworkImage.

            'https://flutter.github.io/assets-for-api-docs/assets/widgets/owl-2.jpg',

            width: 150,

            height: 150),

      ],

    );

  }

}

Image.file, for obtaining an image from a File.

import 'dart:io'; // tham chiếu cho object File

import 'package:flutter/material.dart';

class MyFileImageWidget extends StatelessWidget {

  final File imageFile; // phải import 'dart:io'

  MyFileImageWidget(this.imageFile);

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: Scaffold(

        appBar: AppBar(

          title: Text('File Image Example'),

        ),

        body: Center(

          child: Image.file(imageFile),

        ),

      ),

    );

  }

}

void main() {

  runApp(MyFileImageWidget(File('/path/to/your/image.jpg')));

}

Image.memory, for obtaining an image from a Uint8List.

The following image formats are supported: JPEG, PNG, GIF, Animated GIF, WebP, Animated WebP, BMP, and WBMP. Additional formats may be supported by the underlying platform. Flutter will attempt to call platform API to decode unrecognized formats, and if the platform API supports decoding the image Flutter will be able to render it.

To automatically perform pixel-density-aware asset resolution, specify the image using an AssetImage and make sure that a MaterialApp, WidgetsApp, or MediaQuery widget exists above the Image widget in the widget tree.

The image is painted using paintImage, which describes the meanings of the various fields on this class in more detail.

### Một số thuộc tính của widget Image

* Mức ưu tiên các properties set kích thước ảnh: có cảnh hưởng lẫn nhau nhưng ko được lớn hơn kích thước set với width/height
* Scale: ko ưu tiên bằng width/height. Thay đổi kích thước ảnh, default = 1, scale càng lớn ảnh càng nhỏ và ngược lại
* cacheWidth/cacheHeight: cung cấp kích thước ước tính của hình ảnh. Các giá trị này có thể giúp Flutter tối ưu hóa việc đọc và hiển thị hình ảnh từ cache, đặc biệt khi kích thước hiển thị không cần phải giống với kích thước gốc của hình ảnh.
* semanticLabel property: A Semantic description of the image. Used to provide a description of the image to TalkBack on Android, and VoiceOver on iOS.

TalkBack là một tính năng trợ năng tích hợp sẵn trên Android, được thiết kế để giúp đọc nội dung trên màn hình cho người dùng. Cụ thể, những mục, nội dung mà người dùng chạm vào sẽ được đọc thành tiếng khi tính năng này được bật.

TalkBack sẽ rất hữu ích với những người khiếm thị hoặc trong những tình huống mà người dùng không thể nhìn vào màn hình, nhưng vẫn muốn nắm bắt được nội dung đang hiển thị trên đó. Do vậy, nếu bạn hoàn toàn có thể nhìn và đọc nội dung trên màn hình thiết bị hoàn toàn bình thường, có thể tắt tính năng này đi chỉ với một vài thao tác đơn giản.

* opacity: độ trong suốt của ảnh – value = 1 is 100%, set < 1.

Trường hợp animation tĩnh - phải truyền bằng An animation that is always stopped at a given value như sau:

opacity: const AlwaysStoppedAnimation(0.7),

* color & colorBlendMode: trộn màu cho ảnh: color - màu muốn trộn với ảnh, colorBlendMode – kiểu trộn

Example

 child: Column(

            mainAxisAlignment: MainAxisAlignment.center,

            children: [

              Image.asset(

                'assets/bear.jpg',

                width: 200,

                height: 200,

                color: Colors.yellow[500], // set màu trộn

                colorBlendMode: BlendMode.color, // set kiểu trộn

              ),

              SizedBox(height: 10),

              Image.asset(

                'assets/bear.jpg',

                width: 200,

                height: 200,

                color: Colors.yellow[500],

                colorBlendMode: BlendMode.colorBurn,

              ),

              SizedBox(height: 10),

              Image.asset(

                'assets/bear.jpg',

                width: 200,

                height: 200,

                color: Colors.yellow[500],

                colorBlendMode: BlendMode.difference,

              ),

            ],

          ),

* fit: Thuộc tính fit của widget Image trong Flutter có nhiều giá trị khác nhau để điều chỉnh cách hình ảnh được hiển thị trong không gian hiển thị. Dưới đây là mô tả về tất cả các giá trị của thuộc tính fit

(Để test các fit value phải đặt image trong 1 widget cha như Container, set border trước)

1. **BoxFit.fill:** Hình ảnh sẽ được kéo dãn để che đậy toàn bộ không gian hiển thị mà có thể làm biến dạng hình ảnh nếu tỷ lệ khung hình không khớp.
2. **BoxFit.contain:** Hình ảnh sẽ được co lại hoặc kéo dãn sao cho nó phù hợp vừa vặn trong không gian hiển thị mà vẫn giữ nguyên tỷ lệ khung hình ban đầu.
3. **BoxFit.cover:** Hình ảnh sẽ được co lại hoặc kéo dãn sao cho nó che đậy toàn bộ không gian hiển thị mà vẫn giữ nguyên tỷ lệ khung hình ban đầu. Một phần của hình ảnh có thể bị cắt bỏ để vừa vặn.
4. **BoxFit.fitWidth:** Hình ảnh sẽ được kéo dãn chiều ngang để vừa vặn với chiều rộng của không gian hiển thị, giữ nguyên tỷ lệ khung hình.
5. **BoxFit.fitHeight:** Hình ảnh sẽ được kéo dãn chiều cao để vừa vặn với chiều cao của không gian hiển thị, giữ nguyên tỷ lệ khung hình.
6. **BoxFit.none:** Hình ảnh sẽ được hiển thị với kích thước thực tế của nó mà không điều chỉnh.
7. **BoxFit.scaleDown:** Hình ảnh sẽ được hiển thị như thường, nhưng nó sẽ co lại (nếu cần) để không vượt qua kích thước gốc của hình ảnh.

Mỗi giá trị của **fit** mang ý nghĩa và ứng dụng khác nhau tùy thuộc vào thiết kế giao diện cụ thể của bạn và cách bạn muốn hiển thị hình ảnh.

* - Top of Form
* alignment: căn lề cho image theo container (Để test các alignment value phải đặt image trong 1 widget cha như Container, set border trước)

Example

  children: [

              Container(

                width: 300,

                height: 200,

                decoration: BoxDecoration(

                  border: Border.all(color: Colors.blue, width: 2.0),

                ),

                child: Image.asset(

                  'assets/bear.jpg',

                  // width: 150,

                  // height: 150,

                  scale:

                      4.0, // size gốc ảnh > size container: set width/height thì ảnh tự bung ra full size container --> phải set qua scale

                  // fit: BoxFit.scaleDown,

                  alignment: Alignment.bottomCenter,

                ),

              ),

              SizedBox(height: 10),

              Container(

                width: 300,

                height: 200,

                decoration: BoxDecoration(

                  border: Border.all(color: Colors.blue, width: 2.0),

                ),

                child: Image.asset(

                  'assets/bear.jpg',

                  scale: 4,

                  alignment: Alignment.topRight,

                ),

              ),

              SizedBox(height: 10),

              Container(

                width: 300,

                height: 200,

                decoration: BoxDecoration(

                  border: Border.all(color: Colors.blue, width: 2.0),

                ),

                child: Image.asset(

                  'assets/bear.jpg',

                  scale: 4,

                  alignment: Alignment.centerRight,

                ),

              ),

* repeate: show lặp ảnh trong container

## Icon widget

[Icon](https://api.flutter.dev/flutter/widgets/Icon/Icon.html)([IconData](https://api.flutter.dev/flutter/widgets/IconData-class.html)? icon, {[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? size, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? fill, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? weight, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? grade, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? opticalSize, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Shadow](https://api.flutter.dev/flutter/dart-ui/Shadow-class.html)>? shadows, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticLabel, [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection})

Hiển thị hình ảnh các icon cơ bản trong *IconData* class.

Tra cứu thư viện icons tại: https://fonts.google.com/icons

Example

class MyIconWidget2 extends StatelessWidget {

  const MyIconWidget2({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: Scaffold(

        appBar: AppBar(

          title: const Text('Icon Widget Demo'),

        ),

        body: Column(

          children: [

            const Row(children: [

//Hiển thị Icon đơn giản

              SizedBox(

                height: 50,

                child: Text('Icon đơn giản'),

              ),

              Icon(

                Icons.star,

                color: Colors.yellow,

                size: 60.0,

              ),

            ]),

//Icon trong Button

            Row(children: [

              const SizedBox(

                height: 20,

                child: Text('Icon trong Button'),

              ),

              ElevatedButton.icon(

                onPressed: () {},

                icon: Icon(Icons.add),

                label: Text('Add'),

              ),

            ]),

// Icon cùng dòng với text

            const Row(children: [

              SizedBox(

                height: 20,

                child: Text(

                  'Icon cùng dòng với text',

                ),

              ),

              Row(

                children: [

                  Icon(Icons.email, color: Colors.blue),

                  SizedBox(width: 5),

                  Text('nmson2504@gmail.com'),

                ],

              ),

            ]),

            //

            /\*

            Đặt ListTile trực tiếp vào Row sẽ gây lỗi:

            Exception caught by scheduler library ═════════════════════════════════

Updated layout information required for RenderParagraph#b0f3f NEEDS-LAYOUT NEEDS-PAINT to calculate semantics.

'package:flutter/src/rendering/object.dart':

object.dart:1

Failed assertion: line 3457 pos 12: '!\_needsLayout'

           const Row(

              children: [

                Text('Icon trong ListTile'),

                ListTile(

                  leading: Icon(Icons.account\_circle),

                  title: Text('John Doe'),

                  subtitle: Text('Software Engineer'),

                  trailing: Icon(Icons.arrow\_forward),

                ),

              ],

            ), \*/

 // Icon trong ListTitle

            const Row(

              children: [

                Expanded(

                  child: Column(

                    children: [

                      Text('Icon trong ListTile'),

                      ListTile(

                        leading: Icon(Icons.account\_circle),

                        title: Text('John Doe'),

                        subtitle: Text('Software Engineer'),

                        trailing: Icon(Icons.arrow\_forward),

                      ),

                    ],

                  ),

                ),

              ],

            ),

// Icon trong nền 1 object

            Row(children: [

              const Text('Icon trong nền 1 object'),

              // Icon trong nền 1 object

              Container(

                width: 100,

                height: 100,

                decoration: const BoxDecoration(

                  color: Colors.blue,

                  shape: BoxShape.circle,

                ),

                child: const Icon(

                  Icons.ac\_unit,

                  color: Colors.white,

                  size: 20.0,

                ),

              ),

            ]),

// Icon cho 1 hành động

            Row(children: [

              const Text('Icon cho 1 hành động'),

              // Icon cho 1 hành động

              GestureDetector(

                onTap: () {

                  // Hành động khi người dùng chạm vào icon

                },

                child: const Icon(Icons.touch\_app),

              )

            ])

          ],

        ),

      ),

    );

  }

}

## Button

Trong Flutter, có một số loại button khác nhau mà bạn có thể sử dụng để tạo các tương tác người dùng khác nhau. Dưới đây là một số loại button phổ biến trong Flutter:

1. **ElevatedButton**: Đây là một loại button trung tính, có hiệu ứng độ nổi bật để làm nổi bật trước mặt. Nó thường được sử dụng cho các hành động quan trọng.
2. **TextButton**: Đây là một loại button văn bản đơn giản, thường được sử dụng cho các tương tác ít quan trọng hơn.
3. **OutlinedButton**: Loại button này có viền xung quanh và không có màu nền. Nó thường được sử dụng khi bạn muốn tạo ra các hành động như "Cancel" hoặc "Dismiss".
4. **IconButton**: Đây là một loại button có biểu tượng bên trong. Nó thường được sử dụng để thực hiện các hành động cụ thể.
5. **FloatingActionButton**: Đây là một button hình tròn nổi bật thường được đặt ở góc dưới bên phải màn hình. Nó thường được sử dụng cho các hành động chính trong ứng dụng.

[TextButton](https://api.flutter.dev/flutter/material/TextButton/TextButton.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onPressed, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onLongPress, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onHover, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle-class.html)? style, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none, [MaterialStatesController](https://api.flutter.dev/flutter/material/MaterialStatesController-class.html)? statesController, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? isSemanticButton = true, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child})

Create a TextButton.

[TextButton.icon](https://api.flutter.dev/flutter/material/TextButton/TextButton.icon.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onPressed, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onLongPress, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onHover, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle-class.html)? style, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? autofocus, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html)? clipBehavior, [MaterialStatesController](https://api.flutter.dev/flutter/material/MaterialStatesController-class.html)? statesController, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) icon, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) label})

Create a text button from a pair of widgets that serve as the button's icon and label.

Định dạng button

Constructor của [ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle-class.html)?

[ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle/ButtonStyle.html)({[MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)?>? textStyle, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? backgroundColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? foregroundColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? shadowColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? surfaceTintColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)?>? elevation, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)?>? padding, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Size](https://api.flutter.dev/flutter/dart-ui/Size-class.html)?>? minimumSize, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Size](https://api.flutter.dev/flutter/dart-ui/Size-class.html)?>? fixedSize, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Size](https://api.flutter.dev/flutter/dart-ui/Size-class.html)?>? maximumSize, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? iconColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)?>? iconSize, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[BorderSide](https://api.flutter.dev/flutter/painting/BorderSide-class.html)?>? side, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[OutlinedBorder](https://api.flutter.dev/flutter/painting/OutlinedBorder-class.html)?>? shape, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)?>? mouseCursor, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? tapTargetSize, [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html)? animationDuration, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enableFeedback, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html)? alignment, [InteractiveInkFeatureFactory](https://api.flutter.dev/flutter/material/InteractiveInkFeatureFactory-class.html)? splashFactory})

Create a [ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle-class.html) constant

Implement:

style: **const** ButtonStyle(

backgroundColor: MaterialStatePropertyAll<Color>(Colors.green),

),

Or static method

[styleFrom](https://api.flutter.dev/flutter/material/ElevatedButton/styleFrom.html)({[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? foregroundColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? disabledForegroundColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? disabledBackgroundColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? shadowColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? surfaceTintColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? elevation, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? textStyle, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [Size](https://api.flutter.dev/flutter/dart-ui/Size-class.html)? minimumSize, [Size](https://api.flutter.dev/flutter/dart-ui/Size-class.html)? fixedSize, [Size](https://api.flutter.dev/flutter/dart-ui/Size-class.html)? maximumSize, [BorderSide](https://api.flutter.dev/flutter/painting/BorderSide-class.html)? side, [OutlinedBorder](https://api.flutter.dev/flutter/painting/OutlinedBorder-class.html)? shape, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? enabledMouseCursor, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? disabledMouseCursor, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? tapTargetSize, [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html)? animationDuration, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enableFeedback, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html)? alignment, [InteractiveInkFeatureFactory](https://api.flutter.dev/flutter/material/InteractiveInkFeatureFactory-class.html)? splashFactory, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? primary, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? onPrimary, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? onSurface}) → [ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle-class.html)

Implement:

style: ElevatedButton.styleFrom(textStyle: const TextStyle(fontSize: 20));

Sử dụng style: TextButton.styleFrom có thể giúp bạn giữ mã ngắn gọn hơn bằng cách tạo ra các cấu hình tập trung. Trong khi đó, style: const ButtonStyle(...) tạo ra một đối tượng ButtonStyle riêng lẻ trong cùng một câu lệnh, có thể làm cho mã trông phân tán hơn.

Dưới đây là 1 số example code cùng với các properties của button

Example 1 - TextButton

//

Widget build(BuildContext context) {

    return Scaffold(body: Center(

      child: FloatingActionButton(onPressed: () {

        showDialog(

            context: context,

            builder: (ctxt) => new  AlertDialog(

                  title: Text("Text Dialog"),

                ));

      }),

    ));

  }

//

class MyAppImpl extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(title: const Text('Alert Dialog Example')),

        body: Center(

          child: TextButton(

            onPressed: () {

              showDialog(

                  context: context,

                  builder: (ctxt) => new AlertDialog(

                        title: Text("Text Dialog"),

                      ));

            },

            child: Text('Text Button'),

          ),

        ));

  }

}

Example 2 - ElevatedButton

class MyAppImpl3 extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(title: const Text('Test Properties Example 2')),

        body: Center(

          child: ElevatedButton.icon(

            style: ElevatedButton.styleFrom(

              backgroundColor: // màu nền button

                  Color.fromARGB(255, 250, 253, 66), // cũ là primary

              foregroundColor: // màu chữ button

                  Color.fromARGB(163, 14, 246, 2), // cũ là onPrimary

              side: const BorderSide( // viền button

                // viền button

                color: Colors.blue,

                width: 2.0,

              ),

              shape: const RoundedRectangleBorder( // tạo định dạng bo góc cho button

                borderRadius: BorderRadius.horizontal(

                  left: Radius.circular(20), // Bán kính cho góc trái

                  right: Radius.circular(30), // Bán kính cho góc phải),

                ),

              ),

            ),

            onHover: (value) { // rê chuột lên

              print('onHover: .......');

            },

            onLongPress: () { // click giữ button

              // ấn giữ button

              print('onLongPress:.........');

            },

            onPressed: () { // click

              print('onPressed - ElevatedButton');

              // Lấy chủ đề hiện tại

            },

// child: Text('Elevated Button'), // khi dùng button.icon thì thuộc tính child: sẽ bị thay thế bằng 2 thuộc tính icon: và label:

            label: const Text('Elevated Button Icon'),

            icon: const Icon(

              Icons.abc,

              size: 30,

              color: Colors.redAccent,

            ),

          ),

        ));

  }

}

Example 3 - ElevatedButton

class MyAppImpl1 extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(title: const Text('Test Properties Example')),

        body: Center(

          child: ElevatedButton(

            style: ElevatedButton.styleFrom(

              backgroundColor:

                  Color.fromARGB(255, 250, 253, 66), // cũ là primary

              foregroundColor: Color.fromARGB(163, 14, 246, 2),

              disabledBackgroundColor: Color.fromARGB(255, 223, 234, 238), // để show hiệu ứng này thì phải disable button bằng lệnh: onPressed: null,

              disabledForegroundColor: const Color.fromARGB(221, 42, 29, 29),

              //  minimumSize: Size(110, 20),

              // maximumSize: Size(160, 20)

              padding: EdgeInsets.all(30),

              elevation: 10, // độ nổi (elevation) 3D so với nền

              shadowColor:

                  Colors.blue, // phải set elevation > 0 mới có tác dụng

              side: BorderSide(

                // viền button

                color: Colors.blue,

                width: 2.0,

              ),

              shape: RoundedRectangleBorder(

                borderRadius: BorderRadius.circular(

                    20), // Đặt hình dáng thành hình tròn với bán kính 20

              ),

            ),

            // onPressed: null, // disable button

            child: Text('Elevated Button'),

          ),

        ));

  }

}

Xử lý lỗi "FlutterError (No MaterialLocalizations found…" click button (run app rồi vẫn chưa báo lỗi):

// Khi widget return ko có MaterialApp và void main() gọi …

class MyButtonAppError1 extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

        home: Scaffold(

            appBar: AppBar(title: const Text('Alert Dialog Example')),

            body: Center(

              child: FloatingActionButton(onPressed: () {

                // Hiển thị hộp thoại khi button được nhấn

                showDialog(

                    context: context,

                    builder: (ctxt) => new AlertDialog(

                          title: const Text("Text Dialog"),

                        ));

              }),

              // child: Text('Hiển thị Hộp thoại'),

            )));

  }

}

// file main.dart – void main() gọi:

void main() {

  runApp(MyButtonAppError1());

}

// Click button sẽ báo lỗi: "FlutterError (No MaterialLocalizations found…"

// Giải pháp: tách MartialApp ra nằm ở widget cha (widget gọi widget chứa các thành phần con của MaterialApp)

// return MaterialApp ở widget cha

class MyButtonApp1 extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(home: MyAppImpl());

  }

}

class MyAppImpl extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(title: const Text('Alert Dialog Example')),

        body: Center(

          child: TextButton(

            onPressed: () {

              showDialog(

                  context: context,

                  builder: (ctxt) => new AlertDialog(

                        title: Text("Text Dialog"),

                      ));

            },

            child: Text('Text Button'),

          ),

        ));

  }

}

// void main() gọi widget cha:  runApp(MyButtonApp1());

// Hoặc: gọi trực tiếp widget con như sau: runApp(MaterialApp(home: MyAppImpl()));

### BackButton class

[BackButton](https://api.flutter.dev/flutter/material/BackButton/BackButton.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle-class.html)? style, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onPressed})

Creates an [IconButton](https://api.flutter.dev/flutter/material/IconButton-class.html) with the appropriate "back" icon for the current target platform.

*const*

A Material Design back icon button.

A [BackButton](https://api.flutter.dev/flutter/material/BackButton-class.html) is an [IconButton](https://api.flutter.dev/flutter/material/IconButton-class.html) with a "back" icon appropriate for the current [TargetPlatform](https://api.flutter.dev/flutter/foundation/TargetPlatform.html). When pressed, the back button calls [Navigator.maybePop](https://api.flutter.dev/flutter/widgets/Navigator/maybePop.html) to return to the previous route unless a custom [onPressed](https://api.flutter.dev/flutter/material/BackButton/onPressed.html) callback is provided.

### CloseButton class

[CloseButton](https://api.flutter.dev/flutter/material/CloseButton/CloseButton.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onPressed, [ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle-class.html)? style})

Creates a Material Design close icon button.

*const*

A Material Design close icon button.

A CloseButton is an IconButton with a "close" icon. When pressed, the close button calls Navigator.maybePop to return to the previous route.

## PopupMenuButton

[PopupMenuButton](https://api.flutter.dev/flutter/material/PopupMenuButton/PopupMenuButton.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [PopupMenuItemBuilder](https://api.flutter.dev/flutter/material/PopupMenuItemBuilder.html)<T> itemBuilder, T? initialValue, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onOpened, [PopupMenuItemSelected](https://api.flutter.dev/flutter/material/PopupMenuItemSelected.html)<T>? onSelected, [PopupMenuCanceled](https://api.flutter.dev/flutter/material/PopupMenuCanceled.html)? onCanceled, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? tooltip, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? elevation, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? shadowColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? surfaceTintColor, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html) padding = const EdgeInsets.all(8.0), [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? splashRadius, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? icon, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? iconSize, [Offset](https://api.flutter.dev/flutter/dart-ui/Offset-class.html) offset = Offset.zero, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) enabled = true, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)? shape, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enableFeedback, [BoxConstraints](https://api.flutter.dev/flutter/rendering/BoxConstraints-class.html)? constraints, [PopupMenuPosition](https://api.flutter.dev/flutter/material/PopupMenuPosition.html)? position, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none})

Creates a button that shows a popup menu.

Displays a menu when pressed and calls onSelected when the menu is dismissed because an item was selected. The value passed to onSelected is the value of the selected menu item.

One of child or icon may be provided, but not both. If icon is provided, then PopupMenuButton behaves like an IconButton.

If both are null, then a standard overflow icon is created (depending on the platform).

Mặc định PopupMenuButton là dấu 3 chấm

Example 1

class PupupMenuButton1 extends StatelessWidget {

  const PupupMenuButton1({super.key});

  @override

  Widget build(BuildContext context) {

    return PopupMenuButton<String>(

      // required PopupMenuItemBuilder<T> itemBuilder

      itemBuilder: (BuildContext context) {

        return <String>['Option 1', 'Option 2', 'Option 3']

            .map((String choice) {

          return PopupMenuItem<String>(

            value:

                choice, // The value that will be returned by [showMenu] if this entry is selected.

            child: Text(choice),

          );

        }).toList();

      },

      initialValue: 'Option 2', // choice value mặc định

      tooltip: 'Click vào', // tooltip khi on hover icon popup menu

      // icon: const Icon(Icons.abc), // icon của popup menu, mặc định là dấu 3 chấm

      onSelected: (String choice) {

        debugPrint('You selected: $choice');

      },

      // child: tương đương thuộc tính icon: của PopupMenuButton (ko phải child của PopupMenuItem)

      // One of child or icon may be provided, but not both

      child: const Icon(Icons.access\_alarms),

    );

  }

}

Example 2

class \_PupupMenuButton3State extends State<PupupMenuButton3> {

List<IconData> listIcon = [Icons.alarm, Icons.mail, Icons.cabin];

  List<String> listItem = ['Option 1', 'Option 2', 'Option 3'];

  @override

  Widget build(BuildContext context) {

    return PopupMenuButton<String>(

      // required PopupMenuItemBuilder<T> itemBuilder

      itemBuilder:

          // Không cần tham số context

          (\_) {

        return listItem.map((String choice) {

          return PopupMenuItem<String>(

            value:

                choice, // The value that will be returned by [showMenu] if this entry is selected.

            child: Row(

              children: [

                Icon(listIcon[listItem.indexOf(choice)]),

                Text(choice),

              ],

            ),

          );

        }).toList();

      },

      initialValue: 'Option 2', // choice value mặc định

      tooltip: 'Click vào', // tooltip khi on hover icon popup menu

      // icon: const Icon(Icons.abc), // icon của popup menu, mặc định là dấu 3 chấm

      onSelected: (String choice) {

        debugPrint('You selected: $choice');

        // index =

        //     -1; // ko hiểu sao index load lần đầu xong giữ value cao nhất luôn - phải reset lại sau mõi lần click

      },

    );

  }

}

Example 3

// This is the type used by the popup menu below.

enum SampleItem { itemOne, itemTwo, itemThree }

class \_PopupMenuButton2State extends State<PopupMenuButton2> {

  SampleItem? selectedMenu;

  // phải convert enum -> list rồi mới .map được

  final List<SampleItem> listSampleItem = SampleItem.values.toList();

  @override

  Widget build(BuildContext context) {

    return PopupMenuButton<SampleItem>(

        initialValue: selectedMenu,

        // Callback that sets the selected popup menu item.

        onSelected: (SampleItem item) {

          setState(() {

            selectedMenu = item;

            debugPrint('$selectedMenu');

          });

        },

        itemBuilder: (\_) {

          return listSampleItem.map((SampleItem item) {

            return PopupMenuItem<SampleItem>(

              value: item,

              child: Text(item.name),

            );

          }).toList();

        });

  }

}

# Updating to MenuAnchor

There is a Material 3 component, MenuAnchor that is preferred for applications that are configured for Material 3 (see ThemeData.useMaterial3). The MenuAnchor widget's visuals are a little bit different, see the Material 3 spec at m3.material.io/components/menus/guidelines for more details.

## MenuAnchor

[MenuAnchor](https://api.flutter.dev/flutter/material/MenuAnchor/MenuAnchor.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [MenuController](https://api.flutter.dev/flutter/material/MenuController-class.html)? controller, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? childFocusNode, [MenuStyle](https://api.flutter.dev/flutter/material/MenuStyle-class.html)? style, [Offset](https://api.flutter.dev/flutter/dart-ui/Offset-class.html)? alignmentOffset = Offset.zero, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) anchorTapClosesMenu = false, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onOpen, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onClose, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) crossAxisUnconstrained = true, required [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> menuChildren, [MenuAnchorChildBuilder](https://api.flutter.dev/flutter/material/MenuAnchorChildBuilder.html)? builder, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a const [MenuAnchor](https://api.flutter.dev/flutter/material/MenuAnchor-class.html).

menuChildren: list các menu item, có thể là [MenuItemButton](https://api.flutter.dev/flutter/material/MenuItemButton-class.html) or/and SubmenuButton

child: label của MenuAnchor

builder: thuộc tính này là một hàm callback, để tạo một widget chứa nút hoặc phần giao diện người dùng mà khi được nhấn, nó sẽ kích hoạt hiển thị menu hoặc đóng menu.

Một widget được sử dụng để đánh dấu "điểm gốc" cho một tập hợp các submenu, vùng hình chữ nhật được sử dụng để định vị menu, điều này có thể thực hiện bằng cách sử dụng vị trí cụ thể hoặc bằng cách sử dụng alignment.

Khi tạo menu bằng MenuBar hoặc SubmenuButton, không cần sử dụng MenuAnchor, vì chúng cung cấp điều này bên trong.

MenuAnchor được thiết kế để là một giao diện có mức độ thấp hơn so với MenuBar, được sử dụng trong các tình huống mà MenuBar không thích hợp hoặc để xây dựng các widget hoặc vùng màn hình có submenu

### [MenuItemButton](https://api.flutter.dev/flutter/material/MenuItemButton/MenuItemButton.html)

[MenuItemButton](https://api.flutter.dev/flutter/material/MenuItemButton/MenuItemButton.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onPressed, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onHover, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) requestFocusOnHover = true, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [MenuSerializableShortcut](https://api.flutter.dev/flutter/widgets/MenuSerializableShortcut-mixin.html)? shortcut, [ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle-class.html)? style, [MaterialStatesController](https://api.flutter.dev/flutter/material/MaterialStatesController-class.html)? statesController, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? leadingIcon, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? trailingIcon, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) closeOnActivate = true, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a const [MenuItemButton](https://api.flutter.dev/flutter/material/MenuItemButton-class.html).

onPressed: hàm xử lý khi click vào item menu

shortcut: shortcut của item menu

child: label của item menu

 MenuAnchor(

          childFocusNode: \_buttonFocusNode,

          // required - menuChildren: (of MenuAnchor) A list of children containing the menu items

          menuChildren: <Widget>[

            MenuItemButton(

              child: Text(MenuEntry.about.label),

              onPressed: () => \_activate(MenuEntry.about),

            ),

            if (\_showingMessage)

              MenuItemButton(

                onPressed: () => \_activate(MenuEntry.hideMessage),

                shortcut: MenuEntry.hideMessage.shortcut,

                child: Text(MenuEntry.hideMessage.label),

              ),

            if (!\_showingMessage)

              MenuItemButton(

                //

                onPressed: () => \_activate(MenuEntry.showMessage),

                shortcut: MenuEntry.showMessage.shortcut,

                child: Text(MenuEntry.showMessage.label),

              ),

### SubmenuButton

[SubmenuButton](https://api.flutter.dev/flutter/material/SubmenuButton/SubmenuButton.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onHover, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onOpen, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onClose, [MenuController](https://api.flutter.dev/flutter/material/MenuController-class.html)? controller, [ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle-class.html)? style, [MenuStyle](https://api.flutter.dev/flutter/material/MenuStyle-class.html)? menuStyle, [Offset](https://api.flutter.dev/flutter/dart-ui/Offset-class.html)? alignmentOffset, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [MaterialStatesController](https://api.flutter.dev/flutter/material/MaterialStatesController-class.html)? statesController, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? leadingIcon, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? trailingIcon, required [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> menuChildren, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a const [SubmenuButton](https://api.flutter.dev/flutter/material/SubmenuButton-class.html).

A menu button that displays a cascading menu.

It can be used as part of a [MenuBar](https://api.flutter.dev/flutter/material/MenuBar-class.html), or as a standalone widget.

This widget represents a menu item that has a submenu. Like the leaf [MenuItemButton](https://api.flutter.dev/flutter/material/MenuItemButton-class.html), it shows a label with an optional leading or trailing icon, but additionally shows an arrow icon showing that it has a submenu.

menuChildren: list các menu item, có thể là [MenuItemButton](https://api.flutter.dev/flutter/material/MenuItemButton-class.html) or/and SubmenuButton

child: label của [SubmenuButton](https://api.flutter.dev/flutter/material/SubmenuButton-class.html)

    SubmenuButton(

              menuChildren: <Widget>[

                // required - menuChildren: (of SubmenuButton) A list of children containing the menu items

                MenuItemButton(

                  onPressed: () => \_activate(

                      MenuEntry.colorRed), // hàm xử lý khi click vào item menu

                  shortcut:

                      MenuEntry.colorRed.shortcut, // shortcut của item menu

                  child: Text(MenuEntry.colorRed.label), // label của item menu

                ),

                MenuItemButton(

                  onPressed: () => \_activate(MenuEntry.colorGreen),

                  shortcut: MenuEntry.colorGreen.shortcut,

                  child: Text(MenuEntry.colorGreen.label),

                ),

                MenuItemButton(

                  onPressed: () => \_activate(MenuEntry.colorBlue),

                  shortcut: MenuEntry.colorBlue.shortcut,

                  child: Text(MenuEntry.colorBlue.label),

                ),

              ],

              child: const Text(

                  'Background Color'), // required - child: The widget to display as the button. (child là con trục tiếp của SubmenuButton ở trên, ko nằm trong menuChildren)

            ),

### MenuAcceleratorLabel

A widget that draws the label text for a menu item (typically a [MenuItemButton](https://api.flutter.dev/flutter/material/MenuItemButton-class.html) or [SubmenuButton](https://api.flutter.dev/flutter/material/SubmenuButton-class.html)) and renders its child with information about the currently active keyboard accelerator.

(Nhập '&X' - Đánh dấu gạch chân ký tự shortcut cho menu item khi bấm phím Alt)

To use the accelerators, press the Alt key to see which letters are underlined in the menu bar, and then press the appropriate letter. Accelerators are not supported on macOS or iOS since those platforms don't support them natively, so this demo will only show a regular Material menu bar on those platforms.

   SubmenuButton(

                    menuChildren: <Widget>[

                      MenuItemButton(

                        onPressed: () {

                          ScaffoldMessenger.of(context).showSnackBar(

                            const SnackBar(

                              content: Text('Magnify!'),

                            ),

                          );

                        },

                        child: const MenuAcceleratorLabel('&Magnify'),

                      ),

                      MenuItemButton(

                        onPressed: () {

                          ScaffoldMessenger.of(context).showSnackBar(

                            const SnackBar(

                              content: Text('Minify!'),

                            ),

                          );

                        },

                        child: const MenuAcceleratorLabel(

                            'Mi&nify'), // &N - phím tắt: Alt + N

                      ),

                    ],

                    child: const MenuAcceleratorLabel('&View'),

                  ),

Example 1

enum SampleItem { itemOne, itemTwo, itemThree }

class \_MenuAnchor1State extends State<MenuAnchor1> {

  SampleItem? selectedMenu;

  @override

  Widget build(BuildContext context) {

    return Center(

      child: MenuAnchor(

        // buider: định nghĩa widget để kích hoạt(click) vào show menu ra

        builder:

            (BuildContext context, MenuController controller, Widget? child) {

          return IconButton(

            onPressed: () {

              if (controller.isOpen) {

                controller.close();

              } else {

                controller.open();

              }

            },

            icon: const Icon(Icons.more\_vert\_rounded),

            tooltip: 'Show menu',

          );

        },

        // menuChildren: A list of children containing the menu items

        menuChildren: List<MenuItemButton>.generate(

          3,

          (int index) => MenuItemButton(

            onPressed: () =>

                setState(() => selectedMenu = SampleItem.values[index]),

             child: Text(selectedMenu?.name ?? 'Item $index'), // selectedMenu?.name là null, thì Text sẽ hiển thị 'Item $index', ngược lại nó sẽ hiển thị giá trị của selectedMenu.name

          ),

        ),

      ),

    );

  }

Example 2

enum EnumItem { item1, item2, item3 }

class \_MenuAnchor0State extends State<MenuAnchor0> {

  EnumItem?

      selectedMenu; // phải đặt ngoài hàm build thì mới update value khi onPressed trên item

  @override

  Widget build(BuildContext context) {

    return Center(

      child: Row(

        mainAxisAlignment: MainAxisAlignment.center,

        children: <Widget>[

          MenuAnchor(

            // buider: định nghĩa widget để kích hoạt(click) vào show menu ra

            builder: (BuildContext context, MenuController controller,

                Widget? child) {

              return IconButton(

                onPressed: () {

                  if (controller.isOpen) {

                    controller.close();

                  } else {

                    controller.open();

                  }

                },

                icon: const Icon(Icons.more\_vert\_rounded),

                tooltip: 'Show menu',

              );

            },

            // menuChildren: A list of children containing the menu items

            menuChildren: [

              MenuItemButton(

                onPressed: () {

                  setState(() {

                    selectedMenu = EnumItem.item1;

                    debugPrint('$selectedMenu');

                  });

                },

                child: const Text('Item 1'),

              ),

              MenuItemButton(

                onPressed: () {

                  setState(() {

                    selectedMenu = EnumItem.item2;

                    debugPrint('$selectedMenu');

                  });

                },

                child: const Text('Item 2'),

              ),

              MenuItemButton(

                onPressed: () {

                  setState(() {

                    selectedMenu = EnumItem.item3;

                    debugPrint('$selectedMenu');

                  });

                },

                child: const Text('Item 3'),

              ),

            ],

          ),

          const SizedBox(height: 20),

          Text(

            'Mục đã chọn: ${selectedMenu?.name ?? "Chưa chọn"}', // nếu selectedMenu = null thì in "Chưa chọn"

            style: const TextStyle(fontSize: 18),

          ),

        ],

      ),

    );

  }

}

## MenuBar

[MenuBar](https://api.flutter.dev/flutter/material/MenuBar/MenuBar.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [MenuStyle](https://api.flutter.dev/flutter/material/MenuStyle-class.html)? style, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none, [MenuController](https://api.flutter.dev/flutter/material/MenuController-class.html)? controller, required [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children})

Creates a const [MenuBar](https://api.flutter.dev/flutter/material/MenuBar-class.html).

A menu bar that manages cascading child menus.

This is a Material Design menu bar that typically resides above the main body of an application (but can go anywhere) that defines a menu system for invoking callbacks in response to user selection of a menu item.

The items are identified with an enum value, and the shortcuts are registered globally with the ShortcutRegistry.

## TabBar

[TabBar](https://api.flutter.dev/flutter/material/TabBar/TabBar.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> tabs, [TabController](https://api.flutter.dev/flutter/material/TabController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isScrollable = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? indicatorColor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) automaticIndicatorColorAdjustment = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) indicatorWeight = 2.0, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html) indicatorPadding = EdgeInsets.zero, [Decoration](https://api.flutter.dev/flutter/painting/Decoration-class.html)? indicator, [TabBarIndicatorSize](https://api.flutter.dev/flutter/material/TabBarIndicatorSize.html)? indicatorSize, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? dividerColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? labelColor, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? labelStyle, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? labelPadding, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? unselectedLabelColor, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? unselectedLabelStyle, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enableFeedback, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[int](https://api.flutter.dev/flutter/dart-core/int-class.html)>? onTap, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [InteractiveInkFeatureFactory](https://api.flutter.dev/flutter/material/InteractiveInkFeatureFactory-class.html)? splashFactory, [BorderRadius](https://api.flutter.dev/flutter/painting/BorderRadius-class.html)? splashBorderRadius, [TabAlignment](https://api.flutter.dev/flutter/material/TabAlignment.html)? tabAlignment})

Creates a Material Design primary tab bar.

*const*

[TabBar.secondary](https://api.flutter.dev/flutter/material/TabBar/TabBar.secondary.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> tabs, [TabController](https://api.flutter.dev/flutter/material/TabController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isScrollable = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? indicatorColor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) automaticIndicatorColorAdjustment = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) indicatorWeight = 2.0, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html) indicatorPadding = EdgeInsets.zero, [Decoration](https://api.flutter.dev/flutter/painting/Decoration-class.html)? indicator, [TabBarIndicatorSize](https://api.flutter.dev/flutter/material/TabBarIndicatorSize.html)? indicatorSize, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? dividerColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? labelColor, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? labelStyle, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? labelPadding, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? unselectedLabelColor, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? unselectedLabelStyle, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enableFeedback, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[int](https://api.flutter.dev/flutter/dart-core/int-class.html)>? onTap, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [InteractiveInkFeatureFactory](https://api.flutter.dev/flutter/material/InteractiveInkFeatureFactory-class.html)? splashFactory, [BorderRadius](https://api.flutter.dev/flutter/painting/BorderRadius-class.html)? splashBorderRadius, [TabAlignment](https://api.flutter.dev/flutter/material/TabAlignment.html)? tabAlignment})

Creates a Material Design secondary tab bar.

*const*

Primary tabs are placed at the top of the content pane under a top app bar. They display the main content destinations.

Typically created as the [AppBar.bottom](https://api.flutter.dev/flutter/material/AppBar/bottom.html) part of an [AppBar](https://api.flutter.dev/flutter/material/AppBar-class.html) and in conjunction with a [TabBarView](https://api.flutter.dev/flutter/material/TabBarView-class.html)

If a [TabController](https://api.flutter.dev/flutter/material/TabController-class.html) is not provided, then a [DefaultTabController](https://api.flutter.dev/flutter/material/DefaultTabController-class.html) ancestor must be provided instead. The tab controller's [TabController.length](https://api.flutter.dev/flutter/material/TabController/length.html) must equal the length of the [tabs](https://api.flutter.dev/flutter/material/TabBar/tabs.html) list and the length of the [TabBarView.children](https://api.flutter.dev/flutter/material/TabBarView/children.html) list.

// Home1 - Use DefaultTabController

class Home1 extends StatelessWidget {

  const Home1({super.key});

  @override

  Widget build(BuildContext context) {

    return DefaultTabController(

      // DefaultTabController ancestor must be provided instead if TabController is not provided

      initialIndex: 0, // initial index of the tab

      length: 3, // length of tabs list

      child: Scaffold(

        appBar: AppBar(

          title: const Text('TabBar Sample'),

          // The tab controller's TabController.length must equal the length of the TabBar.tabs and the length of the TabBarView.children list.

          bottom: const TabBar(

            // Typically TabBar created as the AppBar.bottom

            tabs: <Widget>[

              Tab(

                icon: Icon(Icons.cloud\_outlined),

              ),

              Tab(

                icon: Icon(Icons.beach\_access\_sharp),

              ),

              Tab(

                icon: Icon(Icons.brightness\_5\_sharp),

              ),

            ],

          ),

        ),

        body: const TabBarView(

          children: <Widget>[

            Center(

              child: Text("It's cloudy here"),

            ),

            Center(

              child: Text("It's rainy here"),

            ),

            Center(

              child: Text("It's sunny here"),

            ),

          ],

        ),

      ),

    );

  }

}

// Home2 - User TabController

class Home2 extends StatefulWidget {

  const Home2({super.key});

  @override

  State<Home2> createState() => \_Home2State();

}

/// [AnimationController]s can be created with `vsync: this` because of [TickerProviderStateMixin].

class \_Home2State extends State<Home2> with TickerProviderStateMixin {

  late final TabController \_tabController;

  @override

  void initState() {

    super.initState();

    \_tabController =

        TabController(length: 3, vsync: this); // setup the tab controller

  }

  /// \_tabController.dispose();: The TabController is an object that manages the state of a TabBar and TabBarView. It should be disposed of when it's no longer needed to release any resources it may be holding.

  /// super.dispose();: This calls the dispose method of the superclass (State) to ensure that any additional cleanup required by the superclass is performed.

  @override

  void dispose() {

    \_tabController.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('TabBar Sample'),

        bottom: TabBar(

          controller:

              \_tabController, // set TabBar.controller and TabBarView.controller (below) when using TabController

          tabs: const <Widget>[

            Tab(

              icon: Icon(Icons.cloud\_outlined),

              text: 'data.cloud\_outlined',

            ),

            Tab(

              icon: Icon(Icons.beach\_access\_sharp),

              text: 'data.beach\_access\_sharp',

            ),

            Tab(

              icon: Icon(Icons.brightness\_5\_sharp),

              text: 'data.brightness\_5\_sharp',

            ),

          ],

        ),

      ),

      body: TabBarView(

        controller: \_tabController,

        children: const <Widget>[

          Center(

            child: Text("It's cloudy here"),

          ),

          Center(

            child: Text("It's rainy here"),

          ),

          Center(

            child: Text("It's sunny here"),

          ),

        ],

      ),

    );

  }

}

// Home3 -

// This sample showcases nested Material 3 TabBars. It consists of a primary TabBar with nested a secondary TabBar. The primary TabBar uses a DefaultTabController while the secondary TabBar uses a TabController.

class Home3 extends StatelessWidget {

  const Home3({super.key});

  @override

  Widget build(BuildContext context) {

    return DefaultTabController(

      initialIndex: 1,

      length: 3,

      child: Scaffold(

        appBar: AppBar(

          title: const Text('Primary and secondary TabBar'),

          bottom: const TabBar(

            dividerColor: Color.fromARGB(0, 57, 60, 238),

            tabs: <Widget>[

              Tab(

                text: 'Flights',

                icon: Icon(Icons.flight),

              ),

              Tab(

                text: 'Trips',

                icon: Icon(Icons.luggage),

              ),

              Tab(

                text: 'Explore',

                icon: Icon(Icons.explore),

              ),

            ],

          ),

        ),

        body: const TabBarView(

          children: <Widget>[

            NestedTabBar('Flights'), // NestedTabBar definition below

            NestedTabBar('Trips'),

            NestedTabBar('Explore'),

          ],

        ),

      ),

    );

  }

}

class NestedTabBar extends StatefulWidget {

  const NestedTabBar(this.outerTab, {super.key});

  final String outerTab;

  @override

  State<NestedTabBar> createState() => \_NestedTabBarState();

}

class \_NestedTabBarState extends State<NestedTabBar>

    with TickerProviderStateMixin {

  late final TabController \_tabController;

  @override

  void initState() {

    super.initState();

    \_tabController =

        TabController(length: 2, vsync: this); // setup the tab controller

  }

  /// \_tabController.dispose();: The TabController is an object that manages the state of a TabBar and TabBarView. It should be disposed of when it's no longer needed to release any resources it may be holding.

  /// super.dispose();: This calls the dispose method of the superclass (State) to ensure that any additional cleanup required by the superclass is performed.

  @override

  void dispose() {

    \_tabController.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return Column(

      children: <Widget>[

        TabBar.secondary(

          // first child is the TabBar

          // set TabBar.secondary

          controller:

              \_tabController, // set TabBar.controller and TabBarView.controller (below) when using TabController

          tabs: const <Widget>[

            Tab(text: 'Overview'),

            Tab(text: 'Specifications'),

          ],

        ),

        Expanded(

          // second child is the TabBarView

          child: TabBarView(

            controller: \_tabController,

            children: <Widget>[

              Card(

                margin: const EdgeInsets.all(16.0),

                ///Trong Flutter, thuộc tính widget được sử dụng để tham chiếu đến widget hiện tại trong lớp State của một widget StatefulWidget cụ thể. Nó tham chiếu đến instance của widget mà State đang quản lý. Điều này xảy ra vì lớp State và widget là một cặp - mỗi lần bạn tạo một widget StatefulWidget, một State object riêng biệt được tạo và gắn liền với widget đó.

                ///Trong trường hợp của mã Flutter bạn đã cung cấp, khi bạn tạo một NestedTabBar, widget đó được tạo và có một State object tương ứng được tạo. Do đó, trong phương thức build của NestedTabBar, bạn có thể sử dụng widget để tham chiếu đến widget hiện tại, và widget.outerTab để truy cập giá trị của thuộc tính outerTab đã được cung cấp qua hàm khởi tạo.

                /// Khi bạn tạo một NestedTabBar, bạn cung cấp một giá trị outerTab thông qua hàm khởi tạo, và giá trị này được lưu trong thuộc tính outerTab của widget. Sau đó, trong phần build, bạn sử dụng widget.outerTab để truy cập giá trị outerTab đó và hiển thị nó trong văn bản.

                child: Center(child: Text('${widget.outerTab}: Overview tab')),

              ),

              Card(

                margin: const EdgeInsets.all(16.0),

                child: Center(

                    child: Text('${widget.outerTab}: Specifications tab')),

              ),

            ],

          ),

        ),

      ],

    );

  }

}

## SnackBar

A lightweight message with an optional action which briefly displays at the bottom of the screen.

To display a snack bar, call ScaffoldMessenger.of(context).showSnackBar(), passing an instance of SnackBar that describes the message.

To control how long the SnackBar remains visible, specify a duration.

SnackBar sẽ không tự động biến mất khi TalkBack (Android) hoặc VoiceOver (iOS) được bật. Điều này được điều khiển bởi AccessibilityFeatures.accessibleNavigation.

 MenuItemButton(

                        onPressed: () {

                          ScaffoldMessenger.of(context).showSnackBar(

                            const SnackBar(

                                content: Text('Saved!'),

                                duration: Duration(

                                    seconds:

                                        1)), // show 1s (chỉ chấp nhận int). Default 4s

                          );

                        },

                        child: const MenuAcceleratorLabel(

                            '&Save'), // &S - phím tắt: Alt + S

                      ),

                      MenuItemButton(

                        onPressed: () {

                          ScaffoldMessenger.of(context).showSnackBar(

                            const SnackBar(

                              content: Text('Quit!'),

                              duration: Duration(

                                  milliseconds: 500), // 0.5s. Default 4s

                            ),

                          );

                        },

                        child: const MenuAcceleratorLabel('&Quit'),

                      ),

Example 2

  child: ElevatedButton(

        child: const Text('Show Snackbar'),

        onPressed: () {

          ScaffoldMessenger.of(context).showSnackBar(

            SnackBar(

              content: const Text('Awesome Snackbar!'),

              action: SnackBarAction(

                // action button

                label: 'Action',

                onPressed: () {

                  // Code to execute.

                },

              ),

            ),

          );

        },

      ),

Example 3

 child: ElevatedButton(

        child: const Text('Show Snackbar'),

        onPressed: () {

          ScaffoldMessenger.of(context).showSnackBar(

            SnackBar(

              action: SnackBarAction(

                label: 'Action',

                onPressed: () {

                  // Code to execute.

                },

              ),

              content: const Text('Awesome SnackBar!'),

              duration: const Duration(

                  milliseconds: 1500), // SnackBar display duration.

              // width:250.0, // Width of the SnackBar. //  Width and margin can not be used together

              padding: const EdgeInsets.symmetric(

                horizontal: 20.0, // Inner padding for SnackBar content.

              ),

              margin: const EdgeInsets.all(

                  20), //  Width and margin can not be used together

              behavior: SnackBarBehavior.floating,

              shape: RoundedRectangleBorder(

                borderRadius: BorderRadius.circular(10.0),

              ),

            ),

          );

        },

      ),

## DropDownButton

[DropdownButton](https://api.flutter.dev/flutter/material/DropdownButton/DropdownButton.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[DropdownMenuItem](https://api.flutter.dev/flutter/material/DropdownMenuItem-class.html)<T>>? items, [DropdownButtonBuilder](https://api.flutter.dev/flutter/material/DropdownButtonBuilder.html)? selectedItemBuilder, T? value, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? hint, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? disabledHint, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<T?>? onChanged, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onTap, [int](https://api.flutter.dev/flutter/dart-core/int-class.html) elevation = 8, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? style, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? underline, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? icon, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? iconDisabledColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? iconEnabledColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) iconSize = 24.0, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isDense = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isExpanded = false, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? itemHeight = kMinInteractiveDimension, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? focusColor, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? dropdownColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? menuMaxHeight, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enableFeedback, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html) alignment = AlignmentDirectional.centerStart, [BorderRadius](https://api.flutter.dev/flutter/painting/BorderRadius-class.html)? borderRadius, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding})

Creates a dropdown button.

A DropDownButton is a material design button. The DropDownButton is a widget that we can use to select one unique value from a set of values. It lets the user select one value from a number of items. The default value shows the currently selected value. We can even include a down arrow icon on the list. On clicking the DropDownButton it opens a list of items, from which the user can select the desired option.

(DropdownMenu that is preferred for applications that are configured for Material 3)

Example 1

class \_MyDropDownButton0State extends State<MyDropDownButton0> {

  String selectedOption = 'Option 1'; // Giá trị mặc định được chọn

  @override

  Widget build(BuildContext context) {

    return Center(

      child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        children: <Widget>[

          DropdownButton<String>(

            value: selectedOption, // set value mặc định

            onChanged: (String? newValue) {

              // required ValueChanged<T?>? onChanged - xử lý event khi click chọn thay đổi item

              setState(() {

                selectedOption = newValue!;

              });

            },

            //  required List<DropdownMenuItem<T>>? items - define various items that are to be defined in dropdown menu/list

            items: <String>['Option 1', 'Option 2', 'Option 3']

                .map<DropdownMenuItem<String>>((String value) {

              return DropdownMenuItem<String>(

                value: value,

                child: Text(value),

              );

            }).toList(),

          ),

          Container(

            alignment: Alignment.bottomRight,

            child: Text(

              'Bạn đã chọn: $selectedOption',

              style: const TextStyle(fontSize: 18),

            ),

          ),

        ],

      ),

    );

  }

}

Example 2

class MyDropDownButton1 extends StatefulWidget {

  const MyDropDownButton1({super.key});

  @override

  State<MyDropDownButton1> createState() => \_MyDropDownButton1();

}

const List<String> list = <String>['One', 'Two', 'Three', 'Four'];

String dropdownValue = list.first;

class \_MyDropDownButton1 extends State<MyDropDownButton1> {

  String selectedOption = dropdownValue; // Giá trị mặc định được chọn

  @override

  Widget build(BuildContext context) {

    return Center(

      child: Column(

        children: [

          DropdownButton<String>(

            value: dropdownValue,

            icon: const Icon(Icons.arrow\_downward),

            elevation: 16,

            style: const TextStyle(color: Colors.deepPurple),

            underline: Container(

              height: 2,

              color: Colors.deepPurpleAccent,

            ),

            disabledHint: const Text('still disable'),

            hint: const Text('click choice option'),

            // onChanged: (null), // dropdown button will be disabled

            onChanged: (String? value) {

              // This is called when the user selects an item.

              setState(() {

                dropdownValue = value!;

                selectedOption = value;

              });

            },

            items: list.map<DropdownMenuItem<String>>((String value) {

              return DropdownMenuItem<String>(

                value: value,

                child: Text(value),

              );

            }).toList(),

          ),

          Container(

            alignment: Alignment.bottomRight,

            child: Text(

              'Bạn đã chọn: $selectedOption',

              style: const TextStyle(fontSize: 18),

            ),

          ),

        ],

      ),

    );

  }

}

## [DropdownMenu](https://api.flutter.dev/flutter/material/DropdownMenu/DropdownMenu.html)

[DropdownMenu](https://api.flutter.dev/flutter/material/DropdownMenu/DropdownMenu.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) enabled = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? menuHeight, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? leadingIcon, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? trailingIcon, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? label, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? hintText, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? helperText, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? errorText, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? selectedTrailingIcon, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) enableFilter = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) enableSearch = true, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? textStyle, [InputDecorationTheme](https://api.flutter.dev/flutter/material/InputDecorationTheme-class.html)? inputDecorationTheme, [MenuStyle](https://api.flutter.dev/flutter/material/MenuStyle-class.html)? menuStyle, [TextEditingController](https://api.flutter.dev/flutter/widgets/TextEditingController-class.html)? controller, T? initialSelection, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<T?>? onSelected, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? requestFocusOnTap, required [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[DropdownMenuEntry](https://api.flutter.dev/flutter/material/DropdownMenuEntry-class.html)<T>> dropdownMenuEntries})

Creates a const [DropdownMenu](https://api.flutter.dev/flutter/material/DropdownMenu-class.html).

A dropdown menu that can be opened from a TextField. The selected menu item is displayed in that field.

This widget is used to help people make a choice from a menu and put the selected item into the text input field. People can also filter the list based on the text input or search one item in the menu list.

The menu is composed of a list of DropdownMenuEntrys. People can provide information, such as: label, leading icon or trailing icon for each entry. The TextField will be updated based on the selection from the menu entries. The text field will stay empty if the selected entry is disabled.

The dropdown menu can be traversed by pressing the up or down key. During the process, the corresponding item will be highlighted and the text field will be updated. Disabled items will be skipped during traversal.

The menu can be scrollable if not all items in the list are displayed at once.

Example 1

class DropDownMenu0 extends StatefulWidget {

  const DropDownMenu0({super.key});

  @override

  State<DropDownMenu0> createState() => \_DropDownMenu0State();

}

String selectedOption = 'Option 1';

class \_DropDownMenu0State extends State<DropDownMenu0> {

  @override

  Widget build(BuildContext context) {

    return Center(

      child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        children: <Widget>[

          DropdownMenu<String>(

            requestFocusOnTap: true, // để input text được trên virtual device

            initialSelection:

                selectedOption, // T? initialSelection - giá trị default

            onSelected: (String? newValue) {

              setState(() {

                selectedOption = newValue!;

              });

            },

            // required List<DropdownMenuEntry<T>> dropdownMenuEntries - build list items

            dropdownMenuEntries: <String>['Option 1', 'Option 2', 'Option 3']

                .map<DropdownMenuEntry<String>>((String value) {

              return DropdownMenuEntry<String>(value: value, label: value);

            }).toList(),

          ),

          Text(

            'Bạn đã chọn: $selectedOption',

            style: const TextStyle(fontSize: 18),

          ),

        ],

      ),

    );

  }

}

Example 2

class DropDownMenu1 extends StatefulWidget {

  const DropDownMenu1({super.key});

  @override

  State<DropDownMenu1> createState() => \_DropDownMenu1State();

}

const List<String> list = <String>['One', 'Two', 'Three', 'Four'];

class \_DropDownMenu1State extends State<DropDownMenu1> {

  String dropdownValue = list.first;

  int index = -1;

  @override

  Widget build(BuildContext context) {

    return DropdownMenu<String>(

      requestFocusOnTap: true, // để input text được trên virtual device

      initialSelection: list.first, // T? initialSelection - giá trị default

      label: const Text('Select item'), // Label cho DropdownMenu box

      leadingIcon: const Icon(

          Icons.select\_all), // icon ngay trước(liền trái) DropdownMenu box

      onSelected: (String? value) {

        // This is called when the user selects an item.

        setState(() {

          dropdownValue = value!;

        });

      },

      // required List<DropdownMenuEntry<T>> dropdownMenuEntries - build list items

      dropdownMenuEntries: list.map<DropdownMenuEntry<String>>((String value) {

        if (index < \_iconList.length - 1) {

          index++; // Lấy giá trị index và tăng nó sau đó

        }

        return DropdownMenuEntry<String>(

            value:

                value, // This value must be unique across all entries in a [DropdownMenu].

            label: value, // The label displayed in the center of the menu item.

            leadingIcon:

                Icon(\_iconList[index]), // icon ngay trước(liền trái) item

            trailingIcon:

                const Icon(Icons.check\_outlined)); // icon sát lề phải item

      }).toList(),

    );

  }

}

## Slider

[Slider](https://api.flutter.dev/flutter/material/Slider/Slider.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [double](https://api.flutter.dev/flutter/dart-core/double-class.html) value, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? secondaryTrackValue, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>? onChanged, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>? onChangeStart, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>? onChangeEnd, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) min = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) max = 1.0, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? divisions, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? label, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? inactiveColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? secondaryActiveColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? thumbColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [SemanticFormatterCallback](https://api.flutter.dev/flutter/material/SemanticFormatterCallback.html)? semanticFormatterCallback, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [SliderInteraction](https://api.flutter.dev/flutter/material/SliderInteraction.html)? allowedInteraction})

Creates a Material Design slider.

*const*

[Slider.adaptive](https://api.flutter.dev/flutter/material/Slider/Slider.adaptive.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [double](https://api.flutter.dev/flutter/dart-core/double-class.html) value, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? secondaryTrackValue, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>? onChanged, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>? onChangeStart, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>? onChangeEnd, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) min = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) max = 1.0, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? divisions, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? label, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? inactiveColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? secondaryActiveColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? thumbColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [SemanticFormatterCallback](https://api.flutter.dev/flutter/material/SemanticFormatterCallback.html)? semanticFormatterCallback, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [SliderInteraction](https://api.flutter.dev/flutter/material/SliderInteraction.html)? allowedInteraction})

Creates an adaptive [Slider](https://api.flutter.dev/flutter/material/Slider-class.html) based on the target platform, following Material design's [Cross-platform guidelines](https://material.io/design/platform-guidance/cross-platform-adaptation.html).

Slider là một widget trong Flutter cho phép người dùng chọn giá trị từ một khoảng đã cho bằng cách trượt qua một dải giá trị. Nó thường được sử dụng để điều chỉnh các giá trị như âm lượng, độ sáng, hay bất kỳ giá trị nào có thể nằm trong một khoảng liên tục.

Slider cung cấp cho người dùng một thanh trượt mà họ có thể kéo qua trái hoặc phải để chọn giá trị trong khoảng đã cho. Khi giá trị thay đổi, một hàm gọi lại (callback) được gọi để thông báo về thay đổi này.

class \_SliderExampleState extends State<SliderExample> {

// định nghĩa biến lưu giá trị ban đầu

  var \_sliderValue = 0.0;

  double \_currentSliderValue = 20.0;

  double \_currentSliderValue1 = 20.0;

  @override

  Widget build(BuildContext context) {

    return Column(

      children: [

        const Divider(),

        Slider(

          value: \_sliderValue,

          min: 0,

          max: 100,

          divisions: 20, // chia space trượt ra bao nhiêu phần

          label: \_sliderValue.round().toString(), // phải set divisions: mới có tác dụng

          onChanged: (newValue) {

            setState(() {

              \_sliderValue = newValue;

            });

          },

        ),

        const Divider(),

        Slider(

          value: \_currentSliderValue,

          max: 100,

          divisions: 10, // chia space trượt ra bao nhiêu phần

          label: \_currentSliderValue.round().toString(),

          onChanged: (double value) {

            setState(() {

              \_currentSliderValue = value;

            });

          },

        ),

        const Divider(),

        Slider(

          value: \_currentSliderValue1,

          max: 100,

          divisions: 10, // chia space trượt ra bao nhiêu phần

          label: \_currentSliderValue1.round().toString(),

          onChanged: (double value) {

            setState(() {

              \_currentSliderValue1 = value;

            });

          },

          thumbColor: Colors.amber, // color of nút

          activeColor: Colors.blue, // Màu của vùng slider đã chọn

          inactiveColor: Colors.grey, // Màu của vùng slider chưa chọn

      ],

    );

  }

}

## Center

[Center](https://api.flutter.dev/flutter/widgets/Center/Center.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? widthFactor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? heightFactor, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Center là một widget đặt widget con duy nhất của nó tại trung tâm của nó.

Center và Align khá giống nhau, chúng chỉ có duy nhất một widget con, nhưng Align cho phép bạn tùy biến vị trí của widget con bên trong nó.

Nếu tham số widthFactor không được chỉ định thì chiều rộng của Center sẽ lớn nhất có thể, ngược lại chiều rộng của Center bằng chiều rộng của child nhân với widthFactor. Tham số heightFactor cũng có hành vi tương tự đối với chiều cao của Center. Như vậy theo mặc định kích thước của Center sẽ lớn nhất có thể.

return Container(

      decoration: BoxDecoration(

        border: Border.all(

          color: Colors.black,

          width: 2.0,

        ),

      ),

      child: Center(

        widthFactor: 3, // width = width object con \* widthFactor

        heightFactor: 5, // height = height object con \* heightFactor

        child: ElevatedButton(

          onPressed: () {},

          child: const Text('Hello, World!'),

        ),

      ),

    );

## Container

[Container](https://api.flutter.dev/flutter/widgets/Container/Container.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html)? alignment, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [Decoration](https://api.flutter.dev/flutter/painting/Decoration-class.html)? decoration, [Decoration](https://api.flutter.dev/flutter/painting/Decoration-class.html)? foregroundDecoration, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? height, [BoxConstraints](https://api.flutter.dev/flutter/rendering/BoxConstraints-class.html)? constraints, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? margin, [Matrix4](https://api.flutter.dev/flutter/vector_math_64/Matrix4-class.html)? transform, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html)? transformAlignment, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none})

Container trong Flutter được sử dụng để tạo một khung chứa (container) để bao bọc và tổ chức các widget con bên trong. Nó cung cấp nhiều thuộc tính cho phép adds padding, margins, borders, background color, or other decorations to a widget.. Mục đích chính của việc sử dụng widget Container là để thiết lập và tùy chỉnh hình dáng và vị trí của các phần tử trong giao diện ứng dụng của bạn.

// Set color nền - border – corlor border qua decoration: BoxDecoration

  child: Container(

                  decoration: BoxDecoration(

                      color: const Color.fromARGB(255, 246, 235, 201),// màu nền

                      border: Border.all(

                        color: Colors.red, // Màu của viền

                        width: 1.5, // Độ dày của viền

                      )),

                  // color: const Color.fromRGBO(239, 184, 184, 1),

                  child: Text('Item $index')));

        }));

// Container còn thường được wrap ngoài các widget khác để decor cho nó

* Height, Width:
  + Default: nếu ko có child - bao phủ full screen; nếu có child - tự động kéo vừa kích thước object con
  + Nếu set width, height: container giữ cứng kích thước này.(nếu nó là con của widget khác, nếu nó là root widget thì nó sẽ bung full screen)
* Nếu Container được chỉ định child, và không được chỉ định width, height, constraints và alignment nó sẽ định kích thước của nó nhỏ nhất có thể. Tuy nhiên nếu Container được chỉ định child và alignment nhưng không được chỉ định width, height, constraints nó sẽ định kích thước của nó lớn nhất có thể.

Widget build(BuildContext context) {

    return MaterialApp(

        home: Scaffold(

      appBar: AppBar(title: Text('Flutter Container')),

      body: Container(

        color: Colors.amberAccent,

        width: 400,

        height: 400,

        child: Text('This is container'),

      ),

    ));

  }

* Constraints:

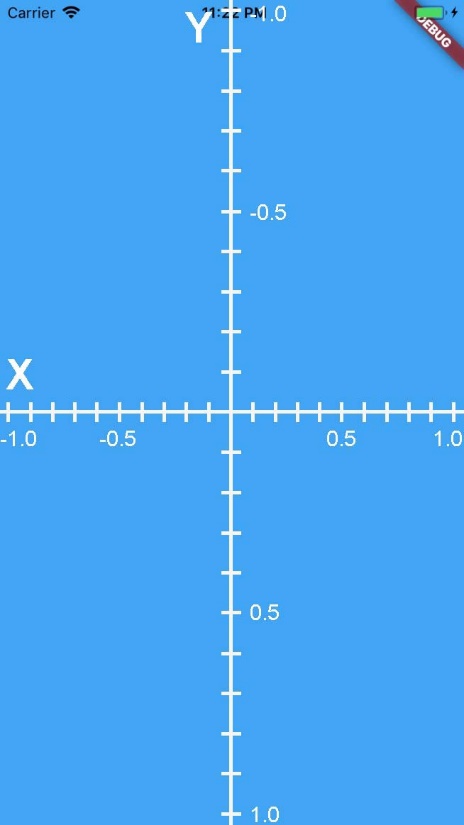
thuộc tính constraints của widget Container được sử dụng để xác định các ràng buộc về kích thước của widget. Bằng cách sử dụng constraints, bạn có thể giới hạn kích thước tối đa và tối thiểu của một widget Container.

(khác với width, height là set cứng size)

 constraints: const BoxConstraints(

          maxHeight: 200, maxWidth: 200, minHeight: 100, minWidth: 100),

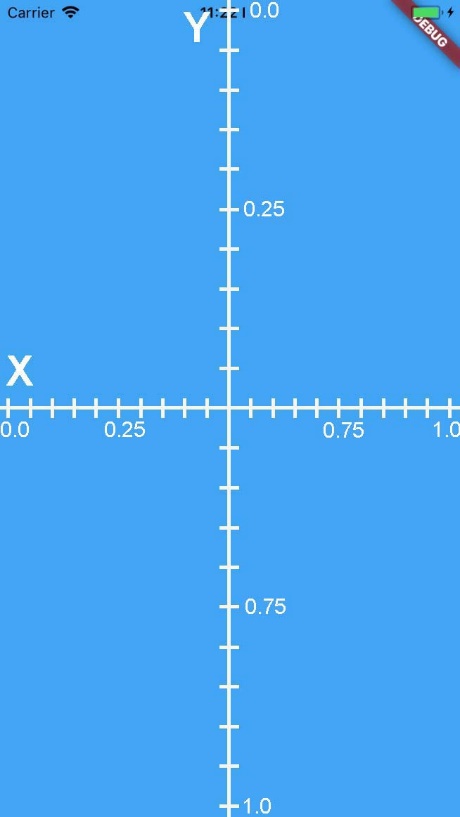
* alignment: set vị trí object con so với container theo các value constant or hệ (x,y) như ảnh dưới: gốc toạ độ (0,0) nằm giữa màn hình



  alignment: Alignment.centerLeft, // alignment object con so với khung container = constant

  alignment: Alignment(-1, -1), // alignment object con so với khung container = (x,y)

set vị trí object con so với container theo hệ FractionalOffset(x,y): x chạy từ left to right (0 -> 1), y chạy từ top to bottom (0 ->1)



  alignment: FractionalOffset(0.25, 0.25),

* padding & margin:

padding: EdgeInsets.all(20), // padding object con với khung container. nếu có gán alignment thì padding += alignment

margin: EdgeInsets.all(20), // margin của container này với object cha chứa nó

* child: chứa nội dung trong container
* decoration: các thuộc tính định dạng container nâng cao.
* decoration: BoxDecoration(
* color: const Color.fromARGB(255, 43, 221, 90).withOpacity(
* 0.3), // ko thể set color cùng lúc với color trước đó của container (chỉ có thể chọn 1 trong 2)
* border: Border.all(
* width: 2,
* color: Colors.red,
* ),
* borderRadius: BorderRadius.circular(20), // or borderRadius: const BorderRadius.all(Radius.circular(20)),
* shape: BoxShape.circle,
* /\*
* Lưu ý khi sử dụng BoxShape.circle và borderRadius cùng lúc sẽ báo lỗi
* Exception has occurred.
* \_AssertionError
* ('package:flutter/src/painting/box\_decoration.dart': Failed assertion: line 128 pos 12: 'shape != BoxShape.circle || borderRadius == null': is not true.)
* --> bỏ set 1 trong 2
* \*/
* boxShadow: const [
* BoxShadow(
* color: Color.fromARGB(255, 109, 239, 22), // Màu của bóng
* offset: Offset(3, 3), // Vị trí tương đối của bóng (ngang, dọc)
* blurRadius: 20, // Độ mờ của bóng
* spreadRadius: 0, // Độ lan tỏa của bóng
* ),
* ],
* backgroundBlendMode: BlendMode.exclusion, // Chế độ blend – trộn màu
* image: const DecorationImage( // set ảnh nền
* image: AssetImage('assets/image3.jpg'), // Đường dẫn hình ảnh
* fit: BoxFit
* .cover, // Cách hình ảnh được hiển thị (cover, contain, ...)
* ),

// gradient: hiệu ứng loang màu. Có 3 loại: SweepGradient, RadialGradient và LinearGradient

gradient: SweepGradient(

* colors: [Colors.red, Colors.blue],
* center: Alignment.center, // Điểm tâm gradient
* startAngle: 0, // Góc bắt đầu gradient (độ)
* endAngle: 3.14, // Góc kết thúc gradient (độ)
* ),
* // or
* gradient: RadialGradient(
* colors: [Colors.yellow, Colors.orange],
* center: Alignment.center, // Điểm tâm gradient
* radius: 0.8, // Bán kính gradient
* ),
* // or
* gradient: LinearGradient(
* colors: [
* Colors.blue,
* Colors.purple
* ], // Danh sách màu tạo gradient
* begin: Alignment.topLeft, // Điểm bắt đầu gradient
* end: Alignment.bottomRight, // Điểm kết thúc gradient
* ),
* ),

## Card

[Card](https://api.flutter.dev/flutter/material/Card/Card.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? shadowColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? surfaceTintColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? elevation, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)? shape, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) borderOnForeground = true, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? margin, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html)? clipBehavior, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) semanticContainer = true})

Creates a Material Design card.

A Material Design card: a panel with slightly rounded corners and an elevation shadow.

A card is a sheet of Material used to represent some related information, for example an album, a geographical location, a meal, contact details, etc.

Card contains related nuggets of information and can be composed from almost any widget, but is often used with ListTile. Card has a single child, but its child can be a column, row, list, grid, or other widget that supports multiple children. By default, a Card shrinks its size to 0 by 0 pixels. You can use SizedBox to constrain the size of a card.

Example 1 – trong Card chứa các container khác

 Widget build(BuildContext context) {

    return Center(

      child: Card(

        child: Column(

          mainAxisSize: MainAxisSize.min,

          children: <Widget>[

            const ListTile(

              leading: Icon(Icons.album),

              title: Text('The Enchanted Nightingale'),

              subtitle: Text('Music by Julie Gable. Lyrics by Sidney Stein.'),

            ),

            Row(

              mainAxisAlignment: MainAxisAlignment.end,

              children: <Widget>[

                TextButton(

                  child: const Text('BUY TICKETS'),

                  onPressed: () {/\* ... \*/},

                ),

                const SizedBox(width: 10),

                TextButton(

                  child: const Text('LISTEN'),

                  onPressed: () {/\* ... \*/},

                ),

                const SizedBox(width: 10),

              ],

            ),

          ],

        ),

      ),

    );

Example 2 – Container chứa nhiều Card

Các thuộc tính

Nếu muốn tuỳ biến kích thước của Card hãy đặt nó trong một Container hoặc SizedBox.

 Widget build(BuildContext context) {

    return Column(

      children: [

        Card(

          margin: EdgeInsets.all(50),

          color: Color.fromARGB(255, 200, 245, 201),

          shadowColor: Color.fromARGB(255, 139, 96, 130),

          surfaceTintColor: Colors

              .amber, // trộn với color:, 3 thuộc tính color - surfaceTintColor - elevation có ảnh hưởng lẫn nhau

          elevation: 30,

          // shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(50)),

          //  shape: BeveledRectangleBorder(borderRadius: BorderRadius.circular(20)),

          shape: StadiumBorder(side: BorderSide(width: 2)), // kiểu border

          child: Column(

            mainAxisSize: MainAxisSize.min,

            children: <Widget>[

              ListTile(

                leading: Icon(Icons.album, color: Colors.cyan, size: 45),

                title: Text(

                  "Let's Talk About Love",

                  style: TextStyle(fontSize: 20),

                ),

                subtitle: Text('Modern Talking Album'),

              ),

            ],

          ),

        ),

        // Nếu bạn muốn tuỳ biến kích thước của Card hãy đặt nó trong một Container hoặc SizedBox.

        SizedBox(

          width: 500,

          height: 200,

          child: Card(

            margin: EdgeInsets.all(50),

            color: Color.fromARGB(255, 200, 245, 201),

            shadowColor: Color.fromARGB(255, 139, 96, 130),

            surfaceTintColor: Colors

                .amber, // trộn với color:, 3 thuộc tính color - surfaceTintColor - elevation có ảnh hưởng lẫn nhau

            elevation: 50,

// shape: kiểu border – có các cách cài đặt sau

            // shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(50)),

            //  shape: BeveledRectangleBorder(borderRadius: BorderRadius.circular(20)),

            shape: StadiumBorder(side: BorderSide(width: 2)), // kiểu border

            /\*

            borderOnForeground của widget Card được sử dụng để xác định liệu viền của Card có được vẽ lên phần nền của các phần tử con bên trong hay không. Nếu bạn đặt borderOnForeground thành true, viền của Card sẽ được vẽ đè lên nền của các phần tử con. Nếu bạn đặt nó thành false, nền của các phần tử con sẽ được đè lên phía trên viền của Card. (phải set màu nền, viền cho các object liên quan mới test được)

             \*/

            borderOnForeground: false, //

            child: Column(

              mainAxisSize: MainAxisSize.min,

              children: <Widget>[

// dùng ListTileTheme để setup định dạng cho ListTile

                ListTileTheme(

                  shape: RoundedRectangleBorder(

                    borderRadius: BorderRadius.circular(10.0),

                    side: BorderSide(

                        color: Color.fromARGB(255, 79, 22, 236), width: 2.0),

                  ),

                  child: ListTile(

                    tileColor: const Color.fromARGB(255, 243, 33, 180),

                    leading: Icon(Icons.album, color: Colors.cyan, size: 45),

                    title: Text(

                      "Let's Talk About Love",

                      style: TextStyle(fontSize: 20),

                    ),

                    subtitle: Text('Modern Talking Album'),

                  ),

                ),

              ],

            ),

          ),

        ),

      ],

    );

  }

Property color được sử dụng để sét đặt mầu nền cho Card.

Nếu property này là null thì CardTheme.color của ThemeData.cardTheme được sử dụng. Nếu CardTheme.color cũng null thì ThemeData.cardColor sẽ được sử dụng.

elevation là toạ độ theo trục Z của Card, nó có ảnh hưởng tới kích thước của bóng đổ (shadow) của Card.

Nếu property này null thì CardTheme.elevation của ThemeData.cardTheme được sử dụng. Nếu CardTheme.elevation cũng null thì giá trị mặc định là 1.0.

Property shape được sử dụng để định nghĩa hình dạng đường viền của Card.

Nếu property này là null thì CardTheme.shape của ThemeData.cardTheme sẽ được sử dụng. Nếu CardTheme.shape cũng null thì hình dạng sẽ là RoundedRectangleBorder với bán kính góc tròn là 4.0.

## Wrap

[Wrap](https://api.flutter.dev/flutter/widgets/Wrap/Wrap.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) direction = Axis.horizontal, [WrapAlignment](https://api.flutter.dev/flutter/rendering/WrapAlignment.html) alignment = WrapAlignment.start, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) spacing = 0.0, [WrapAlignment](https://api.flutter.dev/flutter/rendering/WrapAlignment.html) runAlignment = WrapAlignment.start, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) runSpacing = 0.0, [WrapCrossAlignment](https://api.flutter.dev/flutter/rendering/WrapCrossAlignment.html) crossAxisAlignment = WrapCrossAlignment.start, [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection, [VerticalDirection](https://api.flutter.dev/flutter/painting/VerticalDirection.html) verticalDirection = VerticalDirection.down, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[]})

Widget này gần tương tự với [**Flex**](https://api.flutter.dev/flutter/widgets/Flex-class.html)(lớp cha của Row và Column), nhưng khác ở điểm có thể tự sắp xếp các widget con sang cột / hàng mới tuỳ vào kích thước còn lại.

 Widget build(BuildContext context) {

    return Wrap(

      direction:

          Axis.vertical, // phương ngang hay dọc. default is Axis.horizontal

      // textDirection: TextDirection.rtl, // chiều xếp childs object .ltr or rtl

      verticalDirection: VerticalDirection

          .down, // chiều xếp childs object up - up to down; down - up to down

      alignment: WrapAlignment

          .spaceAround, // cách xếp đặt vị trí các elements của range chưa được lấp đầy (thường là range cuối)

      crossAxisAlignment: WrapCrossAlignment

          .center, // cách gióng hàng theo phuong crossAxis của 1 range

      spacing: 8.0, // gap between adjacent chips

      runSpacing: 4.0, // gap between lines

      children: <Widget>[

        Chip(

          avatar: CircleAvatar(

              backgroundColor: Colors.blue.shade900, child: const Text('A')),

          label: const Text('Hamilton'),

        ),

        Chip(

          avatar: CircleAvatar(

              backgroundColor: Colors.blue.shade900, child: const Text('B')),

          label: const Text('Lafayette-nmson'),

        ),

        Chip(

          avatar: CircleAvatar(

              backgroundColor: Colors.blue.shade900, child: const Text('C')),

          label: const Text('Mulligan'),

        ),

        Chip(

          avatar: CircleAvatar(

              backgroundColor: Colors.blue.shade900, child: const Text('D')),

          label: const Text('Laurens'),

        ),

        Chip(

          avatar: CircleAvatar(

              backgroundColor: Colors.blue.shade900, child: const Text('E')),

          label: const Text('LafayetteS'),

        ),

        Chip(

          avatar: CircleAvatar(

              backgroundColor: Colors.blue.shade900, child: const Text('F')),

          label: const Text('MulliganS'),

        ),

        Chip(

          avatar: CircleAvatar(

              backgroundColor: Colors.blue.shade900, child: const Text('G')),

          label: const Text('LaurensS'),

        ),

      ],

    );

  }

## ColoredBox

[ColoredBox](https://api.flutter.dev/flutter/widgets/ColoredBox/ColoredBox.html)({required [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html) color, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child, [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key})

A widget that paints its area with a specified Color and then draws its child on top of that color.

\*The issue is that you are using a **Column** without any height constraints, and it's being placed inside a **ColoredBox**. The **ColoredBox** doesn't provide constraints to its child, and therefore, the **Column** tries to take as much vertical space as it needs, leading to the overflow error.

 Widget build(BuildContext context) {

    return const ColoredBox(

      color: Colors.red,

      child: SizedBox(

        width: 100,

        height: 100,

        child: Text('Hello'),

      ),

    );

  }

## Align

[Align](https://api.flutter.dev/flutter/widgets/Align/Align.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html) alignment = Alignment.center, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? widthFactor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? heightFactor, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates an alignment widget.

*Const*

Align widget dùng căn chỉnh object con bên trong nó và tùy chọn thay đổi kích thước dựa trên kích thước của object con.

 child: Column(

        children: [

          Container(

            decoration: BoxDecoration(

                color: const Color.fromARGB(255, 233, 244, 199),

                border: Border.all(color: Colors.black, width: 2)),

            // height: 200.0,

            // width: 200.0,

            // color: const Color.fromARGB(255, 233, 244, 199),

            child: const Align(

              alignment: Alignment.bottomLeft,

              // widthFactor: 3 - width container = 3 \* width FlutterLogo

              // 2 thuộc tính height/width của container phải not set

              widthFactor: 2,

              heightFactor: 2,

              child: FlutterLogo(

                size: 100,

              ),

            ),

          ),

          Container(

            decoration: BoxDecoration(

                color: const Color.fromARGB(255, 233, 244, 199),

                border: Border.all(color: Colors.black, width: 2)),

            height: 200.0,

            width: 200.0,

            child: const Align(

              alignment:

                  Alignment(0.2, 0.6), // truyền value double Alignment(x, y)

              child: FlutterLogo(

                size: 50,

              ),

            ),

          ),

          Container(

            decoration: BoxDecoration(

                color: const Color.fromARGB(255, 233, 244, 199),

                border: Border.all(color: Colors.black, width: 2)),

            // height: 200.0,

            // width: 200.0,

            child: const Align(

              alignment: FractionalOffset(

                  0.5, 0.6), // truyền value double Alignment(x, y)

              child: FlutterLogo(

                size: 50,

              ),

            ),

          ),

        ],

      ),

Trong Flutter, Alignment là một lớp được sử dụng để xác định vị trí tương đối của một điểm trong một hộp chứa. Các giá trị mà bạn cung cấp cho Alignment xác định vị trí theo một hệ tọa độ 2D với tâm là (0, 0) ở giữa và phạm vi từ -1 đến 1 trong mỗi chiều.

Trong trường hợp của bạn, alignment: Alignment(0.2, 0.6) có nghĩa là bạn đang xác định một điểm nằm ở 20% khoảng cách từ trái đến phải của hộp chứa và 60% khoảng cách từ trên xuống đáy của hộp chứa.

Để biểu diễn trực quan hơn, hãy tưởng tượng hộp chứa có kích thước width: 100 và height: 100. Khi bạn đặt alignment: Alignment(0.2, 0.6), điểm được đặt ở vị trí x = 20 (20% của 100) và y = 60 (60% của 100) trên hộp chứa này.

Tóm lại, Alignment(0.2, 0.6) định nghĩa vị trí tương đối của một điểm bên trong một hộp chứa và dựa trên hệ tọa độ có phạm vi từ -1 đến 1.

 // Set toạ độ theo Alignment(x, y)

/// The top left corner.

  static const Alignment topLeft = Alignment(-1.0, -1.0);

  /// The center point along the top edge.

  static const Alignment topCenter = Alignment(0.0, -1.0);

  /// The top right corner.

  static const Alignment topRight = Alignment(1.0, -1.0);

  /// The center point along the left edge.

  static const Alignment centerLeft = Alignment(-1.0, 0.0);

  /// The center point, both horizontally and vertically.

  static const Alignment center = Alignment(0.0, 0.0);

  /// The center point along the right edge.

  static const Alignment centerRight = Alignment(1.0, 0.0);

  /// The bottom left corner.

  static const Alignment bottomLeft = Alignment(-1.0, 1.0);

  /// The center point along the bottom edge.

  static const Alignment bottomCenter = Alignment(0.0, 1.0);

  /// The bottom right corner.

  static const Alignment bottomRight = Alignment(1.0, 1.0);

//----------------------------------------------

// Set toạ độ theo FractionalOffset(x, y)

  /// The top left corner.

  static const FractionalOffset topLeft = FractionalOffset(0.0, 0.0);

  /// The center point along the top edge.

  static const FractionalOffset topCenter = FractionalOffset(0.5, 0.0);

  /// The top right corner.

  static const FractionalOffset topRight = FractionalOffset(1.0, 0.0);

  /// The center point along the left edge.

  static const FractionalOffset centerLeft = FractionalOffset(0.0, 0.5);

  /// The center point, both horizontally and vertically.

  static const FractionalOffset center = FractionalOffset(0.5, 0.5);

  /// The center point along the right edge.

  static const FractionalOffset centerRight = FractionalOffset(1.0, 0.5);

  /// The bottom left corner.

  static const FractionalOffset bottomLeft = FractionalOffset(0.0, 1.0);

  /// The center point along the bottom edge.

  static const FractionalOffset bottomCenter = FractionalOffset(0.5, 1.0);

  /// The bottom right corner.

  static const FractionalOffset bottomRight = FractionalOffset(1.0, 1.0);

## SizedBox

[SizedBox](https://api.flutter.dev/flutter/widgets/SizedBox/SizedBox.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? height, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a fixed size box. The [width](https://api.flutter.dev/flutter/widgets/SizedBox/width.html) and [height](https://api.flutter.dev/flutter/widgets/SizedBox/height.html) parameters can be null to indicate that the size of the box should not be constrained in the corresponding dimension.

*const*

[SizedBox.expand](https://api.flutter.dev/flutter/widgets/SizedBox/SizedBox.expand.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a box that will become as large as its parent allows.

*const*

[SizedBox.fromSize](https://api.flutter.dev/flutter/widgets/SizedBox/SizedBox.fromSize.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child, [Size](https://api.flutter.dev/flutter/dart-ui/Size-class.html)? size})

Creates a box with the specified size.

[SizedBox.shrink](https://api.flutter.dev/flutter/widgets/SizedBox/SizedBox.shrink.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a box that will become as small as its parent allows.

*const*

[SizedBox.square](https://api.flutter.dev/flutter/widgets/SizedBox/SizedBox.square.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? dimension})

Creates a box whose [width](https://api.flutter.dev/flutter/widgets/SizedBox/width.html) and [height](https://api.flutter.dev/flutter/widgets/SizedBox/height.html) are equal.

*const*

SizedBox là một hộp trong suốt, khác với Container bạn không thể thiết lập kiểu dáng cho nó (chẳng hạn mầu nền, margin, padding,...). Nếu bạn chỉ định một kích thước cụ thể cho SizedBox, kích thước đó cũng sẽ được áp dụng cho widget con của nó. Ngược lại nếu chiều rộng của SizedBox không được chỉ định hoặc null thì widget con của sẽ có chiều rộng theo thiết lập riêng hoặc bằng 0 (nếu không được thiết lập). Với chiều cao cũng có hành vi tương tự.

Tất cả các tham số tham gia vào việc tạo ra một SizedBox như width, height, size, child đều là các tuỳ chọn (Không bắt buộc).

return MaterialApp(

      home: Scaffold(

          appBar: AppBar(title: const Text('Flutter SizedBox')),

          body: SizedBox(

            width: 300,

            height: double

                .maxFinite, // double.infinity - giá trị vô cực, full hết màn hình

            /\*

            cực âm: double.negativeInfinity, double.minPositive

            cực dương: double.infinity, double.maxFinite

             \*/

            child: Container(

              color: Colors.blue,

              child: const Center(

                child: Text('SizedBox Example'),

              ),

            ),

          )),

    );

  }

Một ứng dụng thường được sử dụng là chèn SizedBox vào làm khoảng cách giữa các object khác

  Widget build(BuildContext context) {

    return MaterialApp(

      home: Scaffold(

        appBar: AppBar(title: const Text('Flutter SizedBox Space')),

        body: const Row(children: [

          Text(

            'text 001',

            style: TextStyle(backgroundColor: Colors.deepOrange),

          ),

          SizedBox(

            width: 10,

          ),

          Text(

            'text 011',

            style: TextStyle(backgroundColor: Colors.deepOrange),

          ),

          SizedBox(

            width: 10,

          ),

          Text(

            'text 101',

            style: TextStyle(backgroundColor: Colors.deepOrange),

          ),

          SizedBox(

            width: 10,

          ),

          Text(

            'text 111',

            style: TextStyle(backgroundColor: Colors.deepOrange),

          ),

        ]),

      ),

    );

  }

Constructor SizedBox.expand được sử dụng để tạo ra một SizedBox với kích thước lớn nhất theo chỉ định từ widget cha của nó.

 Widget build(BuildContext context) {

    return Scaffold(

      body: SizedBox.expand(

        child: Container(

          color: blue,

          child: const Column(

            children: [

              Text('Hello!'),

              Text('Goodbye!'),

            ],

          ),

        ),

      ),

    );

  }

Constructor SizedBox.shrink được sử dụng để tạo ra một SizedBox với kích thước nhỏ nhất theo chỉ định từ widget cha của nó.

 Widget build(BuildContext context) {

    return MaterialApp(

      home: Scaffold(

          appBar: AppBar(title: const Text('Flutter SizedBox 2')),

          body: ConstrainedBox(

              constraints: const BoxConstraints(

                // Min: 80x20 - ghi đè lên các thuộc tính tương đương của object con

                // \*Nếu ko set values cặp thuộc tính minWidth/minHeight thì object con sẽ ko show ra

                minWidth: 200.0,

                minHeight: 200.0,

              ),

              child: SizedBox.shrink(

                child: ElevatedButton(

                    child: const Text('Button'),

                    onPressed: () {},

                    style: ElevatedButton.styleFrom(

                        backgroundColor:

                            Color.fromARGB(255, 250, 253, 66), // cũ là primary

                        foregroundColor: Color.fromARGB(163, 14, 246, 2),

                        fixedSize: const Size(300, 300), // ko tác dụng

                        minimumSize: Size(200, 200))), // ko tác dụng

              ))),

    );

  }

}

Constructor SizedBox.fromSize được sử dụng để tạo một SIzedBox với kích thước được chỉ định thông qua tham số tuỳ chọn - size.

 home: Scaffold(

            appBar: AppBar(title: const Text('Flutter SizedBox.fromSize')),

            body: SizedBox.fromSize(

              size: const Size(200, 200),

              child: ElevatedButton(

                  onPressed: () {},

                  style: ElevatedButton.styleFrom(

                    backgroundColor: const Color.fromARGB(255, 250, 253, 66), // cũ là primary

                  ),

                  child: const Text('Button SizedBox.fromSize')),

            )));

## FittedBox

[FittedBox](https://api.flutter.dev/flutter/widgets/FittedBox/FittedBox.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [BoxFit](https://api.flutter.dev/flutter/painting/BoxFit.html) fit = BoxFit.contain, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html) alignment = Alignment.center, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a widget that scales and positions its child within itself according to [fit](https://api.flutter.dev/flutter/widgets/FittedBox/fit.html).

FittedBox là một widget trong Flutter được sử dụng để điều chỉnh và điều tiết kích thước của widget con để vừa vặn trong một không gian đã cho. Nó thường được sử dụng để tỷ lệ hoặc căn chỉnh nội dung con trong một không gian bố trí.

\*Text ko đặt trong FittedBox – quá dài sẽ auto xuống dòng; Đặt trong FittedBox – điều chỉnh kích thước font sao cho vừa 1 dòng (nếu ko xuống dòng bằng '\n')

\* FittedBox chỉ có thể điều chỉnh kích thước widget con bị GIỚI HẠN (có chiều rộng và chiều cao non-infinite, nếu set infinite sẽ văng lỗi).

Khi đặt một widget con vào FittedBox, FittedBox sẽ tự động điều chỉnh kích thước của widget con để phù hợp với các giá trị của thuộc tính fit:

class FittedBoxExample extends StatelessWidget {

  const FittedBoxExample({super.key});

  @override

  Widget build(BuildContext context) {

    return Column(

      children: [

        Container(

          height: 200,

          width: 300,

          color: Colors.blue,

          child: FittedBox(

            // TRY THIS: Try changing the fit types to see how they change the way

            // the placeholder fits into the container.

            fit: BoxFit.fitHeight,

            child: Image.asset('assets/lion.jpg'),

          ),

        ),

        Container(

          height: 200,

          width: 300,

          color: Colors.blue,

          child: FittedBox(

            fit: BoxFit.fill,

            child: Image.asset('assets/lion.jpg'),

          ),

        ),

        // Container(

        //   height: 200,

        //   width: 300,

        //   color: Colors.blue,

        //   child: FittedBox(

        //     fit: BoxFit.scaleDown,

        //     child: Image.asset('assets/lion.jpg'),

        //   ),

        // ),

        Container(

          height: 200,

          width: 300,

          color: Colors.blue,

          child: FittedBox(

            fit: BoxFit.contain, // lấn sang các space lân cận: fitWidth, cover

            child: Image.asset('assets/lion.jpg'),

          ),

        ),

      ],

    );

  }

}

## ConstrainedBox

[ConstrainedBox](https://api.flutter.dev/flutter/widgets/ConstrainedBox/ConstrainedBox.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [BoxConstraints](https://api.flutter.dev/flutter/rendering/BoxConstraints-class.html) constraints, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a widget that imposes additional constraints on its child.

ConstrainedBox là một widget trong Flutter được sử dụng để giới hạn kích thước của widget con bên trong nó. Bạn có thể sử dụng ConstrainedBox để đặt giới hạn về chiều rộng, chiều cao hoặc cả hai cho widget con.

Nếu ConstrainedBox ở vị trí root(ko là con của widget nào) sẽ không set 4 tham số width, height được do bị ràng buộc từ screen áp xuống – phải wrap ConstrainedBox bằng container cha trước.

Ghi đè lên các thuộc tính tương đương của object con

Widget build(BuildContext context) {

    return MaterialApp(

      home: Scaffold(

          appBar: AppBar(title: const Text('Flutter SizedBox 2')),

          body: ConstrainedBox(

            constraints: const BoxConstraints(

              minWidth: 100, // Chiều rộng tối thiểu

              maxWidth: 200, // Chiều rộng tối đa

              minHeight: 50, // Chiều cao tối thiểu

              maxHeight: 150, // Chiều cao tối đa

            ),

            child: Container(

              color: Colors.blue,

              child: const Center(

                child: Text('ConstrainedBox Example'),

              ),

            ),

          )),

    );

  }

\*Nếu ko set values cặp thuộc tính minWidth/minHeight thì object con sẽ ko show ra

## BoxConstraints class

[BoxConstraints](https://api.flutter.dev/flutter/rendering/BoxConstraints/BoxConstraints.html)({[double](https://api.flutter.dev/flutter/dart-core/double-class.html) minWidth = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) maxWidth = double.infinity, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) minHeight = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) maxHeight = double.infinity})

Creates box constraints with the given constraints.

*const*

[BoxConstraints.expand](https://api.flutter.dev/flutter/rendering/BoxConstraints/BoxConstraints.expand.html)({[double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? height})

Creates box constraints that expand to fill another box constraints.

*const*

[BoxConstraints.loose](https://api.flutter.dev/flutter/rendering/BoxConstraints/BoxConstraints.loose.html)([Size](https://api.flutter.dev/flutter/dart-ui/Size-class.html) size)

Creates box constraints that forbid sizes larger than the given size.

[BoxConstraints.tight](https://api.flutter.dev/flutter/rendering/BoxConstraints/BoxConstraints.tight.html)([Size](https://api.flutter.dev/flutter/dart-ui/Size-class.html) size)

Creates box constraints that is respected only by the given size.

[BoxConstraints.tightFor](https://api.flutter.dev/flutter/rendering/BoxConstraints/BoxConstraints.tightFor.html)({[double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? height})

Creates box constraints that require the given width or height.

*const*

[BoxConstraints.tightForFinite](https://api.flutter.dev/flutter/rendering/BoxConstraints/BoxConstraints.tightForFinite.html)({[double](https://api.flutter.dev/flutter/dart-core/double-class.html) width = double.infinity, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) height = double.infinity})

Creates box constraints that require the given width or height, except if they are infinite.

*const*

## UnconstrainedBox

[UnconstrainedBox](https://api.flutter.dev/flutter/widgets/UnconstrainedBox/UnconstrainedBox.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child, [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html) alignment = Alignment.center, [Axis](https://api.flutter.dev/flutter/painting/Axis.html)? constrainedAxis, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none})

Creates a widget that imposes no constraints on its child, allowing it to render at its "natural" size. If the child overflows the parents constraints, a warning will be given in debug mode.

Ngược lại với ConstrainedBox, UnconstrainedBox không bị ràng buộc bởi screen

Widget build(BuildContext context) {

    return UnconstrainedBox(

      child: Container(color: red, width: 20, height: 50),

    );

  }

// ConstrainedBox ở vị trí root - sẽ bị screen áp full screen xuống, child ko set kích thước được

Widget build(BuildContext context) {

    return ConstrainedBox(

      constraints: const BoxConstraints(

        minWidth: 70,

        minHeight: 70,

        maxWidth: 150,

        maxHeight: 150,

      ),

      child: Container(color: red, width: 10, height: 100),

    );

  }

// UnconstrainedBox ở vị trí root - ko bị ràng buộc của screen áp xuống , child có set kích thước tuỳ ý

Widget build(BuildContext context) {

    return UnconstrainedBox(

      child: Container(color: red, width: 20, height: 50),

    );

  }

## OverflowBox

[OverflowBox](https://api.flutter.dev/flutter/widgets/OverflowBox/OverflowBox.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html) alignment = Alignment.center, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? minWidth, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? maxWidth, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? minHeight, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? maxHeight, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a widget that lets its child overflow itself.

OverflowBox is similar to UnconstrainedBox, and the difference is that it won't display any warnings if the child overflow fit the space.'

In this case the Container is 4000 pixels wide, and is too big to fit in the OverflowBox, but the OverflowBox simply shows as much as it can, with no warnings given.

(Tuy set width/height overflow ko báo lỗi sọc vàng, nhưng set với giá trị double.infinity thì app sẽ văng lỗi, treo ko render ra UI được. UnconstrainedBox cũng văng lỗi khi set double.infinity )

 Widget build(BuildContext context) {

    return OverflowBox(

      minWidth: 0,

      minHeight: 0,

      maxWidth: double.infinity,

      maxHeight: double.infinity,

      child: Container(color: red, width: 4000, height: 50),

    );

  }

## LimitedBox

[LimitedBox](https://api.flutter.dev/flutter/widgets/LimitedBox/LimitedBox.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) maxWidth = double.infinity, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) maxHeight = double.infinity, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a box that limits its size only when it's unconstrained.

Creates a box that limits its size only when it's unconstrained – Chỉ áp dụng giới hạn kích thước khi có unconstrained áp xuống nó (Điều này giải thích sự khác biệt giữa LimitedBox và ConstrainedBox)

/\*

Trường hợp này sẽ ko có lỗi, vì khi LimitedBox nhận được một kích thước không giới hạn từ UnconstrainedBox, nó truyền giới hạn chiều rộng tối đa là 100 xuống cho con của nó.

Nếu bạn thay thế UnconstrainedBox bằng một Center widget, LimitedBox sẽ không áp dụng giới hạn nữa (vì giới hạn của nó chỉ được áp dụng khi nó nhận được ràng buộc vô hạn), và chiều rộng của Container có thể tăng lên vượt quá 100.

\*/

 Widget build(BuildContext context) {

    return UnconstrainedBox(

      child: LimitedBox(

        maxWidth: 100,

        child: Container(

          color: Colors.red,

          width: double.infinity,

          height: 100,

        ),

      ),

    );

  }

}

## RotatedBox class

A widget that rotates its child by a integral number of quarter turns.

Unlike [Transform](https://api.flutter.dev/flutter/widgets/Transform-class.html), which applies a transform just prior to painting, this object applies its rotation prior to layout, which means the entire rotated box consumes only as much space as required by the rotated child.

(RotatedBox xoay object làm thay đổi layout, Transform ko ảnh hưởng layout)

RotateBox Example

//---------------

class RotatedBoxDemo extends StatefulWidget {

  const RotatedBoxDemo({super.key});

  @override

  State<RotatedBoxDemo> createState() => \_RotatedBoxDemoState();

}

class \_RotatedBoxDemoState extends State<RotatedBoxDemo> {

// Variable to track the number of rotations

// Rotations back clockwise using negative numbers

  int rotations = 0;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('RotatedBox Demo'),

      ),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.end,

          children: [

            const Text(

                'Text line 1 -  Rotate the child by the specified number of turns'),

            const Text(

                'Text line 2 -  Rotate the child by the specified number of turns'),

            const Text(

                'Text line 3 - Rotate the child by the specified number of turns'),

            const SizedBox(

              height: 20,

            ),

            RotatedBox(

              quarterTurns:

                  rotations, // Rotate the child by the specified number of turns

              child: Container(

                color: Colors.amber,

                width: 100,

                height: 50,

                child: const Center(

                  child: Text(

                    'Rotated Object',

                    style: TextStyle(color: Colors.white),

                  ),

                ),

              ),

            ),

            const SizedBox(height: 40),

            ElevatedButton(

              onPressed: () {

                // Update the rotations variable and trigger a rebuild

                setState(() {

                  rotations += 1;

                });

              },

              child: const Text('Click to rotate'),

            ),

          ],

        ),

      ),

    );

  }

}

Transform example 1

// Demo tổng hợp Transform.translate - Transform.scale - Transform.rotate

class DemoTonghop extends StatefulWidget {

  const DemoTonghop({super.key});

  @override

  State<DemoTonghop> createState() => \_DemoTonghopState();

}

class \_DemoTonghopState extends State<DemoTonghop> {

  double translateX = 0.0;

  double translateY = 0.0;

  double scaleFactor = 1;

  double myAngle = 0;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        backgroundColor: Colors.purple,

        title: const Text("Transforms"),

      ),

      body: Center(

        child: Row(

          mainAxisAlignment: MainAxisAlignment.spaceEvenly,

          crossAxisAlignment: CrossAxisAlignment.center,

          children: [

            buildTranslate(),

            buildScale(),

            buildRotate(),

          ],

        ),

      ),

    );

  }

  Widget buildTranslate() {

    return Column(

      mainAxisAlignment: MainAxisAlignment.center,

      children: [

        Transform.translate(

          // Transform.translate widget is specifying the amount by which the child (in this case, an Icon with the Icons.star symbol) will be translated or moved along the x and y axes.

          offset: Offset(translateX, translateY),

          child: const Icon(

            Icons.star,

            color: Colors.blue,

            size: 50,

          ),

        ),

        Column(

          mainAxisAlignment: MainAxisAlignment.spaceEvenly,

          children: [

            IconButton(

              icon: const Icon(Icons.keyboard\_arrow\_up),

              color: Colors.blue,

              onPressed: () {

                translate(1);

              },

            ),

            Row(

              mainAxisAlignment: MainAxisAlignment.spaceBetween,

              children: [

                IconButton(

                  icon: const Icon(

                    Icons.keyboard\_arrow\_left,

                  ),

                  color: Colors.blue,

                  onPressed: () {

                    translate(4);

                  },

                ),

                const SizedBox(

                  width: 25,

                ),

                IconButton(

                  icon: const Icon(Icons.keyboard\_arrow\_right),

                  color: Colors.blue,

                  onPressed: () {

                    translate(2);

                  },

                ),

              ],

            ),

            IconButton(

              icon: const Icon(

                Icons.keyboard\_arrow\_down,

              ),

              color: Colors.blue,

              onPressed: () {

                translate(3);

              },

            ),

          ],

        )

      ],

    );

  }

  translate(int i) {

    switch (i) {

      case 1:

        setState(() {

          translateY -= 10;

        });

        break;

      case 2:

        setState(() {

          translateX += 10;

        });

        break;

      case 3:

        setState(() {

          translateY += 10;

        });

        break;

      case 4:

        setState(() {

          translateX -= 10;

        });

        break;

      default:

        break;

    }

  }

  Widget buildScale() {

    return Column(

      mainAxisAlignment: MainAxisAlignment.center,

      children: [

        // Creates a widget that scales its child along the 2D plane.

        Transform.scale(

          scale: scaleFactor,

          child: const Icon(

            Icons.star,

            color: Colors.green,

            size: 50,

          ),

        ),

        TextButton(

          child: const Text(

            "Scale",

            style: TextStyle(color: Colors.green),

          ),

          onPressed: () {

            scale();

          },

        ),

      ],

    );

  }

  scale() async {

    /// var i - number of iterations increment scaleFactor - max scaleFactor of widget

    /// Daration - speed change scaleFactor

    for (var i = 0; i < 40; i++) {

      await Future.delayed(const Duration(milliseconds: 20), () {

        setState(() {

          scaleFactor += 0.1;

        });

      });

    }

    for (var i = 0; i < 40; i++) {

      await Future.delayed(const Duration(milliseconds: 20), () {

        setState(() {

          scaleFactor -= 0.1;

        });

      });

    }

  }

  Widget buildRotate() {

    return Column(

      mainAxisAlignment: MainAxisAlignment.center,

      children: [

        // Creates a widget that transforms its child using a rotation around the center.

        // The angle argument gives the rotation in clockwise radians.

        // rotation back clockwise using negative values

        Transform.rotate(

          angle: myAngle,

          child: const Icon(

            Icons.arrow\_upward,

            color: Colors.orange,

            size: 50,

          ),

        ),

        TextButton(

          child: const Text(

            "Rotate",

            style: TextStyle(color: Colors.orange),

          ),

          onPressed: () {

            rotate();

          },

        ),

      ],

    );

  }

  rotate() async {

    // xoay 1 vòng, mỗi nấc 1/8 đường tròn trong 0,5s

    for (var i = 0; i < 8; i++) {

      await Future.delayed(const Duration(milliseconds: 500), () {

        setState(() {

          // -math.pi / 12.0;

          // myAngle += 45.0 \* (3.14159 / 180.0);

          myAngle += 1 / 8 \* 2 \* 3.14159;

        });

      });

    }

  }

}

Transform example 2

// TransformDemo1 - Transform với thuộc tính transform

/// Creates a widget that transforms its child using a rotation around the center.

/// The angle argument gives the rotation in clockwise radians.

class TransformDemo1 extends StatefulWidget {

  const TransformDemo1({super.key});

  @override

  State<TransformDemo1> createState() => \_TransformDemo1State();

}

class \_TransformDemo1State extends State<TransformDemo1> {

  double rotationAngle = 0;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Transform1 Demo'),

      ),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            const Text('Transformed'),

            const SizedBox(height: 20),

            /// . Thuộc tính alignment được sử dụng để đặt vị trí của Transform trong Container, trong khi transform được sử dụng để xoay và làm biến đổi ma trận của Transform. Trong trường hợp này, Matrix4.skewY(0.3) sẽ làm nghiêng theo trục Y, và rotateZ(-math.pi / 12.0) sẽ xoay theo trục Z.

            Transform(

              alignment: Alignment.topCenter,

              transform: Matrix4.skewY(0.2)..rotateZ(rotationAngle),

              child: Container(

                width: 100,

                height: 100,

                color: Colors.red,

                child: const Center(

                  child: Text(

                    'Transformed Box',

                    style: TextStyle(color: Colors.white),

                  ),

                ),

              ),

            ),

            const SizedBox(height: 20),

            ElevatedButton(

              onPressed: () {

                setState(() {

                  // Xoay 45 độ mỗi lần nút được nhấn

                  rotationAngle += math.pi / 4;

                });

              },

              child: const Text('Rotate'),

            ),

          ],

        ),

      ),

    );

  }

}

## Row

[Row](https://api.flutter.dev/flutter/widgets/Row/Row.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [MainAxisAlignment](https://api.flutter.dev/flutter/rendering/MainAxisAlignment.html) mainAxisAlignment = MainAxisAlignment.start, [MainAxisSize](https://api.flutter.dev/flutter/rendering/MainAxisSize.html) mainAxisSize = MainAxisSize.max, [CrossAxisAlignment](https://api.flutter.dev/flutter/rendering/CrossAxisAlignment.html) crossAxisAlignment = CrossAxisAlignment.center, [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection, [VerticalDirection](https://api.flutter.dev/flutter/painting/VerticalDirection.html) verticalDirection = VerticalDirection.down, [TextBaseline](https://api.flutter.dev/flutter/dart-ui/TextBaseline.html)? textBaseline, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[]})

Creates a horizontal array of children.

*const*

A widget that displays its children in a horizontal array.

Nếu kích thước childs overflow screen sẽ xuất hiện cảnh báo sọc vàng.

Wrap child trong Expanded để object con tự điều chỉnh kích thước, ko còn cảnh báo sọc vàng. Object con nào được wrap bởi Expanded mới được tự điều chỉnh kích thước. Các object con khác khi tăng kích thước sẽ lấn con của Expended đến khi hết chỗ thì vẫn bị lỗi sọc vàng.

Mặc định Row widget không thể scroll được. Nếu bạn có một dòng widget và muốn chúng có thể scroll nếu không đủ chỗ, hãy xem xét sử dụng ListView Class.

Widget build(BuildContext context) {

    return Center(

        child: Row(children: [ // children: chứa list object con - List<Widget> children: const <Widget>[],

      ElevatedButton(child: const Text("BTN 1"), onPressed: () {}),

      const Icon(Icons.ac\_unit, size: 64, color: Colors.blue),

      ElevatedButton(

          onPressed: () {},

          style: ButtonStyle(

              minimumSize: MaterialStateProperty.all(const Size.square(100))),

          child: const Text("Button 2")),

      ElevatedButton(child: const Text("BTN 3"), onPressed: () {}),

    ]));

  }

To cause a child to expand to fill the available horizontal space, wrap the child in an Expanded widget.

  Widget build(BuildContext context) {

    return Container(

      width: 500,

      height: 150,

      color: Colors.amber,

      child: Row(mainAxisSize: MainAxisSize.max, children: [

        ElevatedButton(child: const Text("BTN 1"), onPressed: () {}),

        const Expanded(

          child: Icon(

            Icons.ac\_unit,

            size: 64,

            color: Colors.blue,

          ),

        ),

        Expanded(

          child: ElevatedButton(

              onPressed: () {},

              style: ButtonStyle(

                  minimumSize:

                      MaterialStateProperty.all(const Size.square(50))),

              child: const Text("Button 2")),

        ),

        ElevatedButton(child: const Text("BTN 3"), onPressed: () {}),

      ]),

    );

  }

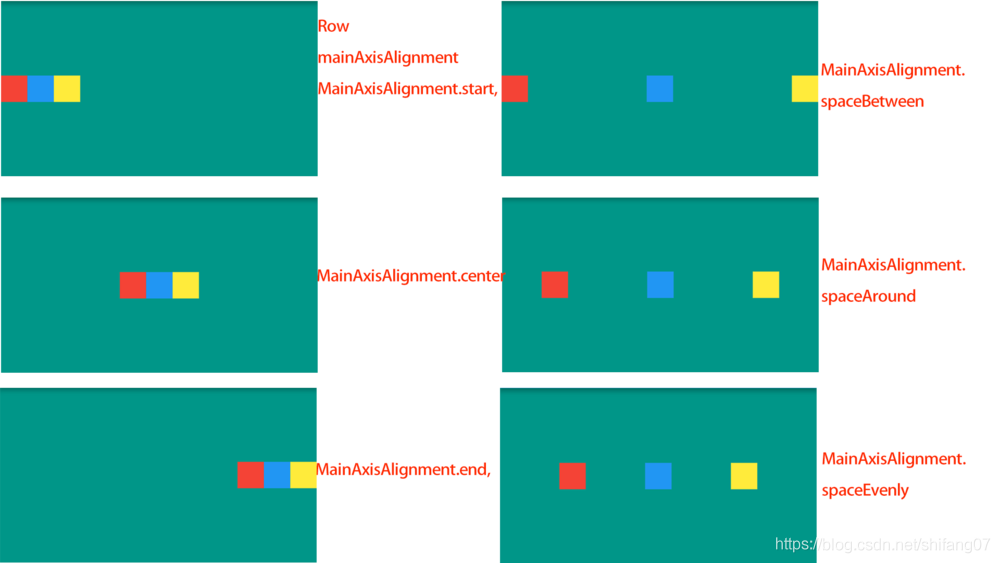
\*Expanded:

Wrap object nào với Expanded widget thì object đó sẽ bung hết value width ra kín với width còn trống của container cha.

Nếu trong row có nhiều Expanded thì các object đó chia nhau space width của container còn trống

mainAxisAlignment được sử dụng để chỉ định cách mà các widget con sẽ được bố trí trên trục chính (main axis). Đối với Row, trục chính (main axis) là chính là trục nằm ngang.

Bao gồm 6 constant với cách bố trí như sau: (Với textDirection = TextDirection.ltr (Mặc định) – left to right, .start is default)



Example

 Widget build(BuildContext context) {

    return Container(

      height: 300,

      color: const Color.fromARGB(255, 249, 122, 17),

      child: Row(mainAxisAlignment: MainAxisAlignment.spaceAround,

      children: [

        ElevatedButton(child: const Text("BTN 1"), onPressed: () {}),

        const Icon(Icons.ac\_unit, size: 64, color: Colors.blue),

        ElevatedButton(

            onPressed: () {},

            style: ButtonStyle(

                minimumSize: MaterialStateProperty.all(const Size.square(60))),

            child: const Text("Button 2")),

        ElevatedButton(child: const Text("BTN 3"), onPressed: () {}),

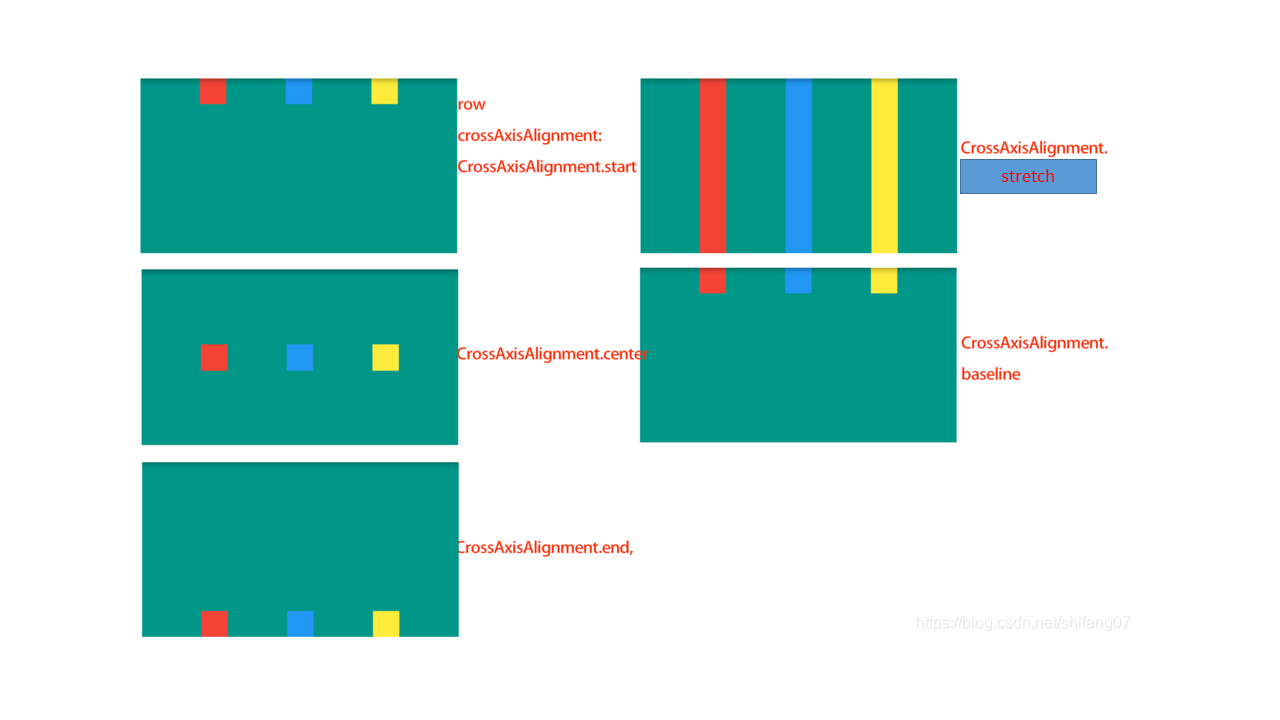
      ]),

    );

  }

crossAxisAlignment được sử dụng để chỉ định cách mà các widget con sẽ được bố trí trên trục chéo (cross axis) với trục ngang chính là trục thẳng đứng.

Với verticalDirection = VerticalDirection.down (Mặc định) – top to bottom, ta có các 6 constant với cách bố trí như sau: center is default

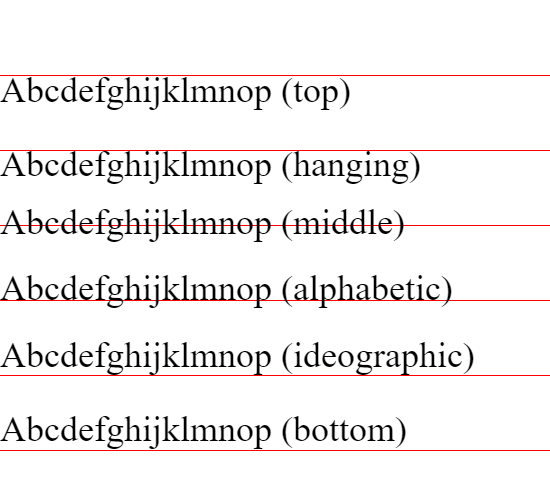


\*Lưu ý:

- CrossAxisAlignment.stretch – các object sẽ bung max height của container chứa nó

- CrossAxisAlignment.baseline – phải set kèm theo thuộc tính textBaseline: TextBaseline.ideographic or . alphabetic

Về alphabetic và ideographic:



Example

child: Row(

          crossAxisAlignment: CrossAxisAlignment.baseline,

          textBaseline: TextBaseline.alphabetic,

          children: [

            ElevatedButton(child: const Text("BTN 1"), onPressed: () {}),

            const Icon(Icons.ac\_unit, size: 64, color: Colors.blue),

            ElevatedButton(

                onPressed: () {},

                style: ButtonStyle(

                    minimumSize:

                        MaterialStateProperty.all(const Size.square(60))),

                child: const Text("Button 2")),

            ElevatedButton(child: const Text("BTN 3"), onPressed: () {}),

          ]),

mainAxisSize chỉ định bao nhiêu không gian nằm ngang được chiếm giữ bởi Row, giá trị mặc định của nó là MainAxisSize.max điều này có nghĩa là Row sẽ cố gắng chiếm giữ không gian nằm ngang nhiều nhất có thể.

Nếu có một widget con với "flex > 0 && fit != FlexFit.loose" thì Row sẽ cố gắng chiếm giữ nhiều không gian nhiều nhất có thể mà không phụ thuộc vào giá trị của mainAxisSize.

Ngược lại, nếu mainAxisSize = MainAxisSize.min thì Row sẽ có một chiều rộng vừa đủ cho tất cả các widget con của nó. (Nếu object con trong Row được bao bởi Expanded thì MainAxisSize.min ko tác dụng)

return Container(

      width: 500,

      height: 120,

      color: const Color.fromARGB(255, 240, 195, 57),

      child: Row(mainAxisSize: MainAxisSize.min, children: [

        ElevatedButton(child: const Text("BTN 01"), onPressed: () {}),

        const Expanded( // do Expanded ở đây nên MainAxisSize.min ở trên mất tác dụng

          child: Icon(

            Icons.ac\_unit,

            size: 64,

            color: Colors.blue,

          ),

        ),

        ElevatedButton(

            onPressed: () {},

            style: ButtonStyle(

                minimumSize: MaterialStateProperty.all(const Size.square(50))),

            child: const Text("Button 2")),

        ElevatedButton(child: const Text("BTN 03"), onPressed: () {}),

      ]),

    );

verticalDirection: đảo ngược cách hiển thị của crossAxisAlignment: (verticalDirection có ý nghĩa hơn với Column – đảo ngược list các elements)

* Nếu crossAxisAlignment.start show object ở top thì verticalDirection.up đảo xuống bottom
* Nếu crossAxisAlignment.end show object ở bottom thì verticalDirection.up đảo lên top
* child: Row(
* crossAxisAlignment: CrossAxisAlignment.start,
* verticalDirection: VerticalDirection.up, // Đổi hướng sắp xếp dọc
* children: [
* ElevatedButton(child: const Text("BTN 01"), onPressed: () {}),
* const Icon(
* Icons.ac\_unit,
* size: 64,
* color: Colors.blue,
* ),
* ElevatedButton(
* onPressed: () {},
* style: ButtonStyle(
* minimumSize:
* MaterialStateProperty.all(const Size.square(50))),
* child: const Text("Button 2")),
* ElevatedButton(child: const Text("BTN 03"), onPressed: () {}),
* ]),

Trong Flutter, không có một phương thức tương tự verticalDirection của Column có thể đảo ngược hướng sắp xếp các widget con trong một Row một cách trực tiếp. Tuy nhiên, bạn có thể đảo ngược thứ tự của các widget con trong Row bằng cách sử dụng một số cách khác.

Một cách thông dụng là sử dụng hàm List.reversed để đảo ngược danh sách các widget con trước khi đưa chúng vào Row. Dưới đây là một ví dụ:

Widget build(BuildContext context) {

    List<Widget> children = [

      Container(

        color: Colors.red,

        width: 50,

        height: 50,

      ),

      Container(

        color: Colors.green,

        width: 50,

        height: 50,

      ),

      Container(

        color: Colors.blue,

        width: 50,

        height: 50,

      ),

    ];

    ElevatedButton(child: const Text("BTN 02"), onPressed: () {});

    return Row(

      mainAxisAlignment: MainAxisAlignment.center,

      children: children.reversed.toList(), // Đảo ngược danh sách widget con

    );

  }

## Column

[Column](https://api.flutter.dev/flutter/widgets/Column/Column.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [MainAxisAlignment](https://api.flutter.dev/flutter/rendering/MainAxisAlignment.html) mainAxisAlignment = MainAxisAlignment.start, [MainAxisSize](https://api.flutter.dev/flutter/rendering/MainAxisSize.html) mainAxisSize = MainAxisSize.max, [CrossAxisAlignment](https://api.flutter.dev/flutter/rendering/CrossAxisAlignment.html) crossAxisAlignment = CrossAxisAlignment.center, [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection, [VerticalDirection](https://api.flutter.dev/flutter/painting/VerticalDirection.html) verticalDirection = VerticalDirection.down, [TextBaseline](https://api.flutter.dev/flutter/dart-ui/TextBaseline.html)? textBaseline, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[]})

Creates a vertical array of children.

*Const*

Một số giá trị setup default quan trọng:

* mainAxisAlignment: gióng hàng chiều dọc start
* mainAxisSize: chiều dọc full size container
* crossAxisAlignment: gióng hàng chiều ngang center(các elements gióng center trong column, vị trí column phải được set trong container cha nó).  
  Nếu set CrossAxisAlignment.stretch thì size chiều ngang full size container, nhưng các elements lại gióng tại vị trí lề right. Lúc này để canh ngang elements center thì bao elements với setup: Container(alignment: Alignment.center,

Example alignment

//---------------------------

// Column1 - alignments hàng loạt

class Column1 extends StatelessWidget {

  Column1({super.key});

  final List<Quote> quotes = [

    Quote(

      text: 'Class Quotes1 can be separated into another class',

      author: 'NMSon',

    ),

    Quote(

      text: 'Quotes1: And import at the beginning of this file',

      author: 'NXTruong',

    ),

    Quote(

      text: 'Quotes1: Custom classes can be - ',

      author: 'NTLuong',

    ),

  ];

  @override

  Widget build(BuildContext context) {

    return Container(

      alignment: Alignment.center,

      color: Colors.amber,

      child: Column(

        mainAxisSize: MainAxisSize.min,

        mainAxisAlignment: MainAxisAlignment.start,

        crossAxisAlignment: CrossAxisAlignment.stretch,

        // Khi crossAxisAlignment: CrossAxisAlignment.stretch, để alignment crossAxisAlignment vô center thì phải dùng Container với alignment: Alignment.center.

        children: quotes

            .map((quote) => Container(

                  alignment:

                      Alignment.center, // Align text within the container

                  child: Text(

                    quote.text,

                    style: const TextStyle(

                        fontSize: 16), // Adjust the text size as needed

                  ),

                ))

            .toList(),

      ),

    );

  }

}

// Column2 - alignment từng element theo conditions of each element

class Column2 extends StatelessWidget {

  Column2({super.key});

  final List<Quote> quotes = [

    Quote(

      text: 'Class Quotes1 can be separated into another class',

      author: 'NMSon',

    ),

    Quote(

      text: 'Quotes1: And import at the beginning of this file',

      author: 'NXTruong',

      alignment: 'left',

    ),

    Quote(

      text: 'Quotes1: Custom classes can be - ',

      author: 'NTLuong',

      alignment: 'right',

    ),

  ];

  @override

  Widget build(BuildContext context) {

    return Container(

      alignment: Alignment.center,

      color: Colors.amber,

      child: Column(

        mainAxisSize: MainAxisSize.min,

        mainAxisAlignment: MainAxisAlignment.start,

        crossAxisAlignment: CrossAxisAlignment.stretch,

        // Khi crossAxisAlignment: CrossAxisAlignment.stretch, để alignment crossAxisAlignment vô center thì phải dùng Container với alignment: Alignment.center.

        // Using crossAxisAlignment: CrossAxisAlignment.stretch combined textAlign of Text widget with conditions to align each line

        children: quotes

            .map((quote) => Text(

                  textAlign: quote.alignment == 'right'

                      ? TextAlign.right

                      : quote.alignment == 'left'

                          ? TextAlign.left

                          : TextAlign.center,

                  quote.text,

                  style: const TextStyle(

                      fontSize: 14), // Adjust the text size as needed

                ))

            .toList(),

      ),

    );

  }

}

//

class Quote {

  late String text;

  late String author;

  late String? alignment;

  // Khai báo object with parameters named - phải gọi parameters named khi khai báo object

  Quote({

    required this.text,

    required this.author,

    this.alignment,

  });

}

Column là một widget hiển thị các widget con của nó trên một cột. Một biến thể khác là Row, hiển thị các widget con của nó trên một hàng.

Để làm cho một widget con của Column có thể mở rộng lấp đầy khoảng không gian thẳng đứng sẵn có bạn có thể gói nó bên trong một đối tượng Expanded.

Column đặt các con của nó trên một cột và không thể cuộn, nếu bạn muốn có một bộ chứa (container) tương tự và có thể cuộn được hãy cân nhắc sử dụng ListView.

Các properties trong Column cũng tương đương trong Row, hoán đổi trục ngang - dọc so với Row.

Đảo ngược list object con trong Column:

Widget build(BuildContext context) {

    return Container(

      width: 150,

      height: 500,

      color: const Color.fromARGB(255, 240, 195, 57),

      child: Column(

          mainAxisAlignment: MainAxisAlignment.spaceEvenly,

          crossAxisAlignment: CrossAxisAlignment.start,

          verticalDirection: VerticalDirection.up, // default .down

          mainAxisSize: MainAxisSize.min,

          children: [

            ElevatedButton(child: const Text("BTN 0"), onPressed: () {}),

            ElevatedButton(child: const Text("BTN 2"), onPressed: () {}),

            ElevatedButton(child: const Text("BTN 3"), onPressed: () {}),

            ElevatedButton(child: const Text("BTN 4"), onPressed: () {}),

            ElevatedButton(child: const Text("BTN 5"), onPressed: () {}),

          ]),

    );

  }

}

## Table

Table(

{[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key,

[List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[TableRow](https://api.flutter.dev/flutter/widgets/TableRow-class.html)> children = const <TableRow>[],

[Map](https://api.flutter.dev/flutter/dart-core/Map-class.html)<[int](https://api.flutter.dev/flutter/dart-core/int-class.html), [TableColumnWidth](https://api.flutter.dev/flutter/rendering/TableColumnWidth-class.html)>? columnWidths,

[TableColumnWidth](https://api.flutter.dev/flutter/rendering/TableColumnWidth-class.html) defaultColumnWidth = const FlexColumnWidth(),

[TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection,

[TableBorder](https://api.flutter.dev/flutter/rendering/TableBorder-class.html)? border,

[TableCellVerticalAlignment](https://api.flutter.dev/flutter/rendering/TableCellVerticalAlignment.html) defaultVerticalAlignment = TableCellVerticalAlignment.top,

[TextBaseline](https://api.flutter.dev/flutter/dart-ui/TextBaseline.html)? textBaseline}

)

Creates a table.

The [children](https://api.flutter.dev/flutter/widgets/Table/children.html), [defaultColumnWidth](https://api.flutter.dev/flutter/widgets/Table/defaultColumnWidth.html), and [defaultVerticalAlignment](https://api.flutter.dev/flutter/widgets/Table/defaultVerticalAlignment.html) arguments must not be null.

Table widget trong Flutter được sử dụng khi bạn muốn sắp xếp các phần tử hoặc widget trong một bảng có cấu trúc cố định. Table widget cho phép bạn tạo một bảng có nhiều dòng và cột, và bạn có thể kiểm soát cách chúng sắp xếp.

class Home0 extends StatelessWidget {

  const Home0({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: const Text('Table Widget Demo'),

        ),

        body: Table(

          defaultColumnWidth:

              const FlexColumnWidth(), //  defaultColumnWidth - not null

          border: TableBorder.all(),

          children: [

            TableRow(

              children: <Widget>[

                Container(

                  color: Colors.blue,

                  height: 100,

                ),

                Container(

                  color: Colors.red,

                  height: 100,

                ),

                Container(

                  color: Colors.green,

                  height: 100,

                ),

              ],

            ),

            TableRow(

              children: [

                Container(

                  color: Colors.orange,

                  height: 100,

                ),

                Container(

                  color: Colors.purple,

                  height: 100,

                ),

                Container(

                  color: Colors.yellow,

                  height: 100,

                ),

              ],

            ),

          ],

        ));

  }

}

// Home1 -

class Home1 extends StatelessWidget {

  const Home1({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(title: const Text('Table Sample')),

        body: Table(

          border: TableBorder.all(),

          columnWidths: const <int, TableColumnWidth>{

            ///

// Dưới đây là giải thích ý nghĩa của đoạn mã:

// columnWidths là một ánh xạ (map) từ chỉ mục cột (số nguyên) đến độ rộng mong muốn của cột đó. Chúng ta đang xác định độ rộng cho mỗi cột dựa trên chỉ mục của cột.

// Cột có chỉ mục 0 sử dụng IntrinsicColumnWidth(). Điều này có nghĩa là độ rộng của cột này sẽ được tự động điều chỉnh để phù hợp với nội dung bên trong cột. Nếu nội dung của cột đủ rộng, cột sẽ có độ rộng tối ưu, không quá lớn hoặc nhỏ. Điều này thường được sử dụng cho các cột chứa văn bản hoặc nội dung động.

// Cột có chỉ mục 1 sử dụng FlexColumnWidth(). Điều này có nghĩa rằng cột này sẽ sử dụng một mô hình phần trăm (flex) để xác định độ rộng. Cụ thể, nó sẽ sử dụng một phần trăm tương đối so với các cột khác trong bảng. Điều này cho phép bạn điều chỉnh tỷ lệ độ rộng giữa các cột.

// Cột có chỉ mục 2 sử dụng FixedColumnWidth(64). Điều này đặt độ rộng của cột này là một giá trị cố định là 64 đơn vị. Cột này sẽ có kích thước cố định không thay đổi dựa trên nội dung hay tỷ lệ.

            0: IntrinsicColumnWidth(), // auto resize column width

            1: FlexColumnWidth(),

            2: FixedColumnWidth(64),

          },

          defaultVerticalAlignment:

              TableCellVerticalAlignment.middle, // not null

          children: <TableRow>[

            // children - not null. Every row in a table must have the same number of children (số column bằng nhau), and all the children must be non-null.

            TableRow(

              children: <Widget>[

                Container(

                  height: 32,

                  color: Colors.green,

                ),

                TableCell(

                  verticalAlignment: TableCellVerticalAlignment.top,

                  child: Container(

                    height: 32,

                    width: 32,

                    color: Colors.red,

                  ),

                ),

                Container(

                  height: 64,

                  color: Colors.blue,

                ),

              ],

            ),

            TableRow(

              decoration: const BoxDecoration(

                color: Colors.grey,

              ),

              children: <Widget>[

                Container(

                  height: 64,

                  width: 128,

                  color: Colors.purple,

                ),

                Container(

                  height: 32,

                  color: Colors.yellow,

                ),

                Center(

                  child: Container(

                    height: 32,

                    width: 32,

                    color: Colors.orange,

                  ),

                ),

              ],

            ),

          ],

        ));

  }

}

## SingleChildScrollView

[SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/SingleChildScrollView.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual})

Creates a box in which a single widget can be scrolled.

SingleChildScrollView là một widget trong Flutter được sử dụng để tạo một phần của giao diện có khả năng cuộn (scrollable). Nó cho phép bạn đặt nội dung bên trong nó và khi nội dung đó vượt ra ngoài kích thước màn hình, người dùng có thể cuộn qua nó để xem nội dung còn lại.

SingleChildScrollView có thể ảnh hưởng đến hiệu suất nếu bạn đặt quá nhiều widgets con hoặc nội dung quá lớn bên trong nó. Điều này có thể dẫn đến tốn tài nguyên và hiệu suất yếu.

Example 1

 Widget build(BuildContext context) {

    return SingleChildScrollView(

      child: Column(

        children: <Widget>[

          Container(

            height: 200.0,

            color: Colors.blue,

            child: const Center(

              child: Text(

                'Phần đầu trang',

                style: TextStyle(fontSize: 24.0, color: Colors.white),

              ),

            ),

          ),

      Container(

... ... .. ..

Example 2

Widget build(BuildContext context) {

    return Center(

      child: SingleChildScrollView(

        scrollDirection: Axis.horizontal,

        child: Row(

          children: <Widget>[

            Container(

              width: 200.0,

              height: 200.0,

              color: Colors.red,

            ),

            Container(....

Example 3

 Widget build(BuildContext context) {

    return const SingleChildScrollView(

      child: ListBody(

        children: <Widget>[

          ListTile(

            title: Text('Item 1'),

            leading: Text('Leading: data '),

            subtitle: Icon(Icons.ac\_unit),

          ),

          ListTile( . . . .

Example 4

// lồng SingleChildScrollView trong SingleChildScrollView

class SingleChildScrollView3 extends StatelessWidget {

  const SingleChildScrollView3({super.key});

  @override

  Widget build(BuildContext context) {

    return SingleChildScrollView(

      child: Column(

        children: <Widget>[

          Padding(

            padding: const EdgeInsets.all(8.0),

            child: SingleChildScrollView(

              scrollDirection: Axis.horizontal,

              child: Row(

                children: [

                  Container(

                    color: Colors.red,

                    height: 150,

                    width: 250,

                  ),

                  Container(

                    color: const Color.fromARGB(255, 211, 239, 116),

                    height: 150,

                    width: 250,

                  ),

                  Container(

                    color: Colors.green,

                    height: 150,

                    width: 250,

                  )

                ],

              ),

            ),

          ),

          Container(

            height: 200,

            color: const Color.fromARGB(255, 254, 83, 217),

          ),

          Container(

            height: 200,

            color: Colors.blueAccent,

          ),

          Container(

            height: 200,

            color: const Color.fromARGB(255, 63, 247, 216),

          ),

          Container(

            height: 200,

            color: const Color.fromARGB(255, 243, 245, 150),

          ),

        ],

      ),

    );

  }

}

## ListView

[ListView](https://api.flutter.dev/flutter/widgets/ListView/ListView.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? itemExtent, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? prototypeItem, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[], [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, linear array of widgets from an explicit [List](https://api.flutter.dev/flutter/dart-core/List-class.html).

[ListView.builder](https://api.flutter.dev/flutter/widgets/ListView/ListView.builder.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? itemExtent, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? prototypeItem, required [NullableIndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/NullableIndexedWidgetBuilder.html) itemBuilder, [ChildIndexGetter](https://api.flutter.dev/flutter/widgets/ChildIndexGetter.html)? findChildIndexCallback, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? itemCount, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, linear array of widgets that are created on demand.

[ListView.custom](https://api.flutter.dev/flutter/widgets/ListView/ListView.custom.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? itemExtent, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? prototypeItem, required [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html) childrenDelegate, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, linear array of widgets with a custom child model.

*const*

[ListView.separated](https://api.flutter.dev/flutter/widgets/ListView/ListView.separated.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, required [NullableIndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/NullableIndexedWidgetBuilder.html) itemBuilder, [ChildIndexGetter](https://api.flutter.dev/flutter/widgets/ChildIndexGetter.html)? findChildIndexCallback, required [IndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/IndexedWidgetBuilder.html) separatorBuilder, required [int](https://api.flutter.dev/flutter/dart-core/int-class.html) itemCount, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a fixed-length scrollable linear array of list "items" separated by list item "separators"

A specialized Column for organizing a list of boxes

Can be laid out horizontally or vertically

Detects when its content won’t fit and provides scrolling

Less configurable than Column, but easier to use and supports scrolling

Một số thuộc tính

* itemBuilder: (context, index)…

Hàm callback tạo items cho list. Index khởi tạo bằng 0 và tự động tăng sau mỗi lần gọi callback

* itemExtent

Thuộc tính itemExtent trong ListView của Flutter được sử dụng để đặt chiều cao cố định cho từng mục (item) trong danh sách. Nó xác định chiều cao mà tất cả các mục sẽ có và giúp cải thiện hiệu suất cuộn của ListView, đặc biệt khi bạn biết trước rằng các mục trong danh sách của bạn có cùng kích thước.

Khi bạn đặt itemExtent, ListView sẽ tối ưu hóa cuộn bằng cách tính toán trước số lượng mục có thể hiển thị trên màn hình mà không cần phải thực tế xây dựng chúng, làm giảm tải và tăng hiệu suất.

Ví dụ, nếu bạn có một danh sách các mục có cùng kích thước và bạn biết rằng chiều cao của mỗi mục là 50 pixels, bạn có thể đặt itemExtent thành 50 để cải thiện hiệu suất cuộn của ListView.

Tuy nhiên, lưu ý rằng khi bạn sử dụng itemExtent, bạn sẽ mất đi tính linh hoạt trong việc xử lý các mục có kích thước khác nhau. Do đó, bạn nên sử dụng itemExtent chỉ khi bạn thực sự biết rằng tất cả các mục trong danh sách của bạn có kích thước giống nhau và muốn tối ưu hóa hiệu suất.

* addAutomaticKeepAlives: Thuộc tính addAutomaticKeepAlives trong một ListView trong Flutter quyết định liệu các phần tử con của danh sách có được giữ cho hiển thị trạng thái giữa các lần xem lại hay không. Khi addAutomaticKeepAlives được đặt thành true (giá trị mặc định), các phần tử con của ListView sẽ cố gắng duy trì trạng thái của họ khi bạn cuộn qua danh sách và trở lại. Điều này đặc biệt hữu ích khi bạn có các phần tử con có trạng thái giữa các lần cuộn và không muốn mất trạng thái đó.

Ví dụ, nếu bạn có một danh sách các hình ảnh và người dùng đã thực hiện tương tác với một số hình ảnh (ví dụ: đã chọn yêu thích một hình ảnh), nếu addAutomaticKeepAlives được đặt thành true, các tương tác này sẽ được duy trì khi bạn cuộn qua danh sách và quay lại chúng, giúp người dùng thấy được trạng thái đã chọn của hình ảnh.

Khi addAutomaticKeepAlives được đặt thành false, các phần tử con sẽ không được duy trì trạng thái giữa các lần cuộn và quay lại. Điều này có thể giúp tối ưu hóa hiệu suất nếu bạn không cần giữ trạng thái của các phần tử con trong ListView.

* physics: được sử dụng để xác định cách cuộn (scrolling) hoạt động trong danh sách. Nó thường được sử dụng để điều chỉnh cách danh sách phản ứng với cuộn và hành vi cuộn nào sẽ được áp dụng.

Dưới đây là một số giá trị phổ biến mà bạn có thể sử dụng cho thuộc tính physics:

* + AlwaysScrollableScrollPhysics: Đây là giá trị mặc định. Nó luôn cho phép cuộn, dù danh sách có thể rất nhỏ.
  + BouncingScrollPhysics: Đây là giá trị mặc định cho ListView trên các thiết bị iOS. Nó tạo ra hiệu ứng giật khi bạn đạt đến đầu hoặc cuối danh sách. Phù hợp cho danh sách có thể cuộn.
  + ClampingScrollPhysics: Đây là giá trị mặc định cho ListView trên các thiết bị Android. Nó giữ danh sách ở đầu hoặc cuối khi bạn cố gắng cuộn quá giới hạn.
  + NeverScrollableScrollPhysics: Không cho phép cuộn trong danh sách. Điều này thường được sử dụng khi bạn muốn tắt tính năng cuộn.

ListView() default – thường dùng với số items nhỏ (items lớn chiếm nhiều bộ nhớ, view giật lag)

* prototypeItem: (cũng áp dụng trong ListView.builder)

If non-null, the prototypeItem forces the children to have the same extent as the given widget in the scroll direction.

You can't specify both itemExtent and prototypeItem, only one or none of them.

Thuộc tính prototypeItem trong ListView của Flutter được sử dụng để định rõ kích thước và nội dung của tất cả các mục (items) trong danh sách. Khi bạn sử dụng prototypeItem, tất cả các mục sẽ có cùng kích thước và nội dung giống như prototypeItem.

Điều này có thể hữu ích khi bạn muốn đảm bảo rằng tất cả các mục trong danh sách có kích thước và nội dung giống nhau, đặc biệt khi dữ liệu trong danh sách có sự biến đổi, nhưng bạn muốn các mục vẫn có cùng kích thước để cải thiện hiệu suất cuộn.

Ví dụ, nếu bạn có một danh sách của các widg

et có kích thước biến đổi (ví dụ: các hình ảnh có kích thước khác nhau), nhưng bạn muốn chúng hiển thị dưới dạng các mục có cùng kích thước như prototypeItem, bạn có thể sử dụng prototypeItem để định rõ kích thước và nội dung mẫu mà tất cả các mục sẽ tuân theo.

Màu sắc, text… trong prototypeItem chỉ là tượng trưng, mang tính chỉ dẫn, 2 thuộc tính width và height là giá trị max – ko cho phép các items vượt hơn.

Example 1

 Widget build(BuildContext context) {

    return ListView(

      padding: const EdgeInsets.all(8),

      children: <Widget>[

        Container(

          height: 50,

          color: Colors.amber[600],

          child: const Center(child: Text('Entry A')),

        ),

        Container(

          height: 50,

          color: Colors.amber[500],

          child: const Center(child: Text('Entry B')),

        ),

        Container(

          height: 50,

          color: Colors.amber[100],

          child: const Center(child: Text('Entry C')),

        ),

      ],

    );

  }

Example 2

 Widget build(BuildContext context) {

    return ListView(

      // Chỉ có thể set 1 trong 2 thuộc tính itemExtent: or prototypeItem:

      // itemExtent: 200.0, // Đặt chiều cao cho tất cả các mục

      prototypeItem: Container(

        // Mục mẫu là một Container

        color: Colors.grey[300],

        child: Center(

          child: Image.asset(

            'assets/bear.jpg', // Đường dẫn tới hình ảnh mẫu

            height: 120.0, // Chiều cao của hình ảnh

            width: 120.0, // Chiều rộng của hình ảnh

          ),

        ),

      ),

      children: [

        // Các mục thực tế trong danh sách

        Container(

          color: const Color.fromARGB(255, 243, 28, 28),

          child: Center(

            child: Image.asset(

              'assets/bear.jpg', // Đường dẫn tới hình ảnh thứ nhất

              height: 120.0,

              width: 120.0,

            ),

          ),

        ),

        Container(

          color: Colors.grey[300],

          child: Center(

            child: Image.asset(

              'assets/lion.jpg', // Đường dẫn tới hình ảnh thứ hai

              height: 140.0,

              width: 120.0,

            ),

          ),

        ),

        // Thêm các mục hình ảnh khác tại đây

      ],

    );

  }

Example 3

Widget build(BuildContext context) {

    return Container(

      color: const Color.fromARGB(255, 240, 234, 215),

      child: ListView(

        // itemExtent: set fix height cho items

        itemExtent: 50,

        // Thuộc tính padding để thêm khoảng cách xung quanh danh sách

        padding: const EdgeInsets.all(15.0),

        // Đảo list

        reverse: true,

        // Thuộc tính scrollDirection để thiết lập hướng cuộn (mặc định là vertical)

        scrollDirection: Axis.vertical,

        // Thuộc tính shrinkWrap để đóng gói danh sách theo chiều dọc

        shrinkWrap: true,

        // Thuộc tính physics để điều khiển cách cuộn

        physics: const BouncingScrollPhysics(),

        // lưu giữ trạng thái các item khi cuộn qua và quay về lại

        addAutomaticKeepAlives: true,

        children: items.map((item) {

          return Container(

            color: const Color.fromARGB(255, 184, 192, 240),

            child: Column(

              children: [

                ListTile(

                  title: Text('Item $item'),

                ),

              ],

            ),

          );

        }).toList(),

      ),

    );

  }

ListView.builder

* itemCount: Số item được load lên ListView, phải <= số elements trong list. Ko set sẽ báo lỗi tràn index khi cuộn màn hình

Nếu bạn không đặt giá trị cho thuộc tính itemCount trong ListView.builder, mặc định giá trị của nó sẽ là null. Khi giá trị của itemCount là null, ListView.builder sẽ tạo ra một danh sách vô hạn, nghĩa là index sẽ không bao giờ dừng lại và tăng lên một cách vô hạn khi bạn cuộn danh sách.

Việc này có thể gây ra vấn đề về hiệu suất và gây ra vòng lặp vô hạn nếu bạn không kiểm soát cẩn thận. Do đó, thường thì bạn nên đặt giá trị cho itemCount để xác định số lượng mục trong danh sách hoặc sử dụng nó với một giá trị cố định để đảm bảo rằng chỉ mục không tăng vô hạn.

* itemBuilder (cũng áp dụng trong ListView.separated)

itemBuilder là một hàm được gọi mỗi khi một mục trong danh sách cần được hiển thị. Hàm này nhận vào hai đối số: BuildContext và int index, trong đó BuildContext là ngữ cảnh xây dựng của mục và index là chỉ mục của mục trong danh sách. itemBuilder sẽ trả về một widget hoặc một phần tử con để hiển thị cho mục tương ứng với chỉ mục đó.

Example 1

class ListViewBuider1 extends StatelessWidget {

  ListViewBuider1({super.key});

  final List<String> entries =

      List.generate(50, (index) => 'Item ${index + 1}');

  @override

  Widget build(BuildContext context) {

    return ListView.builder(

        itemExtent: 50,

        padding: const EdgeInsets.all(8),

        itemCount: entries.length,

        itemBuilder: (BuildContext context, int index) {

          return SizedBox(

            height: 50,

            child: Center(child: Text('Entry ${entries[index]}')),

          );

        });

  }

}

Example 2

class ListViewBuilder2 extends StatelessWidget {

  // Tạo danh sách 30 mã màu

  final List<String> colorCodes = List.generate(30, (index) {

    // Tạo mã màu ngẫu nhiên (ví dụ: "#RRGGBB")

    final r = (index \* 10 % 256).toRadixString(16).padLeft(2, '0');

    final g = (index \* 5 % 256).toRadixString(16).padLeft(2, '0');

    final b = (index \* 20 % 256).toRadixString(16).padLeft(2, '0');

    return "0xFF$r$g$b";

  });

  ListViewBuilder2({super.key});

  @override

  Widget build(BuildContext context) {

    return ListView.builder(

      // prototypeItem: ,

      itemCount: colorCodes.length,

      itemBuilder: (context, index) {

        final colorCode = colorCodes[index];

        return Container(

          color: Color(int.parse(colorCode.replaceAll("#", "0xFF"))),

          height: 100, // Đặt chiều cao cho từng mục

          child: Center(

            child: Text(

              colorCode,

              style: const TextStyle(color: Colors.white),

            ),

          ),

        );

      },

    );

  }

}

ListView.separated – tạo list item xen kẻ với list separated

* separatorBuilder: là một hàm được gọi để xây dựng phần tách biệt giữa hai mục liền kề itemBuilder, ListView load lên cho ta separatorBuilder xen kẻ với itemBuilder
* class ListViewSeparated extends StatelessWidget {
* // Tạo danh sách 30 mã màu
* final List<String> colorCodes = List.generate(30, (index) {
* // Tạo mã màu ngẫu nhiên (ví dụ: "#RRGGBB")
* final r = (index \* 5 % 256).toRadixString(16).padLeft(2, '0');
* final g = (index \* 20 % 256).toRadixString(16).padLeft(2, '0');
* final b = (index \* 30 % 256).toRadixString(16).padLeft(2, '0');
* return "#$r$g$b";
* });
* ListViewSeparated({super.key});
* @override
* Widget build(BuildContext context) {
* return ListView.separated(
* itemCount: colorCodes.length,
* // separatorBuilder: list xen kẻ list itemBuilder:
* separatorBuilder: (BuildContext context, index) => const Divider(),
* itemBuilder: (context, index) {
* final colorCode = colorCodes[index];
* return Container(
* color: Color(int.parse(colorCode.replaceAll("#", "0xFF"))),
* height: 100, // Đặt chiều cao cho từng mục
* child: Center(
* child: Text(
* colorCode,
* style: const TextStyle(color: Color.fromARGB(255, 255, 255, 255)),
* ),
* ),
* );
* },
* );
* }
* }

ListView.custom

* constructor takes a property childrenDelegate kiểu SliverChildDelegate, which provides the ability to customize additional aspects of the child model. For example, a SliverChildDelegate can control the algorithm used to estimate the size of children that are not actually visible.
* The main parameter of this constructor is a childrenDelegate which builds the items.

The types of SliverChildDelegates are :

* + SliverChildListDelegate
  + SliverChildBuilderDelegate

The SliverChildListDelegate accepts a list of children widgets. whereas the SliverChildBuilderDelegate accepts an IndexedWidgetBuilder, simply a builder() function. Digging deeper, we can infer that ListView.builder was created using a ListView.custom with a SliverChildBuilderDelegate. Also, the default ListView() constructor is a ListView.custom with a SliverChildListDelegate.

Tóm lại, SliverChildListDelegate phù hợp khi bạn có một danh sách cố định của các widget con, trong khi SliverChildBuilderDelegate phù hợp khi bạn cần xây dựng danh sách các widget con dựa trên dữ liệu hoặc logic động.

SliverChildListDelegate

Widget build(BuildContext context) {

    return ListView.custom(

      childrenDelegate: SliverChildListDelegate(

        [

          const Text('Item 1'),

          const Text('Item 2'),

          const Text('Item 3'),

        ],

      ),

    );

  }

SliverChildBuilderDelegate

Example 1

Widget build(BuildContext context) {

    return ListView.custom(

      childrenDelegate: SliverChildBuilderDelegate(

        // index auto increment từ 0 đến childCount - 1. Nếu ko set childCount thì load vô hạn

        (BuildContext context, int index) {

          return ListTile(

            title: Text('Item: $index'),

          );

        },

        childCount: 20,

      ),

    );

  }

Example 2

class ListViewCustom2 extends StatelessWidget {

  final List<String> items =

      List<String>.generate(50, (index) => 'Item $index');

  ListViewCustom2({super.key});

  @override

  Widget build(BuildContext context) {

    return ListView.custom(

      childrenDelegate: SliverChildBuilderDelegate(

        (BuildContext context, int index) {

          return ListTile(

            title: Text(items[index]),

          );

        },

        childCount: items.length,

      ),

    );

  }

}

## GridView

[GridView](https://api.flutter.dev/flutter/widgets/GridView/GridView.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, required [SliverGridDelegate](https://api.flutter.dev/flutter/rendering/SliverGridDelegate-class.html) gridDelegate, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[], [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId})

Creates a scrollable, 2D array of widgets with a custom [SliverGridDelegate](https://api.flutter.dev/flutter/rendering/SliverGridDelegate-class.html).

[GridView.builder](https://api.flutter.dev/flutter/widgets/GridView/GridView.builder.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, required [SliverGridDelegate](https://api.flutter.dev/flutter/rendering/SliverGridDelegate-class.html) gridDelegate, required [NullableIndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/NullableIndexedWidgetBuilder.html) itemBuilder, [ChildIndexGetter](https://api.flutter.dev/flutter/widgets/ChildIndexGetter.html)? findChildIndexCallback, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? itemCount, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, 2D array of widgets that are created on demand.

[GridView.count](https://api.flutter.dev/flutter/widgets/GridView/GridView.count.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, required [int](https://api.flutter.dev/flutter/dart-core/int-class.html) crossAxisCount, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) mainAxisSpacing = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) crossAxisSpacing = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) childAspectRatio = 1.0, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[], [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, 2D array of widgets with a fixed number of tiles in the cross axis.

[GridView.custom](https://api.flutter.dev/flutter/widgets/GridView/GridView.custom.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, required [SliverGridDelegate](https://api.flutter.dev/flutter/rendering/SliverGridDelegate-class.html) gridDelegate, required [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html) childrenDelegate, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, 2D array of widgets with both a custom [SliverGridDelegate](https://api.flutter.dev/flutter/rendering/SliverGridDelegate-class.html) and a custom [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html).

*const*

[GridView.extent](https://api.flutter.dev/flutter/widgets/GridView/GridView.extent.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, required [double](https://api.flutter.dev/flutter/dart-core/double-class.html) maxCrossAxisExtent, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) mainAxisSpacing = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) crossAxisSpacing = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) childAspectRatio = 1.0, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[], [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, 2D array of widgets with tiles that each have a maximum cross-axis extent.

Detects when the column content exceeds the render box and automatically provides scrolling

### GridView mặc định

Sử dụng khi bạn có một danh sách cố định của các phần tử (item) trong lưới và chúng không thay đổi.

Cung cấp danh sách các phần tử trực tiếp trong thuộc tính children.

Thích hợp khi bạn có một số phần tử cố định và muốn hiển thị chúng mà không cần tạo chúng động hoặc nâng cao hiệu suất.

Widget build(BuildContext context) {

    return GridView(

      gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(

        crossAxisCount: 3, // Required - Số cột trong grid

        crossAxisSpacing: 5.0, // Khoảng cách giữa các cột

        mainAxisSpacing: 5.0, // Khoảng cách giữa các dòng

        childAspectRatio:

            0.8, // Tỷ lệ khung hình của mỗi phần tử - tỉ lệ nghịch với kích thước item (theo crossAxis)

      ),

      children: <Widget>[

        // Các phần tử trong grid

        Container(

          color: Colors.red,

          child: const Center(

            child: Text('Item 1'),

          ),

        ),

        Container(

          color: Colors.blue,

          child: const Center(

            child: Text('Item 2'),

          ),

        ),

        Container(

          color: Colors.green,

          child: const Center(

            child: Text('Item 3'),

          ),

        ),

        Container(

          color: Colors.orange,

          child: const Center(

            child: Text('Item 4'),

          ),

        ),

      ],

    );

  }

Example 2

 Widget build(BuildContext context) {

    return GridView(

      gridDelegate: const SliverGridDelegateWithMaxCrossAxisExtent(

        maxCrossAxisExtent: 150.0, // required -  chiều rộng tối đa cho mỗi item

        mainAxisSpacing: 5.0, // Khoảng cách giữa các dòng

        crossAxisSpacing: 5.0, // khoảng cách giữa các column

        // childAspectRatio: 2.0, // Tỷ lệ khung hình của mỗi phần tử - tỉ lệ nghịch với kích thước item (theo crossAxis)

      ),

      children: <Widget>[

        // Các phần tử trong grid

        Container(

          color: Colors.red,

          child: const Center(

            child: Text('Item 1'),

          ),

        ),

        Container(

          color: Colors.blue,

          child: const Center(

            child: Text('Item 2'),

          ),

        ),

        Container(

          color: Colors.green,

          child: const Center(

            child: Text('Item 3'),

          ),

        ),

        Container(

          color: Colors.orange,

          child: const Center(

            child: Text('Item 4'),

          ),

        ),

      ],

    );

  }

### GridView.builder

Sử dụng khi bạn muốn xây dựng lưới dựa trên dữ liệu động hoặc lớn.

Sử dụng itemBuilder để tạo từng phần tử (item) của lưới dựa trên dữ liệu.

Hiệu suất tốt hơn khi bạn có một lượng lớn dữ liệu hoặc dữ liệu động vì nó chỉ xây dựng các phần tử cần thiết và tái sử dụng chúng khi cuộn.

Example 1

final List<String> items = List<String>.generate(50, (index) => 'Item $index');

class GridViewBuilder1 extends StatelessWidget {

  const GridViewBuilder1({super.key});

  @override

  Widget build(BuildContext context) {

    return GridView.builder(

      gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(

        crossAxisCount: 3, // Số cột

      ),

      itemBuilder: (BuildContext context, int index) {

        return Container(

          decoration: BoxDecoration(

            border: Border.all(

              color: Colors.red, // Màu của viền

              width: 2.0, // Độ dày của viền

            ),

          ),

          child: ListTile(

            title: Text(items[index]),

          ),

        );

      },

      itemCount: items.length,

    );

  }

}

Example 2

class ItemTile extends StatelessWidget {

  final int itemNo;

  const ItemTile(this.itemNo, {super.key});

  @override

  Widget build(BuildContext context) {

    // itemNo % Colors.primaries.length sẽ là một số nguyên trong khoảng từ 0 đến Colors.primaries.length - 1, và nó được sử dụng để chọn một màu cụ thể từ danh sách Colors.primaries.

    final Color color = Colors.primaries[itemNo % Colors.primaries.length];

    return Padding(

      padding: const EdgeInsets.all(8.0),

      child: ListTile(

        tileColor: color.withOpacity(0.2),

        onTap: () {},

        leading: Container(

          width: 50,

          height: 30,

          color: color.withOpacity(0.5),

          child: Placeholder(

            color: color,

          ),

        ),

        title: Text(

          'Product $itemNo',

          key: Key('text\_$itemNo'),

        ),

      ),

    );

  }

}

class GridViewBuilder2 extends StatelessWidget {

  const GridViewBuilder2({super.key});

  @override

  Widget build(BuildContext context) {

    return GridView.builder(

      itemCount: 100,

      itemBuilder: (context, index) => ItemTile(index), // call function ItemTile

      gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(

        crossAxisCount: 2,

        childAspectRatio: 2,

      ),

    );

  }

}

Example 3

class MyGrid extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: MyGridPage(),

    );

  }

}

class MyGridPage extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Grid Overlay Example'),

      ),

      body: Stack(

        children: [

          // Your main content goes here

          Center(

            child: Text('Main Content'),

          ),

          // Overlay grid

          GridView.builder(

            gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(

              crossAxisCount: 10, // Số cột trong lưới

            ),

            itemBuilder: (context, index) {

              return Container(

                decoration: BoxDecoration(

                  border: Border.all(color: Colors.grey.shade300),

                ),

              );

            },

          ),

        ],

      ),

    );

  }

### GridView.count

Allows you to specify the number of columns

Example 1

 Widget build(BuildContext context) {

    return GridView.count(

      crossAxisCount: 2, // Requỉed - Số lượng cột

      children: const <Widget>[

        // Danh sách các phần tử

        Text('Item 1'),

        Text('Item 2'),

        Text('Item 3'),

        Text('Item 4'),

        Text('Item 5'),

        Text('Item 6'),

        // ...

      ],

    );

  }

Example 2

class DialKey extends StatelessWidget {

  final String? number;

  final String? letters;

  const DialKey({super.key, this.number, this.letters});

  @override

  Widget build(BuildContext context) {

    return Center(

      child: SizedBox(

        width: 80,

        height: 80,

        child: FloatingActionButton(

          onPressed: () {},

          backgroundColor: Colors.grey.withOpacity(0.7),

          child: Column(

            mainAxisAlignment: MainAxisAlignment.center,

            children: [

              Text(

                '$number',

                style: const TextStyle(

                    color: Colors.white,

                    fontSize: 24,

                    fontWeight: FontWeight.bold),

              ),

              Text(

                '$letters',

                style: const TextStyle(

                    color: Colors.white,

                    fontSize: 16,

                    fontWeight: FontWeight.bold),

              )

            ],

          ),

        ),

      ),

    );

  }

}

class GridViewCount2 extends StatelessWidget {

  const GridViewCount2({super.key});

  @override

  Widget build(BuildContext context) {

    return GridView.count(

      crossAxisCount: 3,

      shrinkWrap: true,

      padding: const EdgeInsets.only(left: 24, right: 24),

      children: const [

        DialKey( // call function DialKey

          number: '1',

          letters: '',

        ),

        DialKey(

          number: '2',

          letters: 'ABC',

        ),

        DialKey(

          number: '3',

          letters: 'DEF',

        ),

        // DialKey(...)

      ],

    );

  }

}

Example 3

class Choice {

  const Choice({this.title, this.icon});

  final String? title;

  final IconData? icon;

}

const List<Choice> choices = <Choice>[

  Choice(title: 'Home', icon: Icons.home),

  Choice(title: 'Contact', icon: Icons.contacts),

  Choice(title: 'Map', icon: Icons.map),

  Choice(title: 'Phone', icon: Icons.phone),

  Choice(title: 'Camera', icon: Icons.camera\_alt),

  Choice(title: 'Setting', icon: Icons.settings),

  Choice(title: 'Album', icon: Icons.photo\_album),

  Choice(title: 'WiFi', icon: Icons.wifi),

];

class SelectCard extends StatelessWidget {

  const SelectCard({Key? key, required this.choice}) : super(key: key);

  final Choice choice;

  @override

  Widget build(BuildContext context) {

    final TextStyle? textStyle = Theme.of(context).textTheme.headlineSmall;

    return Card(

        color: const Color.fromARGB(255, 252, 198, 117),

        child: Center(

          child: Column(

              crossAxisAlignment: CrossAxisAlignment.center,

              children: <Widget>[

                Expanded(

                    child:

                        Icon(choice.icon, size: 50.0, color: textStyle?.color)),

                Text(choice.title!, style: textStyle),

              ]),

        ));

  }

}

class GridViewCount3 extends StatelessWidget {

  const GridViewCount3({super.key});

  @override

  Widget build(BuildContext context) {

    return GridView.count(

      crossAxisCount: 3,

      crossAxisSpacing: 4.0,

      mainAxisSpacing: 8.0,

      children: List.generate(choices.length, (index) {

        return Center(

          child: SelectCard(choice: choices[index]),

        );

      }),

    );

  }

}

### GridView.custom

GridView.custom là cách mạnh mẽ để tạo một GridView tùy chỉnh, cho phép bạn kiểm soát hoàn toàn cách các phần tử (item) trong lưới được xây dựng dựa trên dữ liệu hoặc logic phức tạp của riêng bạn.

Khi bạn sử dụng GridView.custom, bạn cần cung cấp một SliverGridDelegate tùy chỉnh thông qua thuộc tính gridDelegate, và một SliverChildDelegate thông qua thuộc tính childrenDelegate. Điều này cho phép bạn kiểm soát cả cấu trúc của lưới và cách xây dựng các phần tử.

gridDelegate: có 2 option

* SliverGridDelegateWithFixedCrossAxisCount: chỉ định số column cho lưới qua thuộc tính crossAxisCount
* SliverGridDelegateWithMaxCrossAxisExtent: chỉ định chiều rộng tối đa cho mỗi column qua thuộc tính maxCrossAxisExtent

childrenDelegate: có 2 option

* SliverChildListDelegate: phù hợp khi bạn có một danh sách cố định của các widget con
* SliverChildBuilderDelegate: phù hợp khi bạn cần xây dựng danh sách các widget con dựa trên dữ liệu hoặc logic động.

Example 1: SliverGridDelegateWithFixedCrossAxisCount và SliverChildBuilderDelegate

Widget build(BuildContext context) {

    return GridView.custom(

      gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(

        crossAxisCount: 2, // Required

      ),

      childrenDelegate: SliverChildBuilderDelegate(

        (BuildContext context, int index) {

          // Xây dựng các phần tử dựa trên dữ liệu hoặc logic phức tạp

          return ListTile(title: Text('Item $index'));

        },

        childCount: 20, // Số lượng phần tử trong lưới, ko set sẽ scroll vô hạn

      ),

    );

  }

Example 2: SliverGridDelegateWithMaxCrossAxisExtent và SliverChildListDelegate

final List<String> items2 = List<String>.generate(30, (index) => 'Item $index');

class GridViewCustom2 extends StatelessWidget {

  const GridViewCustom2({super.key});

  @override

  Widget build(BuildContext context) {

    return GridView.custom(

        gridDelegate: const SliverGridDelegateWithMaxCrossAxisExtent(

          maxCrossAxisExtent: 200.0, // Required - Chiều rộng tối đa của mỗi ô con

          mainAxisSpacing: 10.0, // Khoảng cách giữa các ô con theo chiều dọc

          crossAxisSpacing: 10.0, // Khoảng cách giữa các ô con theo chiều ngang

        ),

        childrenDelegate: SliverChildListDelegate(

          items2.map((String item) {

            return Card(

                elevation: 2.0, // Độ nổi của thẻ

                child: Center(child: Text(item)));

          }).toList(),

        ));

  }

}

### GridView.extent

GridView.extent allows you to specify the maximum pixel width of a tile

GridView.extent được sử dụng để tạo một lưới với các ô con có kích thước xác định bởi maxCrossAxisExtent. Điều này cho phép bạn xác định chiều rộng tối đa của mỗi ô con theo chiều ngang hoặc chiều cao tối đa theo chiều dọc

 Widget build(BuildContext context) {

    return GridView.extent(

        maxCrossAxisExtent: 150.0, // Required - Chiều rộng tối đa của mỗi ô con

        mainAxisSpacing: 10.0, // Khoảng cách giữa các ô con theo chiều dọc

        crossAxisSpacing: 10.0, // Khoảng cách giữa các ô con theo chiều ngang

        children: List.generate(30, (index) {

          return Center(

              child: Container(

                  decoration: BoxDecoration(

                      color: const Color.fromARGB(255, 246, 235, 201),

                      border: Border.all(

                        color: Colors.red, // Màu của viền

                        width: 1.5, // Độ dày của viền

                      )),

                  // color: const Color.fromRGBO(239, 184, 184, 1),

                  child: Text('Item $index')));

        }));

  }

## PageView

A scrollable list that works page by page.

Each child of a page view is forced to be the same size as the viewport.

You can use a [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html) to control which page is visible in the view. In addition to being able to control the pixel offset of the content inside the [PageView](https://api.flutter.dev/flutter/widgets/PageView-class.html), a [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html) also lets you control the offset in terms of pages, which are increments of the viewport size.

[PageView](https://api.flutter.dev/flutter/widgets/PageView/PageView.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.horizontal, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html)? controller, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) pageSnapping = true, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[int](https://api.flutter.dev/flutter/dart-core/int-class.html)>? onPageChanged, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[], [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) allowImplicitScrolling = false, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)? scrollBehavior, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) padEnds = true})

Creates a scrollable list that works page by page from an explicit [List](https://api.flutter.dev/flutter/dart-core/List-class.html) of widgets.

class PageView1 extends StatelessWidget {

  const PageView1({super.key});

  @override

  Widget build(BuildContext context) {

    final PageController controller = PageController();

    return PageView(

      /// [PageView.scrollDirection] defaults to [Axis.horizontal].

      /// Use [Axis.vertical] to scroll vertically.

      controller: controller,

      reverse: true, // reverse order pages

      children: const <Widget>[

        Center(

          child: Text('First Page'),

        ),

        Center(

          child: Text('Second Page'),

        ),

        Center(

          child: Text('Third Page'),

        ),

      ],

    );

  }

}

[PageView.builder](https://api.flutter.dev/flutter/widgets/PageView/PageView.builder.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.horizontal, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html)? controller, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) pageSnapping = true, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[int](https://api.flutter.dev/flutter/dart-core/int-class.html)>? onPageChanged, required [NullableIndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/NullableIndexedWidgetBuilder.html) itemBuilder, [ChildIndexGetter](https://api.flutter.dev/flutter/widgets/ChildIndexGetter.html)? findChildIndexCallback, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? itemCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) allowImplicitScrolling = false, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)? scrollBehavior, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) padEnds = true})

Creates a scrollable list that works page by page using widgets that are created on demand.

// PageView2 - PageView.builder - Creates a scrollable list that works page by page using widgets that are created on demand.

class PageView2 extends StatelessWidget {

  PageView2({super.key});

  final List<String> imageUrls = [

    'https://images.pexels.com/photos/276267/pexels-photo-276267.jpeg',

    'https://images.pexels.com/photos/2280549/pexels-photo-2280549.jpeg',

    'https://images.pexels.com/photos/34299/herbs-flavoring-seasoning-cooking.jpg',

  ];

  @override

  Widget build(BuildContext context) {

    return PageView.builder(

      itemCount: imageUrls

          .length, // tránh lỗi: RangeError (RangeError (index): Invalid value: Not in inclusive range 0....)

      itemBuilder: (context, index) {

        return Image.network(

          imageUrls[index],

          fit: BoxFit.scaleDown,

        );

      },

    );

  }

}

//  PageController.initialPage - set the initial page

class PageView3 extends StatelessWidget {

  PageView3({super.key});

  final List<String> imageUrls = [

    'https://images.pexels.com/photos/276267/pexels-photo-276267.jpeg',

    'https://images.pexels.com/photos/2280549/pexels-photo-2280549.jpeg',

    'https://images.pexels.com/photos/34299/herbs-flavoring-seasoning-cooking.jpg',

  ];

  final PageController \_pageController = PageController(

      initialPage: 2); //PageController.initialPage - set the initial page

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Image Viewer'),

      ),

      body: PageView.builder(

        controller: \_pageController,

        itemCount: imageUrls.length,

        itemBuilder: (context, index) {

          return Image.network(

            imageUrls[index],

            fit: BoxFit.cover,

          );

        },

      ),

    );

  }

}

[PageView.custom](https://api.flutter.dev/flutter/widgets/PageView/PageView.custom.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.horizontal, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html)? controller, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) pageSnapping = true, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[int](https://api.flutter.dev/flutter/dart-core/int-class.html)>? onPageChanged, required [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html) childrenDelegate, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) allowImplicitScrolling = false, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)? scrollBehavior, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) padEnds = true})

Creates a scrollable list that works page by page with a custom child model.

// PageView.custom

class PageView4 extends StatelessWidget {

  PageView4({super.key});

  final List<String> data = ['Page 1', 'Page 2', 'Page 3'];

  @override

  Widget build(BuildContext context) {

    return PageView.custom(

      childrenDelegate: SliverChildBuilderDelegate(

        // required

        (context, index) {

          return Container(

            color: index % 2 == 0 ? Colors.blue : Colors.green,

            alignment: Alignment.center,

            child: Text(

              data[index],

              style: const TextStyle(fontSize: 20, color: Colors.white),

            ),

          );

        },

        childCount: data

            .length, // tránh lỗi: RangeError (RangeError (index): Invalid value: Not in inclusive range 0...)

      ),

      onPageChanged: (index) {

        // whenever the page changes

        debugPrint('Page changed to: $index');

        ScaffoldMessenger.of(context).showSnackBar(SnackBar(

            content: Text('This is a ${data[index]}'),

            duration: const Duration(seconds: 1)));

      },

      controller: PageController(initialPage: 1),

    );

  }

}

## Expanded

[Expanded](https://api.flutter.dev/flutter/widgets/Expanded/Expanded.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [int](https://api.flutter.dev/flutter/dart-core/int-class.html) flex = 1, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child})

Creates a widget that expands a child of a [Row](https://api.flutter.dev/flutter/widgets/Row-class.html), [Column](https://api.flutter.dev/flutter/widgets/Column-class.html), or [Flex](https://api.flutter.dev/flutter/widgets/Flex-class.html) so that the child fills the available space along the flex widget's main axis.

*const*

Expanded là một widget giúp mở rộng không gian đến giới hạn full của container cha cho một widget con của Row hoặc Column theo trục chính (main axis). Chú ý, trục chính của Row là trục nằm ngang, và trục chính của Column là trục thẳng đứng.

Một object con ko wrap với Extended – chỉ điều chỉnh kích thước theo trục main nếu vượt quá phạm vi cho phép, lấn các object con khác cho đến khi hết không gian lấn thì sẽ bị lỗi cảnh báo sọc vàng.

Một object con được wrap với Extended – auto điều chỉnh kích thước theo trục cross nếu vượt quá phạm vi cho phép, ko còn bị lỗi cảnh báo sọc vàng với object này (vẫn còn với các object con khác)

All object con wrap bởi Extended – all object auto điều chỉnh theo trục cross, ko còn lỗi cảnh báo sọc vàng

Thuộc tính flex: chỉ định tỉ lệ space cho mỗi Expanded

Example 1

 Widget build(BuildContext context) {

    return Container(

      // width: 200,

      // height: 600,

      color: const Color.fromARGB(255, 240, 195, 57),

      child: Row(

          children: [

            Expanded(

                child: ElevatedButton(

                    child: const Text("BTN 1"), onPressed: () {})),

            Expanded(

                flex: 1,

                child: ElevatedButton(

                    child: const Text("BTN 2"), onPressed: () {})),

            Expanded(

                flex: 0, // như object ko wrap Expanded

                child: ElevatedButton(

                    child: const Text("BTN 3"), onPressed: () {})),

            Expanded(

                flex: 2,

                child: ElevatedButton(

                    child: const Text("BTN 4"), onPressed: () {})),

            ElevatedButton(child: const Text("BTN 5"), onPressed: () {}),

          ]),

    );

  }

\*Lưu ý:

- Object ko wrap bởi Expanded có kích thước = object có flex: 0 – kích thước default của object

- Object wrap với Expanded nhưng ko set flex: lấn full space của container cha

- Nhiều object được set flex: n – các object chia nhau space còn trống theo tỉ lệ đã set

Thuộc tính flex: còn áp dụng cho các widget khác như Spacer, Flexible,…

## Flexible

[Flexible](https://api.flutter.dev/flutter/widgets/Flexible/Flexible.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [int](https://api.flutter.dev/flutter/dart-core/int-class.html) flex = 1, [FlexFit](https://api.flutter.dev/flutter/rendering/FlexFit.html) fit = FlexFit.loose, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child})

Creates a widget that controls how a child of a [Row](https://api.flutter.dev/flutter/widgets/Row-class.html), [Column](https://api.flutter.dev/flutter/widgets/Column-class.html), or [Flex](https://api.flutter.dev/flutter/widgets/Flex-class.html) flexes.

*const*

Flexible widget điều khiển không gian hiển thị một widget con trong một Row, Column hoặc Flex.

Khác với Expanded, Flexible không yêu cầu widget con phải lấp đầy không gian khả dụng.

Widget Flexible phải là một con của một Row, Column hoặc Flex and the path from the Flexible widget to its enclosing Row, Column, or Flex must contain only StatelessWidgets or StatefulWidgets (not other kinds of widgets, like RenderObjectWidgets).

Thuộc tính flex: (default = 1)

Widget build(BuildContext context) {

    return Column(

      children: <Widget>[

        Flexible(

            flex: 3,

            child: Container(

              // height: 150,

              color: Color.fromARGB(255, 249, 224, 32),

            )),

        Flexible(

            // flex: 1, // default flex = 1

            child: Container(

          color: Colors.green,

        )),

        Flexible(

            flex: 2,

            child: Container(

              color: Colors.blue,

            )),

      ],

    );

  }

Lưu ý:

Nếu Ko set flex all – flex chia đều

Nếu Ko set flex all, set height 1 element – flex chia đều xong, set height cho element đó (ko vượt height max của container cha)

height element <= (height container cha : số elements) 🡪 set height element lớn hơn height container cha : số elements ko có tác dụng

Khi đã set height 1 element thì các element khác ko thể set height > (height container cha : số elements) dù space container còn trống 🡪 để tăng kích thước element phải tăng thuôc tính flex lên

 Widget build(BuildContext context) {

    return Container(

      color: Color.fromARGB(255, 195, 188, 195),

      height: 600,

      child: Column(

        children: <Widget>[

          Flexible(

              // flex: 2,

              child: Container(

            height: 150,

            color: const Color.fromARGB(255, 249, 224, 32),

          )),

          Flexible(

              // flex: 1, // default flex = 1

              // flex: 3,

              child: Container(

            height: 100,

            color: Colors.green,

          )),

          Flexible(

              flex: 1,

              child: Container(

                height: 220,

                color: Colors.blue,

              )),

        ],

      ),

    );

  }

Thuộc tính fit của widget Flexible trong Flutter được sử dụng để xác định cách mà widget con của Flexible được co dãn hoặc mở rộng để lấp đầy không gian có sẵn.

FlexFit.tight: Widget con sẽ được mở rộng để lấp đầy không gian có sẵn. Điều này có nghĩa là widget con sẽ mở rộng theo hướng mà nó được đặt trong một Row hoặc Column. (Lúc này Flexible y như Expanded)

FlexFit.loose: Widget con sẽ được co dãn để vừa với không gian có sẵn. Điều này có thể dẫn đến việc widget con không lấp đầy toàn bộ không gian có thể có. (Khác Expended: không gian cho object con chỉ vừa đủ chứa nó)

## Spacer

[Spacer](https://api.flutter.dev/flutter/widgets/Spacer/Spacer.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [int](https://api.flutter.dev/flutter/dart-core/int-class.html) flex = 1})

Creates a flexible space to insert into a [Flexible](https://api.flutter.dev/flutter/widgets/Flexible-class.html) widget.

*const*

Spacer tạo ra một khoảng không gian trống, có thể điều chỉnh được, được sử dụng để điều chỉnh khoảng cách giữa các Widget con bên trong một flex container như Column, Row, ..

Thuộc tính flex: set % space chiếm giữ trong container cha (default flex: 1)

Tip: What’s the difference between SizedBox and Spacer? Use Spacer when you want to create space using a flex property. Use SizedBox when you want to create space using a specific number of logical pixels.

return Container(

      height: 400,

      color: Colors.amber,

      child: Column(

        children: [

          OutlinedButton(onPressed: () {}, child: const Text('Button A')),

          const Spacer(), // = Spacer(flex: 1),

          OutlinedButton(onPressed: () {}, child: const Text('Button B')),

          const Spacer(flex: 2),

          OutlinedButton(onPressed: () {}, child: const Text('Button C')),

          const Spacer(flex: 2),

          OutlinedButton(onPressed: () {}, child: const Text('Button D')),

        ],

      ),

    );

  }

## Divider

Divider là một widget được sử dụng để tạo ra một đường phân chia ngang trong giao diện của ứng dụng. Đây thường được sử dụng để tách các phần khác nhau của giao diện, như danh sách dữ liệu, hoặc để tạo ra một phân cách giữa các phần tử trên màn hình.

 ListTile(title: Text('Item 1')),

 Divider(), // Đường phân chia ngang

 ListTile(title: Text('Item 2')),

## Flex widget

[Flex](https://api.flutter.dev/flutter/widgets/Flex/Flex.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [Axis](https://api.flutter.dev/flutter/painting/Axis.html) direction, [MainAxisAlignment](https://api.flutter.dev/flutter/rendering/MainAxisAlignment.html) mainAxisAlignment = MainAxisAlignment.start, [MainAxisSize](https://api.flutter.dev/flutter/rendering/MainAxisSize.html) mainAxisSize = MainAxisSize.max, [CrossAxisAlignment](https://api.flutter.dev/flutter/rendering/CrossAxisAlignment.html) crossAxisAlignment = CrossAxisAlignment.center, [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection, [VerticalDirection](https://api.flutter.dev/flutter/painting/VerticalDirection.html) verticalDirection = VerticalDirection.down, [TextBaseline](https://api.flutter.dev/flutter/dart-ui/TextBaseline.html)? textBaseline, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[]})

Creates a flex layout.

*const*

Flex widget hiển thị các thành phần con của nó trong một mảng một chiều.

Widget Flex cho phép bạn kiểm soát trục theo chiều mà các thành phần con được đặt (ngang hoặc dọc). Nếu bạn biết trước trục chính, hãy xem xét sử dụng Row (nếu nó là ngang) hoặc Column (nếu nó là dọc) thay vì sử dụng Flex, bởi vì điều này sẽ ngắn gọn hơn.

Widget Flex không có khả năng cuộn (và nó thường được coi là lỗi khi có nhiều thành phần con hơn trong một Flex so với không gian có sẵn). Nếu bạn có một số widget và muốn chúng có thể cuộn nếu không đủ không gian, xem xét sử dụng ListView.

Widget Flex không cho phép các thành phần con của nó bọc quanh nhiều dòng ngang hoặc dọc. Đối với một widget cho phép các thành phần con bọc quanh nhiều dòng, hãy xem xét sử dụng widget Wrap thay vì Flex.

Nếu bạn chỉ có một thành phần con, thay vì sử dụng Flex, Row hoặc Column, hãy xem xét sử dụng Align hoặc Center để vị trí cho thành phần con.

## Stack

[Stack](https://api.flutter.dev/flutter/widgets/Stack/Stack.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html) alignment = AlignmentDirectional.topStart, [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection, [StackFit](https://api.flutter.dev/flutter/rendering/StackFit.html) fit = StackFit.loose, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[]})

By default, the non-positioned children of the stack are aligned by their top left corners.

Trong Flutter, Stack là một bộ chứa cho phép đặt các widget con của nó chồng lên nhau, widget con đầu tiên (chiều code từ trên xuống) sẽ được đặt ở dưới cùng. Stack là một giải pháp để tiết kiệm không gian của ứng dụng. Bạn có thể thay đổi thứ tự của các widget con để tạo ra hiệu ứng hoạt hình đơn giản.

Stack xắp xếp các widget con của nó theo nguyên tắc: widget con đầu tiên (chiều code từ trên xuống) sẽ được đặt ở dưới cùng, widget con mới nhất sẽ được đặt ở trên cùng. Khi bạn thay đổi thứ tự của các widget con thì Stack sẽ được vẽ lại. Nếu số lượng và thứ tự các widget con thay đổi liên tục, mỗi widget con cần được cung cấp một giá trị Key cụ thể và duy nhất, điều này giúp Stack quản lý hiệu quả các widget con.

Example 1

 Widget build(BuildContext context) {

    return Stack(

      alignment: Alignment.centerLeft, // align all object so container cha

      // alignment: AlignmentDirectional.bottomCenter, // or AlignmentDirectional.bottomEnd

      // textDirection: TextDirection.rtl, // chiều đổ stack

      children: <Widget>[

        Container(

          width: 290,

          height: 190,

          color: Colors.green,

        ),

        Container(

          width: 300,

          height: 170,

          color: Colors.red,

        ),

        Container(

          width: 220,

          height: 400,

          color: Colors.yellow,

        ),

      ],

    );

  }

Example – một số thuộc tính khác

  Widget build(BuildContext context) {

    return Container(

        color: Colors.grey,

        // height: 400,

        width: 500,

        child: Stack(

          // alignment:Alignment.center, // align all object so container cha

          alignment: AlignmentDirectional

              .bottomEnd, // or AlignmentDirectional.bottomEnd

          // textDirection: TextDirection.rtl, // chiều đổ stack

          // fit: StackFit.passthrough,

          // Default is StackFit.loose - ko thay đổi kích thước;

          //StackFit.expand - bung full 2 chiều container cha;

          // StackFit.passthrough - nếu container cha có set height thì bung full width của container cha, nếu container cha có set width thì bung full height của container cha, nếu set cả height & width thì như expand;

          // clipBehavior: Clip.none,

          children: <Widget>[

            Container(

              width: 300,

              height: 190,

              color: Colors.green,

            ),

            Container(

              width: 280,

              height: 170,

              color: Colors.red,

            ),

            Container(

              width: 220,

              height: 520,

              color: Colors.yellow,

            ),

          ],

        ));

  }

  Widget build(BuildContext context) {

    return Container(

        color: Colors.grey,

        height: 400,

        width: 500,

        child: Stack(

clipBehavior: Clip.none

          // alignment:Alignment.center, // align các object trên so với object dưới cùng

          alignment: AlignmentDirectional

              .bottomEnd, // or AlignmentDirectional.bottomEnd

          clipBehavior: Clip

              .none, // xử lý khi object tràn khỏi container (set Positioned(top, left...)). Clip.none - show full object, các value khác - cắt phần tràn ngoài container

          children: <Widget>[

            Container(

              width: 300,

              height: 190,

              color: Colors.green,

            ),

            Container(

              width: 280,

              height: 170,

              color: Colors.red,

            ),

            Align(

              alignment: Alignment.bottomLeft,

              child: Container(

                width: 200,

                height: 550,

                color: Colors.yellow,

              ),

            ),

            Positioned(

              bottom: 0,

              left: 300,

              child: Container(

                width: 220,

                height: 520,

                color: const Color.fromARGB(255, 255, 59, 232),

              ),

            ),

          ],

        ));

  }

9 vị trí align object A so với object B

 canh object A theo object B

  @override

  Widget build(BuildContext context) {

    double parentWidth = 400;

    double parentHeight = 380;

    double childWidth = 200;

    double childHeight = 120;

    double topCenter = (parentHeight - childHeight) / 2;

    double rightCenter = (parentWidth - childWidth) / 2;

    double heightOffset = parentHeight - childHeight;

    double widthOffset = parentWidth - childWidth;

    return Stack(

      clipBehavior: Clip.none,

      alignment: Alignment.center,

      children: <Widget>[

        Container(

          width: parentWidth,

          height: parentHeight,

          color: Colors.green,

        ),

        // align center

        Positioned(

          top: topCenter,

          right: rightCenter,

          child: Container(

            width: childWidth,

            height: childHeight,

            color: Colors.red,

          ),

        ),

        // align bottom - left

        Positioned(

          top: heightOffset,

          right: widthOffset,

          child: Container(

            width: childWidth,

            height: childHeight,

            color: const Color.fromARGB(255, 124, 54, 244),

          ),

        ),

        // align bottom - right

        Positioned(

          top: heightOffset,

          left: widthOffset,

          child: Container(

            width: childWidth,

            height: childHeight,

            color: Color.fromARGB(255, 244, 238, 54),

          ),

        ),

        // align top - right

        Positioned(

          bottom: heightOffset,

          left: widthOffset,

          child: Container(

            width: childWidth,

            height: childHeight,

            color: Color.fromARGB(255, 244, 54, 228),

          ),

        ),

        // align top - left

        Positioned(

          bottom: heightOffset,

          right: widthOffset,

          child: Container(

            width: childWidth,

            height: childHeight,

            color: Color.fromARGB(255, 101, 219, 240),

          ),

        ),

        // align center - left

        Positioned(

          bottom: topCenter,

          right: widthOffset,

          child: Container(

            width: childWidth,

            height: childHeight,

            color: Color.fromARGB(255, 101, 240, 191),

          ),

        ),

        // align center - right

        Positioned(

          bottom: topCenter,

          left: widthOffset,

          child: Container(

            width: childWidth,

            height: childHeight,

            color: Color.fromARGB(255, 240, 177, 101),

          ),

        ),

        // align center - bottom

        Positioned(

          top: heightOffset,

          left: rightCenter,

          child: Container(

            width: childWidth,

            height: childHeight,

            color: Color.fromARGB(255, 121, 174, 168),

          ),

        ),

        // align center - top

        Positioned(

          bottom: heightOffset,

          left: rightCenter,

          child: Container(

            width: childWidth,

            height: childHeight,

            color: Color.fromARGB(255, 230, 175, 199),

          ),

        ),

      ],

    );

  }

### Positioned class

A widget that controls where a child of a Stack is positioned.

A Positioned widget must be a descendant of a Stack.

## ListTile

[ListTile](https://api.flutter.dev/flutter/material/ListTile/ListTile.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? leading, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? title, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? subtitle, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? trailing, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isThreeLine = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? dense, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)? shape, [ListTileStyle](https://api.flutter.dev/flutter/material/ListTileStyle.html)? style, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectedColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? iconColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? textColor, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? titleTextStyle, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? subtitleTextStyle, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? leadingAndTrailingTextStyle, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? contentPadding, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) enabled = true, [GestureTapCallback](https://api.flutter.dev/flutter/gestures/GestureTapCallback.html)? onTap, [GestureLongPressCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressCallback.html)? onLongPress, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) selected = false, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? focusColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? hoverColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? splashColor, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? tileColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectedTileColor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enableFeedback, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? horizontalTitleGap, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? minVerticalPadding, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? minLeadingWidth, [ListTileTitleAlignment](https://api.flutter.dev/flutter/material/ListTileTitleAlignment.html)? titleAlignment})

Creates a list tile.

ListTile là một widget trong Flutter được thiết kế để hiển thị một hàng dữ liệu hoặc mục trong danh sách. Nó thường được sử dụng trong các danh sách dọc (vertical lists) như ListView để hiển thị thông tin trong các ô hoặc hàng riêng biệt.

Use ListTile, a specialized row widget from the Material library, for an easy way to create a row containing up to 3 lines of text and optional leading and trailing icons. ListTile is most commonly used in Card or ListView, but can be used elsewhere.

Một ListTile thông thường thường bao gồm các phần tử sau:

* leading: Một phần tử đầu tiên thường là một biểu tượng (icon) hoặc hình ảnh hiển thị bên trái của dòng dữ liệu.
* title: Một phần tử tiêu đề chứa nội dung chính của dòng dữ liệu.
* subtitle: Một phần tử phụ trợ chứa thông tin bổ sung về dòng dữ liệu.
* trailing: Một phần tử cuối cùng thường là một biểu tượng hoặc hình ảnh hiển thị bên phải của dòng dữ liệu.

Example 1

 Widget build(BuildContext context) {

    return const ListTile(

      leading: Icon(Icons.star),

      title: Text('ListTile Example'),

      subtitle: Text('This is a ListTile in a Flutter app.'),

      trailing: Icon(Icons.arrow\_forward),

    );

  }

Example 2 –

  Widget build(BuildContext context) {

    return Column(

      children: [

        const ListTile(

          leading: Icon(Icons.star),

          title: Text('ListTile Example'),

          subtitle: Text('This is a ListTile in a Flutter app.'),

          trailing: Icon(Icons.arrow\_forward),

        ),

        const ListTile(

          leading: Icon(Icons.album),

          title: Text('The Enchanted Nightingale'),

          subtitle: Text('Music by Julie Gable. Lyrics by Sidney Stein.'),

          trailing: Icon(Icons.arrow\_back\_ios),

          // set color

          // tileColor: Color.fromARGB(255, 248, 173, 167),

          iconColor: Color.fromARGB(255, 90, 113, 239),

          focusColor: Colors.greenAccent,

          hoverColor: Colors.limeAccent,

          selectedColor: Colors.indigo,

          selectedTileColor: Colors.purple,

          //

          // minLeadingWidth:

          //     40, // Đặt giá trị tối thiểu cho chiều rộng của leading

        ),

        // dùng ListTileTheme - shape: để tạo border cho ListTile

        ListTileTheme(

          shape: RoundedRectangleBorder(

            borderRadius: BorderRadius.circular(10.0),

            side: const BorderSide(

                color: Color.fromARGB(255, 79, 22, 236), width: 1.0),

          ),

          child: const ListTile(

            leading: Icon(Icons.album, color: Colors.cyan, size: 45),

            title: Text(

              "Let's Talk About Love",

              style: TextStyle(fontSize: 20),

            ),

            subtitle: Text('Modern Talking Album'),

            //

            horizontalTitleGap:

                40, // space giữa icon và title, subtitle (ko có leading ko tác dụng)

            minLeadingWidth: 50, // giá trị tối thiểu cho chiều rộng của leading

            minVerticalPadding:

                40, // The minimum padding on the top + bottom of the title and subtitle widgets.

          ),

        ),

        ListTileTheme(

          shape: RoundedRectangleBorder(

            borderRadius: BorderRadius.circular(10.0),

            side: const BorderSide(

                color: Color.fromARGB(255, 79, 22, 236), width: 2.0),

          ),

          child: const ListTile(

            tileColor: Color.fromARGB(255, 237, 203, 227),

            leading: Icon(Icons.album, color: Colors.cyan, size: 45),

            title: Text(

              "Let's Talk About Love",

              style: TextStyle(fontSize: 20),

            ),

            subtitle: Text('Modern Talking Album'),

          ),

        ),

// ListTile chứa nhiều loại object khác

        ListTile(

          leading: Checkbox(

            value: false, // Đặt giá trị ban đầu của Checkbox

            onChanged: (bool? value) {

              // Xử lý sự kiện thay đổi giá trị của Checkbox

              print("Checkbox value changed: $value");

            },

          ),

          title: const Text('Checkbox ListTile Example'),

          subtitle: const Text('This ListTile has a Checkbox.'),

          trailing: const Icon(Icons.arrow\_forward\_sharp),

          //

          focusColor: Colors.purpleAccent,

          hoverColor: Colors.yellowAccent,

        ),

        ListTile(

          leading: Checkbox(

            value: false,

            onChanged: (bool? value) {

              print("Checkbox value changed: $value");

            },

          ),

          title: const Column(

            crossAxisAlignment: CrossAxisAlignment.start,

            children: [

              Text('Complex ListTile Example'),

              Text('This ListTile has multiple components.'),

              Text('This ListTile has multiple components.'),

            ],

          ),

          subtitle: const Row(

            children: [

              Icon(Icons.star),

              SizedBox(width: 4.0),

              Text('Subheading'),

            ],

          ),

          trailing: Image.asset(

              'assets/lion.jpg'), // Thay thế 'assets/image.png' bằng đường dẫn thật

        ),

      ],

    );

  }

## CircleAvatar class

[CircleAvatar](https://api.flutter.dev/flutter/material/CircleAvatar/CircleAvatar.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)>? backgroundImage, [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)>? foregroundImage, [ImageErrorListener](https://api.flutter.dev/flutter/painting/ImageErrorListener.html)? onBackgroundImageError, [ImageErrorListener](https://api.flutter.dev/flutter/painting/ImageErrorListener.html)? onForegroundImageError, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? foregroundColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? radius, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? minRadius, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? maxRadius})

Creates a circle that represents a user.

A circle that represents a user.

Typically used with a user's profile image, or, in the absence of such an image, the user's initials. A given user's initials should always be paired with the same background color, for consistency.

If foregroundImage fails then backgroundImage is used. If backgroundImage fails too, backgroundColor is used.

The onBackgroundImageError parameter must be null if the backgroundImage is null. The onForegroundImageError parameter must be null if the foregroundImage is null.

 children: [

        const CircleAvatar(

            backgroundImage: NetworkImage(

                'https://icon-library.com/images/avatar-icon-images/avatar-icon-images-4.jpg')),

        const CircleAvatar(

          backgroundColor: Color.fromARGB(255, 131, 230, 243),

          child: Text('AH'),

        ),

## Switch class

[Switch](https://api.flutter.dev/flutter/material/Switch/Switch.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) value, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onChanged, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeTrackColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? inactiveThumbColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? inactiveTrackColor, [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)>? activeThumbImage, [ImageErrorListener](https://api.flutter.dev/flutter/painting/ImageErrorListener.html)? onActiveThumbImageError, [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)>? inactiveThumbImage, [ImageErrorListener](https://api.flutter.dev/flutter/painting/ImageErrorListener.html)? onInactiveThumbImageError, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? thumbColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? trackColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? trackOutlineColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)?>? trackOutlineWidth, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Icon](https://api.flutter.dev/flutter/widgets/Icon-class.html)?>? thumbIcon, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? focusColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? hoverColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? splashRadius, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false})

Creates a Material Design switch.

[Switch.adaptive](https://api.flutter.dev/flutter/material/Switch/Switch.adaptive.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) value, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onChanged, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeTrackColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? inactiveThumbColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? inactiveTrackColor, [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)>? activeThumbImage, [ImageErrorListener](https://api.flutter.dev/flutter/painting/ImageErrorListener.html)? onActiveThumbImageError, [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)>? inactiveThumbImage, [ImageErrorListener](https://api.flutter.dev/flutter/painting/ImageErrorListener.html)? onInactiveThumbImageError, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? thumbColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? trackColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? trackOutlineColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)?>? trackOutlineWidth, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Icon](https://api.flutter.dev/flutter/widgets/Icon-class.html)?>? thumbIcon, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? focusColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? hoverColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? splashRadius, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? applyCupertinoTheme})

Creates an adaptive [Switch](https://api.flutter.dev/flutter/material/Switch-class.html) based on whether the target platform is iOS or macOS, following Material design's [Cross-platform guidelines](https://material.io/design/platform-guidance/cross-platform-adaptation.html).

Được sử dụng để chuyển đổi trạng thái bật/tắt.

Chính switch không duy trì bất kỳ trạng thái nào. Thay vào đó, khi trạng thái của switch thay đổi, widget sẽ gọi hàm gọi lại onChanged. Hầu hết các widget sử dụng switch sẽ lắng nghe hàm onChanged và xây dựng lại switch với giá trị mới để cập nhật giao diện của switch.

Nếu hàm onChanged là null, thì switch sẽ bị vô hiệu hóa (nó sẽ không phản hồi đối với đầu vào). Nút và đường dẫn của switch bị vô hiệu hóa được hiển thị với các tông màu xám theo mặc định. Giao diện mặc định của switch bị vô hiệu hóa có thể được ghi đè bằng cách sử dụng inactiveThumbColor và inactiveTrackColor.

Yêu cầu widget cha của nó phải là Material widget.

Material Design 3 cung cấp tùy chọn để thêm biểu tượng trên nút bấm của Switch. Nếu ThemeData.useMaterial3 được đặt thành true, người dùng có thể sử dụng Switch.thumbIcon để thêm Biểu tượng tùy chọn dựa trên các trạng thái khác nhau của Switch.

Example

class SwitchExample extends StatefulWidget {

  const SwitchExample({super.key});

  @override

  State<SwitchExample> createState() => \_SwitchExampleState();

}

class \_SwitchExampleState extends State<SwitchExample> {

  // định nghĩa các thuôc tính, hàm cho Switch sử dụng

  // \* thuộc tính value: nếu các switch gán trùng tên biến thì các switch này sẽ update trạng thái cùng lúc với nhau

  // switch 1

  bool light = false;

// switch 2

  bool \_isSwitchOn = true;

  void \_toggleSwitch(bool value) {

    setState(() {

      \_isSwitchOn = value;

    });

  }

  // switch 3

  bool light\_3 = false;

  final MaterialStateProperty<Icon?> thumbIcon =

      MaterialStateProperty.resolveWith<Icon?>(

    (Set<MaterialState> states) {

      if (states.contains(MaterialState.selected)) {

        return const Icon(Icons.check);

      }

      return const Icon(Icons.close);

    },

  );

// switch 4

  bool light\_4 = false;

  @override

  Widget build(BuildContext context) {

    return Column(

      children: [

        Text('Switch Status: $light'),

        Switch(

          // This bool value toggles the switch.

          value: light, // set value true/false ban đầu

          activeColor: Colors.red,

          onChanged: (bool value) {

            // This is called when the user toggles the switch.

            setState(() {

              light = value; // switch value (true/false) khi click switch

            });

          },

        ),

        const Divider(),

        Text('Switch Status: $\_isSwitchOn'),

        Switch(

          value: \_isSwitchOn, // set value true/false ban đầu

          onChanged:

              \_toggleSwitch, // switch value (true/false) khi click switch qua hàm \_toggleSwitch

        ),

        Switch(

          thumbIcon: thumbIcon, // edit icon cho 2 trạng thái true/false

          value: light\_3,

          onChanged: (bool value) {

            setState(() {

              light\_3 = value;

            });

          },

        ),

        Text('Switch Status: $light\_4'),

        Switch(

          value: light\_4,

          onChanged: (bool value) {

            setState(() {

              light\_4 = value;

            });

          },

          activeColor: Colors.green, // Màu nút khi switch bật

          inactiveThumbColor:

              Color.fromARGB(255, 215, 215, 215), // Màu nút bấm khi switch tắt

          inactiveTrackColor: Colors.red, // Màu đường dẫn khi switch tắt

          activeTrackColor: Colors.amber, // Màu đường dẫn khi switch bật

          // set màu qua thumbColor:

          thumbColor: MaterialStateProperty.resolveWith<Color>(

              (Set<MaterialState> states) {

            if (states.contains(MaterialState.selected)) {

              return Colors

                  .blue; // Màu nút bấm khi switch bật - ghi đè activeColor:

            }

            return Colors

                .grey; // Màu nút bấm khi switch tắt - ghi đè inactiveThumbColor:

          }),

        ),

      ],

    );

  }

}

## SwitchListTile

[SwitchListTile](https://api.flutter.dev/flutter/material/SwitchListTile/SwitchListTile.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) value, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onChanged, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeTrackColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? inactiveThumbColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? inactiveTrackColor, [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)>? activeThumbImage, [ImageErrorListener](https://api.flutter.dev/flutter/painting/ImageErrorListener.html)? onActiveThumbImageError, [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)>? inactiveThumbImage, [ImageErrorListener](https://api.flutter.dev/flutter/painting/ImageErrorListener.html)? onInactiveThumbImageError, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? thumbColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? trackColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? trackOutlineColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Icon](https://api.flutter.dev/flutter/widgets/Icon-class.html)?>? thumbIcon, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? splashRadius, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? tileColor, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? title, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? subtitle, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isThreeLine = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? dense, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? contentPadding, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? secondary, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) selected = false, [ListTileControlAffinity](https://api.flutter.dev/flutter/material/ListTileControlAffinity.html) controlAffinity = ListTileControlAffinity.platform, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)? shape, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectedTileColor, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enableFeedback, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? hoverColor})

Creates a combination of a list tile and a switch.

*const*

[SwitchListTile.adaptive](https://api.flutter.dev/flutter/material/SwitchListTile/SwitchListTile.adaptive.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) value, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onChanged, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeTrackColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? inactiveThumbColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? inactiveTrackColor, [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)>? activeThumbImage, [ImageErrorListener](https://api.flutter.dev/flutter/painting/ImageErrorListener.html)? onActiveThumbImageError, [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)>? inactiveThumbImage, [ImageErrorListener](https://api.flutter.dev/flutter/painting/ImageErrorListener.html)? onInactiveThumbImageError, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? thumbColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? trackColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? trackOutlineColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Icon](https://api.flutter.dev/flutter/widgets/Icon-class.html)?>? thumbIcon, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? splashRadius, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? applyCupertinoTheme, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? tileColor, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? title, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? subtitle, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isThreeLine = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? dense, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? contentPadding, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? secondary, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) selected = false, [ListTileControlAffinity](https://api.flutter.dev/flutter/material/ListTileControlAffinity.html) controlAffinity = ListTileControlAffinity.platform, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)? shape, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectedTileColor, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enableFeedback, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? hoverColor})

Creates a Material [ListTile](https://api.flutter.dev/flutter/material/ListTile-class.html) with an adaptive [Switch](https://api.flutter.dev/flutter/material/Switch-class.html), following Material design's [Cross-platform guidelines](https://material.io/design/platform-guidance/cross-platform-adaptation.html).

*const*

Example 1

class \_SwitchListTile1State extends State<SwitchListTile1> {

  // bool switchValue1 = true;

  // bool switchValue2 = true;

  // bool switchValue3 = true;

  bool valueInit = true;

  late bool switchValue1 = valueInit,

      switchValue2 = valueInit,

      switchValue3 = valueInit;

  @override

  Widget build(BuildContext context) {

    return Column(

      children: [

        SwitchListTile(

          title: const Text('Lights'),

          value: switchValue1,

          onChanged: (bool value) {

            setState(() {

              switchValue1 = value;

              debugPrint('$switchValue1');

            });

          },

          secondary: const Icon(Icons.lightbulb\_outline), // add icon

        ),

        const Divider(),

        SwitchListTile(

          title: const Text('Station'),

          value: switchValue2,

          onChanged: (bool value) {

            setState(() {

              switchValue2 = value;

              debugPrint('$switchValue2');

            });

          },

          secondary: const Icon(Icons.ev\_station),

        ),

        const Divider(),

        SwitchListTile(

          title: const Text('account\_balance\_sharp'),

          value: switchValue3,

          onChanged: (bool value) {

            setState(() {

              switchValue3 = value;

              debugPrint('$switchValue3');

            });

          },

          secondary: const Icon(Icons.account\_balance\_sharp),

        ),

        const Divider(),

      ],

    );

  }

}

Example 2 - Gán title: qua biến - đọc lại giá trị title: khi cần

class \_SwitchListTile2State extends State<SwitchListTile2> {

  bool valueInit = true;

  late bool switchValue1 = valueInit,

      switchValue2 = valueInit,

      switchValue3 = valueInit;

// Gán title: qua biến - đọc lại giá trị title: khi cần

  Widget titleWidget0 = const Text('Light'); // Giá trị của thuộc tính title

  Widget titleWidget1 = const Text('Station'); // Giá trị của thuộc tính title

  Widget titleWidget2 = const Text('Sharp'); // Giá trị của thuộc tính title

  @override

  Widget build(BuildContext context) {

    return Column(

      children: [

        SwitchListTile(

          title: titleWidget0, // Sử dụng biến titleWidget

          value: switchValue1,

          onChanged: (bool value) {

            setState(() {

              switchValue1 = value;

              debugPrint('$switchValue1');

              if (switchValue1) {

                String? titleValue = (titleWidget0 as Text).data;

                debugPrint('$titleValue');

              }

            });

          },

          secondary: const Icon(Icons.lightbulb\_outline),

          subtitle: const Text( // thêm subtitle

              'This sample demonstrates how SwitchListTile positions the switch widget relative to the text in different configurations.'),

        ),

        const Divider(),

        SwitchListTile(

          title: titleWidget1,

          value: switchValue2,

          onChanged: (bool value) {

            setState(() {

              switchValue2 = value;

              debugPrint('$switchValue2');

              if (switchValue2) {

                String? titleValue = (titleWidget1 as Text).data;

                debugPrint('$titleValue');

              }

            });

          },

          secondary: const Icon(Icons.ev\_station),

          subtitle: const Text(

              "Longer supporting text to demonstrate how the text wraps and how setting 'SwitchListTile.isThreeLine = true' aligns the switch to the top vertically with the text."),

        ),

        const Divider(),

        SwitchListTile(

          title: titleWidget2,

          value: switchValue3,

          onChanged: (bool value) {

            setState(() {

              switchValue3 = value;

              debugPrint('$switchValue3');

              if (switchValue3) {

                String? titleValue = (titleWidget2 as Text).data;

                debugPrint('$titleValue');

              }

            });

          },

          secondary: const Icon(Icons.account\_balance\_sharp),

          subtitle: const Text(

              "Longer supporting text to demonstrate how the text wraps and how settihe text."),

        ),

        const Divider(),

      ],

    );

  }

}

## Chip

[Chip](https://api.flutter.dev/flutter/material/Chip/Chip.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? avatar, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) label, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? labelStyle, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? labelPadding, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? deleteIcon, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onDeleted, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? deleteIconColor, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? deleteButtonTooltipMessage, [BorderSide](https://api.flutter.dev/flutter/painting/BorderSide-class.html)? side, [OutlinedBorder](https://api.flutter.dev/flutter/painting/OutlinedBorder-class.html)? shape, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? color, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? elevation, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? shadowColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? surfaceTintColor, [IconThemeData](https://api.flutter.dev/flutter/widgets/IconThemeData-class.html)? iconTheme, @[Deprecated](https://api.flutter.dev/flutter/dart-core/Deprecated-class.html)('Migrate to deleteButtonTooltipMessage. ' 'This feature was deprecated after v2.10.0-0.3.pre.') [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) useDeleteButtonTooltip = true})

Creates a Material Design chip.

Chip là một widget được sử dụng để hiển thị một phần tử ngắn gọn, thường là một từ hoặc một biểu tượng, để thể hiện thông tin như tag, thuộc tính hoặc tình trạng. Chip thường được sử dụng trong các trường hợp như danh sách các mục đánh dấu, lọc, tìm kiếm, và nhiều tình huống khác.

  return Column(

      children: [

        const Wrap(

          spacing: 8.0,

          children: [

            Chip(

              label: Text('Chip Flutter'), // The primary content of the chip.

Typically a [Text] widget.

              backgroundColor: Colors.blue,

              labelStyle: TextStyle(color: Colors.white),

            ),

            Chip(

              label: Text('Chip Dart'),

              backgroundColor: Colors.green,

              labelStyle: TextStyle(color: Colors.white),

            ),

            Chip(

              label: Text('Chip Mobile'),

              backgroundColor: Colors.orange,

              labelStyle: TextStyle(color: Colors.white),

            ),

          ],

        ),

        const Divider(),

        Wrap(

          spacing: 8.0,

          children: [

            const Chip(

              avatar: Icon(Icons.star, color: Colors.yellow), // A widget to display prior to the chip's label. Typically a [CircleAvatar] widget.

              label: Text('Favorite'),

              backgroundColor: Colors.blue,

              labelStyle: TextStyle(color: Colors.white),

            ),

            const Chip(

              avatar: CircleAvatar(

                backgroundColor: Colors.orange,

                child: Text('S'),

              ),

              label: Text('Dart'),

              backgroundColor: Color.fromARGB(255, 154, 226, 241),

              labelStyle: TextStyle(color: Colors.white),

            ),

            Chip(

              avatar: Image.asset('assets/image3.jpg',

                  width: 24,

                  height:

                      24), // Thay thế 'assets/image.png' bằng đường dẫn thật

              label: const Text('Image'),

              backgroundColor: Colors.orange,

              labelStyle: const TextStyle(color: Colors.white),

            ),

          ],

        ),

## ChoiceChip

[ChoiceChip](https://api.flutter.dev/flutter/material/ChoiceChip/ChoiceChip.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? avatar, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) label, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? labelStyle, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? labelPadding, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onSelected, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? pressElevation, required [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) selected, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectedColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? disabledColor, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? tooltip, [BorderSide](https://api.flutter.dev/flutter/painting/BorderSide-class.html)? side, [OutlinedBorder](https://api.flutter.dev/flutter/painting/OutlinedBorder-class.html)? shape, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? color, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? elevation, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? shadowColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? surfaceTintColor, [IconThemeData](https://api.flutter.dev/flutter/widgets/IconThemeData-class.html)? iconTheme, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectedShadowColor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? showCheckmark, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? checkmarkColor, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html) avatarBorder = const CircleBorder()})

Create a chip that acts like a radio button.

*const*

[ChoiceChip.elevated](https://api.flutter.dev/flutter/material/ChoiceChip/ChoiceChip.elevated.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? avatar, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) label, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? labelStyle, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? labelPadding, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onSelected, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? pressElevation, required [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) selected, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectedColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? disabledColor, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? tooltip, [BorderSide](https://api.flutter.dev/flutter/painting/BorderSide-class.html)? side, [OutlinedBorder](https://api.flutter.dev/flutter/painting/OutlinedBorder-class.html)? shape, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? color, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? elevation, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? shadowColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? surfaceTintColor, [IconThemeData](https://api.flutter.dev/flutter/widgets/IconThemeData-class.html)? iconTheme, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectedShadowColor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? showCheckmark, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? checkmarkColor, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html) avatarBorder = const CircleBorder()})

Create an elevated chip that acts like a radio button.

ChoiceChips represent a single choice from a set. Choice chips contain related descriptive text or categories.

Requires one of its ancestors to be a Material widget. The selected and label arguments must not be null.

   Wrap(

          spacing: 8,

          children: [

            ChoiceChip(

                label: const Text('ChoiceChip Item '),

                selected: true,

                onSelected: (bool selected) {

                  null;

                }),

            ChoiceChip(

                label: const Text('ChoiceChip Item '),

                selected: false,

                onSelected: (bool selected) {

                  null;

                }),

          ],

        ),

Nguyên lý hoạt động

class ChoiceChip1 extends StatefulWidget {

  const ChoiceChip1({super.key});

  @override

  State<ChoiceChip1> createState() => \_ChoiceChipState();

}

class \_ChoiceChipState extends State<ChoiceChip1> {

  int? \_value = 1;

  int \_selectedChipIndex = 0;

  void \_handleChipSelection(int index) {

    setState(() {

      \_selectedChipIndex = index;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Center(

      child: Column(

        crossAxisAlignment: CrossAxisAlignment.start,

        mainAxisAlignment: MainAxisAlignment.start,

        children: <Widget>[

          const Text('Choose an ChoiceChip item'),

          const SizedBox(height: 10.0),

          Wrap(

            spacing: 5,

            children: [

              ChoiceChip(

                label: const Text('Option 1'),

                selected: \_selectedChipIndex == 0,

                onSelected: (isSelected) => \_handleChipSelection(0),

                elevation: 5, // Đặt giá trị elevation

              ),

              const SizedBox(height: 16),

              ChoiceChip(

                label: const Text('Option 2'),

                selected: \_selectedChipIndex == 1,

                onSelected: (isSelected) => \_handleChipSelection(1),

                elevation: 5, // Đặt giá trị elevation

              ),

            ],

          ),

          const SizedBox(height: 10.0),

          const Text('Choose an ChoiceChip item - List<Widget>.generate'),

          Wrap(

            spacing: 5.0,

            children: List<Widget>.generate(

              3,

              (int index) {

                return ChoiceChip(

                  label: Text('Item $index'),

                  selected: \_value == index, // truyền true/false theo điều kiện

                  onSelected: (bool selected) {

                    // nếu selected = true thì gọi setState() để thực thi các lệnh trong nó và build lại UI

                    setState(() {

                      \_value = selected ? index : null;

                      print('ChoiceChip item $\_value');

                    });

                  },

                );

              },

            ).toList(),

          ),

        ],

      ),

    );

  }

}

Trong ChoiceChip và các loại chip có thuộc tính selected, mặc định ChoiceChip được thiết kế với biểu tượng "checked" chồng lên biểu tượng avatar. Bạn không thể đơn giản tách biểu tượng avatar ra khỏi biểu tượng "checked" một cách dễ dàng mà không sửa đổi một số khía cạnh trong giao diện mặc định.

Tuy nhiên, bạn có thể tạo một biểu tượng "checked" riêng và định vị nó bên phải hoặc bên trái của ChoiceChip thông qua cách tùy chỉnh ChoiceChip

  Widget build(BuildContext context) {

    return Column(

      mainAxisAlignment: MainAxisAlignment.center,

      children: [

        Row(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            ChoiceChip(

              label: const Text('Option 1'),

              selected: \_selectedChipIndex == 0,

              onSelected: (isSelected) => \_handleChipSelection(0),

            ),

            const SizedBox(width: 8),

            Icon(

              \_selectedChipIndex == 0

                  ? Icons.check\_circle

                  : Icons.check\_circle\_outline,

              color: \_selectedChipIndex == 0 ? Colors.blue : Colors.grey,

            ),

          ],

        ),

        const SizedBox(height: 16),

        Row(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            ChoiceChip(

              label: const Text('Option 2'),

              selected: \_selectedChipIndex == 1,

              onSelected: (isSelected) => \_handleChipSelection(1),

            ),

            const SizedBox(width: 8),

            Icon(

              \_selectedChipIndex == 1

                  ? Icons.check\_circle

                  : Icons.check\_circle\_outline,

              color: \_selectedChipIndex == 1 ? Colors.blue : Colors.grey,

            ),

          ],

        ),

      ],

    );

## InputChip

"input chips" đại diện cho một phần thông tin phức tạp, chẳng hạn như một thực thể (người, địa điểm hoặc đồ vật) trong một dạng thu gọn.

Các input chip có thể trở thành có thể chọn bằng cách thiết lập onSelected, có thể xóa bỏ bằng cách thiết lập onDeleted, và có thể nhấn giống như một nút bấm với onPressed. Chúng có một nhãn (label), và chúng có thể có một biểu tượng ở phần đầu (avatar) và một biểu tượng ở phần cuối (deleteIcon). Màu sắc và padding có thể được tùy chỉnh.

 Wrap(

          spacing: 8,

          children: [

            InputChip(

              label: const Text('InputChip 1'),

              selected: true,

              onSelected: (bool selected) {

                null;

              },

              onDeleted: () {

                null;

              },

            ),

            InputChip(

              label: const Text('InputChip 2'),

              selected: false,

              onSelected: (bool selected) {

                null;

              },

              onDeleted: () {

                null;

              },

            ),

          ],

        ),

Nguyên lý hoạt động

class InputChipExample extends StatefulWidget {

  const InputChipExample({super.key});

  @override

  State<InputChipExample> createState() => \_InputChipExampleState();

}

class \_InputChipExampleState extends State<InputChipExample> {

  int inputs = 3;

  int? selectedIndex;

  @override

  Widget build(BuildContext context) {

    return Center(

      child: Column(

        mainAxisSize: MainAxisSize.min,

        mainAxisAlignment: MainAxisAlignment.center,

        children: <Widget>[

          Wrap(

            alignment: WrapAlignment.center,

            spacing: 5.0,

            children: List<Widget>.generate(

              inputs,

              (int index) {

                return InputChip(

                  label: Text('Person ${index + 1}'),

//                   Whether or not this chip is selected.

// If [onSelected] is not null, this value will be used to determine if the select check mark will be shown or not.

// Must not be null. Defaults to false.

                  selected: selectedIndex == index,

                  onSelected: (bool selected) {

                    //Called when the chip should change between selected and de-selected states.

// When the chip is tapped, then the [onSelected] callback, if set, will be applied to !selected (see [selected]).

                    setState(() {

                      if (selectedIndex == index) {

                        // mới load lên selectedIndex = null nên ko có object nào checked

                        selectedIndex =

                            null; // nếu click trên object đang chọn (index đã chọn rồi) thì switch selectedIndex về null

                      } else {

                        selectedIndex = index;

                      }

                      print('selectedIndex: $selectedIndex');

                    });

                  },

                  onDeleted: () {

                    setState(() {

                      inputs = inputs - 1;

                    });

                  },

                );

              },

            ).toList(),

          ),

          const SizedBox(height: 10),

          ElevatedButton(

            onPressed: () {

              setState(() {

                inputs = 3;

              });

            },

            child: const Text('Reset'),

          )

        ],

      ),

    );

  }

}

## ActionChip

ActionChip là một widget trong Flutter, nằm trong gói material, được sử dụng để hiển thị một "chip" chứa một hành động mà người dùng có thể thực hiện, chẳng hạn như mở một cửa sổ mới hoặc thực hiện một tác vụ cụ thể.

Một ActionChip có thể chứa một biểu tượng hoặc văn bản mô tả hành động. Khi người dùng nhấn vào ActionChip, hành động tương ứng sẽ được thực hiện.

Example 1

class ActionChip2 extends StatelessWidget {

  const ActionChip2({super.key});

  @override

  Widget build(BuildContext context) {

    return ActionChip(

      label: const Text('Open Dialog'),

      onPressed: () {

        showDialog(

          context: context,

          builder: (BuildContext context) {

            return AlertDialog(

              title: const Text('Dialog Title'),

              content: const Text('This is a dialog content.'),

              actions: [

                TextButton(

                  onPressed: () {

                    Navigator.pop(context);

                  },

                  child: const Text('Close'),

                ),

              ],

            );

          },

        );

      },

    );

  }

}

Example 2

class ActionChip1 extends StatefulWidget {

  const ActionChip1({super.key});

  @override

  State<ActionChip1> createState() => \_ActionChipExampleState();

}

class \_ActionChipExampleState extends State<ActionChip1> {

  bool favorite = false;

  @override

  Widget build(BuildContext context) {

    return Center(

      child: ActionChip(

        avatar: Icon(favorite ? Icons.favorite : Icons.favorite\_border),

        label: const Text('Save to favorites'),

        onPressed: () {

          setState(() {

            favorite = !favorite;

          });

        },

      ),

    );

  }

}

## FilterChip

Filter chips use tags or descriptive words as a way to filter content.

Filter chips are a good alternative to Checkbox or Switch widgets. Unlike these alternatives, filter chips allow for clearly delineated and exposed options in a compact area.

Example 1

enum ExerciseFilter { walking, running, cycling, hiking }

class FilterChipExample extends StatefulWidget {

  const FilterChipExample({super.key});

  @override

  State<FilterChipExample> createState() => \_FilterChipExampleState();

}

class \_FilterChipExampleState extends State<FilterChipExample> {

  Set<ExerciseFilter> filters = <ExerciseFilter>{}; // định nghĩa Set rỗng để lưu các gtri từ ExerciseFilter – kết quả các value được chọn

  @override

  Widget build(BuildContext context) {

    final TextTheme textTheme = Theme.of(context).textTheme;

    return Center(

      child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        children: <Widget>[

          Text('Choose an exercise', style: textTheme.labelLarge),

          const SizedBox(height: 5.0),

          Wrap(

            spacing: 5.0,

            children: ExerciseFilter.values.map((ExerciseFilter exercise) {

              return FilterChip(

                label: Text(exercise.name),

                selected: filters.contains(exercise),

                onSelected: (bool selected) {

                  setState(() {

                    if (selected) {

                      filters.add(exercise);

                    } else {

                      filters.remove(exercise);

                    }

                  });

                },

              );

            }).toList(),

          ),

          const SizedBox(height: 10.0),

          Text(

            'Looking for: ${filters.map((ExerciseFilter e) => e.name).join(', ')}', // map .name từ ExerciseFilter vào filters

            style: textTheme.labelLarge,

          ),

        ],

      ),

    );

  }

}

Example 2 – FilterChip kết hợp CheckBox

List<String> \_selectedChips = [];

class FilterChip2 extends StatefulWidget {

  const FilterChip2({super.key});

  @override

  State<FilterChip2> createState() => \_FilterChip2State();

}

class \_FilterChip2State extends State<FilterChip2> {

  void \_handleChipSelection(String chipLabel) {

    setState(() {

      if (\_selectedChips.contains(chipLabel)) {

        \_selectedChips.remove(chipLabel);

      } else {

        \_selectedChips.add(chipLabel);

      }

    });

  }

  @override

  Widget build(BuildContext context) {

    return Wrap(

      spacing: 8.0,

      children: [

        for (String label in ['Tag 1', 'Tag 2', 'Tag 3'])

          FilterChip(

            label: Text(label),

            selected: \_selectedChips.contains(label),

            onSelected: (isSelected) => \_handleChipSelection(label),

// onSelected: (isSelected) – tham số tượnh trưng          ),

      ],

    );

  }

}

Example 2 – FilterAndCheckboxDemo

class FilterAndCheckboxDemo extends StatefulWidget {

  const FilterAndCheckboxDemo({super.key});

  @override

  State<FilterAndCheckboxDemo> createState() => \_FilterAndCheckboxDemoState();

}

List<String> \_filterOptions = ['Option 1', 'Option 2', 'Option 3'];

List<bool> \_selectedFilters = [false, false, false];

class \_FilterAndCheckboxDemoState extends State<FilterAndCheckboxDemo> {

  bool \_showChecked = false;

  void \_handleFilterSelection(int index) {

    setState(() {

      \_selectedFilters[index] = !\_selectedFilters[index];

    });

  }

  void \_handleCheckbox(bool value) {

    setState(() {

      \_showChecked = value;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Column(

      mainAxisAlignment: MainAxisAlignment.center,

      children: [

        Wrap(

          spacing: 8.0,

          children: [

            for (int i = 0; i < \_filterOptions.length; i++)

              FilterChip(

                label: Text(\_filterOptions[i]),

                selected: \_selectedFilters[i],

// onSelected: (\_) – truyền tham số tượng trưng, ko cần khai báo trước

                onSelected: (\_) => \_handleFilterSelection(i),

              ),

          ],

        ),

        const SizedBox(height: 16),

        Row(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            Checkbox(

              value: \_showChecked,

// onChanged: tham số tượng trưng

              onChanged: (newValue) {

                \_handleCheckbox(newValue!);

              },

            ),

            const Text('Only Show Checked Items'),

          ],

        ),

        const SizedBox(height: 16),

        ListView.builder(

          shrinkWrap: true,

          itemCount: \_filterOptions.length,

          itemBuilder: (context, index) {

            if (!\_showChecked || \_selectedFilters[index]) {

              return ListTile(

                title: Text(\_filterOptions[index]),

                trailing: \_selectedFilters[index]

                    ? const Icon(Icons.check)

                    : const Icon(Icons.check\_box\_outline\_blank),

              );

            } else {

              return const SizedBox.shrink();

            }

          },

        ),

      ],

    );

  }

}

Example 3 – Filter & CheckBox đồng bộ

class \_FilterChipAndCheckboxDemo2State

    extends State<FilterChipAndCheckboxDemo2> {

  List<String> \_options = ['FilterChip 1', 'FilterChip 2', 'FilterChip 3'];

  List<String> \_options2 = ['CheckBox 1', 'CheckBox 2', 'CheckBox 3'];

  List<bool> \_selectedItems = [false, false, false];

  void \_handleSelection(int index, bool value) {

    setState(() {

      \_selectedItems[index] = value;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Column(

      mainAxisAlignment: MainAxisAlignment.center,

      children: [

        Wrap(

          spacing: 8.0,

          children: List.generate(\_options.length, (index) {

            return FilterChip(

              label: Text(\_options[index]),

              selected: \_selectedItems[index],

              onSelected: (isSelected) => \_handleSelection(index, isSelected),

            );

          }),

        ),

        const SizedBox(height: 16),

        Wrap(

          spacing: 8.0,

          children: List.generate(\_options2.length, (index) {

            return Row(

              mainAxisSize: MainAxisSize.min,

              children: [

                Checkbox(

                  value: \_selectedItems[index],

                  onChanged: (value) => \_handleSelection(index, value!), // value! – khẳng định not null

                ),

                Text(\_options2[index]),

              ],

            );

          }),

        ),

      ],

    );

  }

}

Example 4 – Filter & Switch

List<String> \_filterOptions3 = ['Option a', 'Option b', 'Option c'];

List<bool> \_selectedFilters3 = [false, false, false];

List<bool> \_showSwitches = [false, false, false];

class \_FilterAndSwitchDemo3State extends State<FilterAndSwitchDemo3> {

  void \_handleFilterSelection(int index) {

    setState(() {

      \_selectedFilters3[index] = !\_selectedFilters3[index];

    });

  }

  void \_handleSwitch(int index, bool value) {

    setState(() {

      \_showSwitches[index] = value;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Column(

      mainAxisAlignment: MainAxisAlignment.center,

      children: [

        Wrap(

          spacing: 8.0,

          children: [

            for (int i = 0; i < \_filterOptions3.length; i++)

              Column(

                children: [

                  FilterChip(

                    label: Text(\_filterOptions3[i]),

                    selected: \_selectedFilters3[i],

                    onSelected: (\_) => \_handleFilterSelection(i),

                  ),

                  SizedBox(height: 8),

                  Switch(

                    value: \_showSwitches[i],

                    onChanged: (value) => \_handleSwitch(i, value),

                  ),

                ],

              ),

          ],

        ),

        SizedBox(height: 16),

        ListView.builder(

          shrinkWrap: true,

          itemCount: \_filterOptions3.length,

          itemBuilder: (context, index) {

            if (\_selectedFilters3[index] && \_showSwitches[index]) {

              return ListTile(

                title: Text(\_filterOptions3[index]),

                trailing: Icon(Icons.check),

              );

            } else {

              return SizedBox.shrink();

            }

          },

        ),

      ],

    );

  }

}

## CheckBox

[Checkbox](https://api.flutter.dev/flutter/material/Checkbox/Checkbox.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? value, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) tristate = false, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)?>? onChanged, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? fillColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? checkColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? focusColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? hoverColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? splashRadius, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [OutlinedBorder](https://api.flutter.dev/flutter/painting/OutlinedBorder-class.html)? shape, [BorderSide](https://api.flutter.dev/flutter/painting/BorderSide-class.html)? side, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isError = false, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticLabel})

Creates a Material Design checkbox.

*const*

[Checkbox.adaptive](https://api.flutter.dev/flutter/material/Checkbox/Checkbox.adaptive.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? value, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) tristate = false, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)?>? onChanged, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? fillColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? checkColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? focusColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? hoverColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? splashRadius, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [OutlinedBorder](https://api.flutter.dev/flutter/painting/OutlinedBorder-class.html)? shape, [BorderSide](https://api.flutter.dev/flutter/painting/BorderSide-class.html)? side, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isError = false, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticLabel})

Creates an adaptive [Checkbox](https://api.flutter.dev/flutter/material/Checkbox-class.html) based on whether the target platform is iOS or macOS, following Material design's [Cross-platform guidelines](https://material.io/design/platform-guidance/cross-platform-adaptation.html).

Checkbox trong Flutter là một thành phần giao diện người dùng (widget) cho phép người dùng chọn hoặc bỏ chọn một tùy chọn cụ thể. Nó thường được sử dụng để kiểm soát trạng thái đúng/sai hoặc chọn/bỏ chọn một tùy chọn nhất định.

Checkbox có hai trạng thái chính: đã chọn (checked) và không chọn (unchecked). Khi người dùng nhấn vào Checkbox, trạng thái sẽ thay đổi và giao diện người dùng sẽ tự động cập nhật để hiển thị trạng thái mới.

Dưới đây là một số thuộc tính quan trọng của Checkbox:

* value: Xác định trạng thái của Checkbox. Đặt thành true để chọn và false để không chọn.
* onChanged: Hàm callback sẽ được gọi khi trạng thái của Checkbox thay đổi. Nó thường được sử dụng để cập nhật giá trị value và cập nhật giao diện người dùng.
* Widget build(BuildContext context) {
* // Set color chung cho nhiều sự kiện
* Color getColorGeneral(Set<MaterialState> states) {
* const Set<MaterialState> interactiveStates = <MaterialState>{
* MaterialState.pressed,
* MaterialState.hovered,
* MaterialState.focused,
* MaterialState.selected,
* };
* if (states.any(interactiveStates.contains)) {
* return Colors.blue;
* }
* return Colors.red;
* }
* // Set color riêng cho từng sự kiện
* Color getColorSeparately(Set<MaterialState> states) {
* if (states.contains(MaterialState.pressed)) {
* // Nếu được nhấn
* return Colors.blue;
* } else if (states.contains(MaterialState.hovered)) {
* // Nếu được hover
* return Colors.green;
* } else if (states.contains(MaterialState.focused)) {
* // Nếu được focus
* return Colors.orange;
* } else if (states.contains(MaterialState.selected)) {
* // Nếu được chọn
* return Color.fromARGB(255, 54, 233, 236);
* }
* // Trạng thái mặc định
* return Colors.red;
* }
* return Column(
* mainAxisAlignment: MainAxisAlignment.center,
* children: [
* Checkbox(
* value: \_isChecked,
* onChanged: (newValue) {
* \_handleCheckbox(newValue!);
* },
* ),
* Text('Checkbox is $\_isChecked'),
* Checkbox(
* value: \_isChecked,
* onChanged: null, // bị mờ đi
* ),
* Text('Checkbox is $\_isChecked'),
* Checkbox(
* value: \_isChecked,
* onChanged: (bool? ts) {
* setState(() {
* \_isChecked = ts!;
* });
* },
* ),
* Text('Checkbox is $\_isChecked'),
* // set color inline
* Checkbox(
* checkColor:
* const Color.fromARGB(255, 247, 189, 185), // color dấu check
* // fillColor: MaterialStateProperty.resolveWith((getColor)),
* focusColor: const Color.fromARGB(255, 154, 167, 165),
* hoverColor: const Color.fromARGB(255, 195, 201, 238),
* value: \_isChecked,
* onChanged: (bool? ts) {
* \_handleCheckbox(ts!);
* },
* ),
* Text('Checkbox set color inline - is $\_isChecked'),
* Checkbox(
* checkColor: Colors.red, // color dấu check
* fillColor: MaterialStateProperty.resolveWith<Color>(
* (Set<MaterialState> states) {
* if (states.contains(MaterialState.selected)) {
* // Màu nền khi Checkbox được chọn
* return const Color.fromARGB(
* 255, 208, 243, 33); // Thay đổi thành màu bạn muốn
* }
* // Màu nền mặc định khi Checkbox không được chọn
* return Colors.grey; // Thay đổi thành màu bạn muốn
* },
* ),
* // fillColor: MaterialStateProperty.resolveWith((getColor)),
* focusColor: const Color.fromARGB(255, 154, 167, 165),
* hoverColor: const Color.fromARGB(255, 195, 201, 238),
* value: \_isChecked,
* onChanged: (bool? ts) {
* \_handleCheckbox(ts!);
* },
* ),
* Text('Checkbox set color MaterialStateProperty - is $\_isChecked'),
* // set color với MaterialStateProperty qua hàm
* Checkbox(
* fillColor: MaterialStateProperty.resolveWith((getColorGeneral)),
* value: \_isChecked,
* onChanged: (bool? ts) {
* setState(() {
* \_isChecked = ts!;
* });
* },
* ),
* Text('Checkbox getColorGeneral $\_isChecked'),
* Checkbox(
* fillColor: MaterialStateProperty.resolveWith((getColorSeparately)),
* value: \_isChecked,
* onChanged: (bool? ts) {
* setState(() {
* \_isChecked = ts!;
* });
* },
* ),
* Text('Checkbox getColorSeparately - is $\_isChecked'),
* ],
* );
* }
* }

## CheckboxListTile

[CheckboxListTile](https://api.flutter.dev/flutter/material/CheckboxListTile/CheckboxListTile.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? value, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)?>? onChanged, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? fillColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? checkColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? hoverColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? splashRadius, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)? shape, [BorderSide](https://api.flutter.dev/flutter/painting/BorderSide-class.html)? side, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isError = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enabled, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? tileColor, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? title, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? subtitle, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isThreeLine = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? dense, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? secondary, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) selected = false, [ListTileControlAffinity](https://api.flutter.dev/flutter/material/ListTileControlAffinity.html) controlAffinity = ListTileControlAffinity.platform, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? contentPadding, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) tristate = false, [OutlinedBorder](https://api.flutter.dev/flutter/painting/OutlinedBorder-class.html)? checkboxShape, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectedTileColor, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enableFeedback, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? checkboxSemanticLabel})

Creates a combination of a list tile and a checkbox.

*const*

[CheckboxListTile.adaptive](https://api.flutter.dev/flutter/material/CheckboxListTile/CheckboxListTile.adaptive.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? value, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)?>? onChanged, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? fillColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? checkColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? hoverColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? splashRadius, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)? shape, [BorderSide](https://api.flutter.dev/flutter/painting/BorderSide-class.html)? side, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isError = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enabled, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? tileColor, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? title, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? subtitle, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isThreeLine = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? dense, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? secondary, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) selected = false, [ListTileControlAffinity](https://api.flutter.dev/flutter/material/ListTileControlAffinity.html) controlAffinity = ListTileControlAffinity.platform, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? contentPadding, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) tristate = false, [OutlinedBorder](https://api.flutter.dev/flutter/painting/OutlinedBorder-class.html)? checkboxShape, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectedTileColor, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enableFeedback, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? checkboxSemanticLabel})

Creates a combination of a list tile and a platform adaptive checkbox.

CheckboxListTile trong Flutter là một widget kết hợp giữa Checkbox và ListTile, cho phép bạn tạo một mục trong danh sách có sự kết hợp của ô chọn (checkbox) và các phần tử khác của ListTile như tiêu đề, mô tả, hình ảnh, v.v. Nó thường được sử dụng trong các danh sách tùy chọn, danh sách công việc, cài đặt ứng dụng, và nhiều ngữ cảnh khác.

CheckboxListTile có các thuộc tính và tính năng tương tự như ListTile và Checkbox, cho phép bạn tùy chỉnh tiêu đề, mô tả, hình ảnh, v.v., cùng với ô chọn (checkbox) để người dùng có thể chọn hoặc bỏ chọn mục trong danh sách.

  Widget build(BuildContext context) {

    return Column(

      children: [

        CheckboxListTile(

          tileColor: const Color.fromARGB(255, 245, 188, 184),

          title: const Text('Checkbox List Tile'),

          subtitle: const Text('Subtitle'),

          secondary: const Icon(Icons.star),

          value: \_isChecked,

          onChanged: (bool? newValue) {

            setState(() {

              \_isChecked = newValue!;

              print(\_isChecked);

            });

          },

        ),

        const Divider(),

        CheckboxListTile(

          // format checkbox

          checkColor: const Color.fromARGB(255, 44, 7, 255),

          fillColor: MaterialStateProperty.all<Color>(const Color.fromARGB(

              255, 243, 33, 33)), // Fill color when not pressed

          overlayColor: MaterialStateProperty.all<Color>(const Color.fromARGB(

              255, 154, 235, 157)), // Overlay color when pressed

          checkboxShape: RoundedRectangleBorder(

              borderRadius: BorderRadius.circular(6.0),

              side: const BorderSide(

                  width: 10, color: Color.fromARGB(255, 243, 33, 51))),

          // format ListTile

          tileColor: Color.fromARGB(255, 160, 230, 249),

          contentPadding: const EdgeInsets.all(10),

          title: const Text('Checkbox List Tile'),

          subtitle: const Text('Subtitle'),

          secondary: const Icon(Icons.star),

          value: \_isChecked,

          onChanged: (bool? newValue) {

            setState(() {

              \_isChecked = newValue!;

              print(\_isChecked);

            });

          },

        ),

      ],

    );

  }

}

//

## Radio

[Radio](https://api.flutter.dev/flutter/material/Radio/Radio.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required T value, required T? groupValue, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<T?>? onChanged, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) toggleable = false, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? fillColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? focusColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? hoverColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? splashRadius, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false})

Creates a Material Design radio button.

*const*

[Radio.adaptive](https://api.flutter.dev/flutter/material/Radio/Radio.adaptive.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required T value, required T? groupValue, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<T?>? onChanged, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) toggleable = false, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? fillColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? focusColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? hoverColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? splashRadius, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) useCupertinoCheckmarkStyle = false})

Creates an adaptive [Radio](https://api.flutter.dev/flutter/material/Radio-class.html) based on whether the target platform is iOS or macOS, following Material design's [Cross-platform guidelines](https://material.io/design/platform-guidance/cross-platform-adaptation.html).

Để sử dụng radio button, bạn cần cung cấp danh sách các tùy chọn và giá trị đang được chọn. Radio button trong danh sách sẽ chỉ có thể chọn một trong các giá trị trong danh sách.

Radio, có hai thuộc tính quan trọng là value và groupValue.

* value: Đây là giá trị duy nhất gán cho mỗi radio button. Khi radio button được chọn (click thay đổi trạng thái) thì value sẽ được gửi đến callback onChanged.
* groupValue: Đây là biến đại diện cho lựa chọn hiện tại của nhóm nút radio, biến này phải được định nghĩa trước để gán vào cho tất cả các radio button trong nhóm.

Nếu groupValue bằng với value của một nút radio nào đó, nút radio đó sẽ được chọn. (Khởi đầu thường groupValue bằng value của 1 radio button nào đó để nút đó ở trạng thái chọn).

Khi click chọn 1 radio button thì hàm callback onChanged sẽ update groupValue = value của chính nó.

Một radio button đã ở trạng thái check: click check lại sẽ ko gọi hàm callback onChanged (vô hiệu)

// Kiẻu String:  Radio<String>

class Radio1 extends StatefulWidget {

  const Radio1({super.key});

  @override

  State<Radio1> createState() => \_Radio1State();

}

class \_Radio1State extends State<Radio1> {

  // định nghĩa biến gán cho groupValue

  String selectedOption = 'Option 1';

  void \_handleRadioValueChange(String value) {

    setState(() {

      selectedOption = value;

      print(selectedOption);

    });

  }

  @override

  Widget build(BuildContext context) {

    return Column(

      mainAxisAlignment: MainAxisAlignment.center,

      children: [

        RadioListTile<String>(

          // định nghĩa kiểu String

          title: const Text('Option 1'),

          value:

              'Option 1', // value và groupValue phải cùng kiểu với kiểu String đã khai báo ở trên

          groupValue: selectedOption, //

          onChanged: (nValue) {

            \_handleRadioValueChange(nValue!);

          },

        ),

        RadioListTile<String>(

          title: const Text('Option 2'),

          value: 'Option 2',

          groupValue: selectedOption,

          onChanged: (nValue) {

            \_handleRadioValueChange(nValue!);

          },

        ),

        RadioListTile<String>(

          title: const Text('Option 3'),

          value: 'Option 3',

          groupValue: selectedOption,

          onChanged: (nValue) {

            \_handleRadioValueChange(nValue!);

          },

        ),

      ],

    );

  }

}

// Cách khác - Kiểu int

class Radio2 extends StatefulWidget {

  const Radio2({super.key});

  @override

  State<Radio2> createState() => \_Radio2State();

}

class \_Radio2State extends State<Radio2> {

  // định nghĩa biến gán cho groupValue

  int selectedOption = 1;

  void \_handleRadioValueChange(int value) {

    setState(() {

      selectedOption = value;

      print(selectedOption);

    });

  }

  @override

  Widget build(BuildContext context) {

    return Column(

      mainAxisAlignment: MainAxisAlignment.center,

      children: [

        RadioListTile<int>(

          // định nghĩa kiểu int

          title: const Text('Option 1'),

          value:

              1, // value và groupValue phải cùng kiểu với kiểu int đã khai báo ở trên

          groupValue: selectedOption, //

          onChanged: (nValue) {

            \_handleRadioValueChange(nValue!);

          },

        ),

        RadioListTile<int>(

          title: const Text('Option 2'),

          value: 2,

          groupValue: selectedOption,

          onChanged: (nValue) {

            \_handleRadioValueChange(nValue!);

          },

        ),

        RadioListTile<int>(

          title: const Text('Option 3'),

          value: 3,

          groupValue: selectedOption,

          onChanged: (nValue) {

            \_handleRadioValueChange(nValue!);

          },

        ),

      ],

    );

  }

}

// Cách khác - kiểu tự định nghĩa:  Radio<SingingCharacter>

enum SingingCharacter { lafayette, jefferson }

class RadioExample extends StatefulWidget {

  const RadioExample({super.key});

  @override

  State<RadioExample> createState() => \_RadioExampleState();

}

class \_RadioExampleState extends State<RadioExample> {

  // định nghĩa biến gán cho groupValue

  SingingCharacter? \_character = SingingCharacter.lafayette;

  @override

  Widget build(BuildContext context) {

    return Column(

      children: <Widget>[

        ListTile(

          title: const Text('Lafayette'),

          leading: Radio<SingingCharacter>(

            // định nghĩa kiểu SingingCharacter

            // value và groupValue phải cùng kiểu với kiểu SingingCharacter đã khai báo ở trên

            value: SingingCharacter.lafayette,

            groupValue: \_character,

            onChanged: (SingingCharacter? value) {

              setState(() {

                \_character = value;

                print(\_character);

              });

            },

          ),

        ),

        ListTile(

          title: const Text('Thomas Jefferson'),

          leading: Radio<SingingCharacter>(

            value: SingingCharacter.jefferson,

            groupValue: \_character,

            onChanged: (SingingCharacter? value) {

              setState(() {

                \_character = value;

                print(\_character);

              });

            },

          ),

        ),

      ],

    );

  }

}

## RadioListTile

[RadioListTile](https://api.flutter.dev/flutter/material/RadioListTile/RadioListTile.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required T value, required T? groupValue, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<T?>? onChanged, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) toggleable = false, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? fillColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? hoverColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? splashRadius, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? title, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? subtitle, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isThreeLine = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? dense, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? secondary, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) selected = false, [ListTileControlAffinity](https://api.flutter.dev/flutter/material/ListTileControlAffinity.html) controlAffinity = ListTileControlAffinity.platform, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? contentPadding, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)? shape, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? tileColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectedTileColor, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enableFeedback})

Creates a combination of a list tile and a radio button.

*const*

[RadioListTile.adaptive](https://api.flutter.dev/flutter/material/RadioListTile/RadioListTile.adaptive.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required T value, required T? groupValue, required [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<T?>? onChanged, [MouseCursor](https://api.flutter.dev/flutter/services/MouseCursor-class.html)? mouseCursor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) toggleable = false, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? activeColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? fillColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? hoverColor, [MaterialStateProperty](https://api.flutter.dev/flutter/material/MaterialStateProperty-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? overlayColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? splashRadius, [MaterialTapTargetSize](https://api.flutter.dev/flutter/material/MaterialTapTargetSize.html)? materialTapTargetSize, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? title, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? subtitle, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isThreeLine = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? dense, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? secondary, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) selected = false, [ListTileControlAffinity](https://api.flutter.dev/flutter/material/ListTileControlAffinity.html) controlAffinity = ListTileControlAffinity.platform, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? contentPadding, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)? shape, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? tileColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectedTileColor, [VisualDensity](https://api.flutter.dev/flutter/material/VisualDensity-class.html)? visualDensity, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? enableFeedback, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) useCupertinoCheckmarkStyle = false})

Creates a combination of a list tile and a platform adaptive radio.

Example 1

// kiểu  RadioListTile<Groceries>

enum Groceries {

  pickles,

  tomato,

  lettuce

} // để gán value cho các RadioListTile

Groceries? \_groceryItem = Groceries.pickles; // để gán cho groupValue

class \_RadioListTile1State extends State<RadioListTile1> {

  @override

  Widget build(BuildContext context) {

    return Column(

      children: <Widget>[

        RadioListTile<Groceries>(

          value: Groceries.pickles,

          groupValue: \_groceryItem,

          onChanged: (Groceries? value) {

            setState(() {

              \_groceryItem = value;

              debugPrint('$\_groceryItem');

            });

          },

          title: const Text('Pickles'),

          subtitle: const Text('Supporting text'),

        ),

        RadioListTile<Groceries>(

          value: Groceries.tomato,

          groupValue: \_groceryItem,

          onChanged: (Groceries? value) {

            setState(() {

              \_groceryItem = value;

              debugPrint('$\_groceryItem');

            });

          },

          title: const Text('Tomato'),

          subtitle: const Text(

              'Longer supporting text to demonstrate how the text wraps and the radio is centered vertically with the text.'),

        ),

        RadioListTile<Groceries>(

          value: Groceries.lettuce,

          groupValue: \_groceryItem,

          onChanged: (Groceries? value) {

            setState(() {

              \_groceryItem = value;

              debugPrint('$\_groceryItem');

            });

          },

          title: const Text('Lettuce'),

          subtitle: const Text(

              "Longer supporting text to demonstrate how the text wraps and how setting 'RadioListTile.isThreeLine = true' aligns the radio to the top vertically with the text."),

          isThreeLine: true,

        ),

      ],

    );

  }

}

Example 2

// kiểu RadioListTile<int>

// map từ list data cho trước

class \_RadioListTileDemoState extends State<RadioListTileDemo> {

  int selectedValue = 1;

  List<int> radioValues = [0, 1, 2];

  List<String> radioSubtitle = [

    "RadioListTile 0",

    "RadioListTile 1",

    "RadioListTile 2"

  ];

  @override

  Widget build(BuildContext context) {

    return Column(

      children: radioValues.map((value) {

        return RadioListTile<int>(

          value: value,

          groupValue: selectedValue,

          title: Text('Option $value'),

          subtitle: Text(radioSubtitle[value]),

          onChanged: (newValue) {

            setState(() {

              selectedValue = newValue!;

              debugPrint('$selectedValue');

            });

          },

        );

      }).toList(),

    );

  }

}

## TextField & TextFormField

Text fields allow users to type text into an app. They are used to build forms, send messages, create search experiences, and more. In this recipe, explore how to create and style text fields.

Flutter provides two text fields: [TextField](https://api.flutter.dev/flutter/material/TextField-class.html) and [TextFormField](https://api.flutter.dev/flutter/material/TextFormField-class.html).

[TextField](https://api.flutter.dev/flutter/material/TextField-class.html) is the most commonly used text input widget.

By default, a TextField is decorated with an underline. You can add a label, icon, inline hint text, and error text by supplying an [InputDecoration](https://api.flutter.dev/flutter/material/InputDecoration-class.html) as the [decoration](https://api.flutter.dev/flutter/material/TextField/decoration.html) property of the TextField. To remove the decoration entirely (including the underline and the space reserved for the label), set the decoration to null.

Example 1

// TextField0 - Retrieve the value of a text field

/// to retrieve the text a user has entered into a text field using the following steps:

// 1- Create a TextEditingController.

// 2- Supply the TextEditingController to a TextField

// Important: Call dispose of the TextEditingController when you’ve finished using it. This ensures that you discard any resources used by the object.

// Lifecycle

// Upon completion of editing, like pressing the "done" button on the keyboard, two actions take place:

// 1st: Editing is finalized. The default behavior of this step includes an invocation of onChanged. That default behavior can be overridden. See onEditingComplete for details.

// 2nd: onSubmitted is invoked with the user's input value.

// onSubmitted can be used to manually move focus to another input widget when a user finishes with the currently focused input widget.

class TextField0 extends StatefulWidget {

  const TextField0({super.key});

  @override

  State<TextField0> createState() => \_TextField0State();

}

class \_TextField0State extends State<TextField0> {

  final TextEditingController \_textEditingController = TextEditingController();

  String \_inputText = "";

  @override

  void dispose() {

    \_textEditingController.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return Padding(

      padding: const EdgeInsets.all(16.0),

      child: Column(

        children: <Widget>[

          TextField(

            controller: \_textEditingController,

            decoration: const InputDecoration(labelText: 'Enter text'),

            // The text field calls the onChanged callback whenever the user changes the text in the field.

            onChanged: (text) {

              setState(() {

                \_inputText = text;

              });

            },

          ),

          const SizedBox(height: 20),

          Text('You entered: $\_inputText',

              style: const TextStyle(fontSize: 18)),

          const Padding(

            padding: EdgeInsets.symmetric(horizontal: 8, vertical: 16),

            child: TextField(

              obscureText:

                  true, // Whether to hide the text being edited (e.g., for passwords).

              decoration: InputDecoration(

                border: OutlineInputBorder(),

                labelText: 'Password',

              ),

            ),

          )

        ],

      ),

    );

  }

}

Example 2

// The text field calls the onChanged callback whenever the user changes the text in the field. If the user indicates that they are done typing in the field (e.g., by pressing a button on the soft keyboard), the text field calls the onSubmitted callback.

class TextField1 extends StatefulWidget {

  const TextField1({super.key});

  @override

  State<TextField1> createState() => \_TextField1State();

}

class \_TextField1State extends State<TextField1> {

  final TextEditingController \_controller = TextEditingController();

  final TextEditingController \_controller1 = TextEditingController();

  @override

  void dispose() {

    \_controller.dispose();

    \_controller1.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Center(

        child: Column(

          children: [

            TextField(

              controller: \_controller,

              onSubmitted: (String value) async {

                await showDialog<void>(

                  context: context,

                  builder: (BuildContext context) {

                    return AlertDialog(

                      title: const Text('Thanks!'),

                      content: Text(

                          'You typed "$value", which has length ${value.characters.length}.'),

                      actions: <Widget>[

                        TextButton(

                          onPressed: () {

                            Navigator.pop(context);

                          },

                          child: const Text('OK'),

                        ),

                      ],

                    );

                  },

                );

              },

            ),

            TextField(

              controller: \_controller1,

              onEditingComplete: () async {

                await showDialog<void>(

                  context: context,

                  builder: (BuildContext context) {

                    return AlertDialog(

                      title: const Text('Thanks!'),

                      content: Text(

                          'You typed "${\_controller1.text}", which has length ${\_controller1.text.characters.length}.'),

                      actions: <Widget>[

                        TextButton(

                          onPressed: () {

                            Navigator.pop(context);

                          },

                          child: const Text('OK'),

                        ),

                      ],

                    );

                  },

                );

              },

            ),

          ],

        ),

      ),

    );

  }

}

[TextFormField](https://api.flutter.dev/flutter/material/TextFormField-class.html) wraps a TextField and integrates it with the enclosing [Form](https://api.flutter.dev/flutter/widgets/Form-class.html). This provides additional functionality, such as validation and integration with other [FormField](https://api.flutter.dev/flutter/widgets/FormField-class.html) widgets.

// Home1 - demo basic

class Home1 extends StatelessWidget {

  const Home1({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('My TextField - TextFormField'),

      ),

      body: Column(

        crossAxisAlignment: CrossAxisAlignment.start,

        children: <Widget>[

          const Padding(

            padding: EdgeInsets.symmetric(horizontal: 8, vertical: 16),

            child: TextField(

              decoration: InputDecoration(

                border: OutlineInputBorder(),

                labelText: "My TextField",

                hintText: 'Enter a search term',

              ),

            ),

          ),

          Padding(

            padding: const EdgeInsets.symmetric(horizontal: 8, vertical: 16),

            child: TextFormField(

              decoration: const InputDecoration(

                border: UnderlineInputBorder(),

                labelText: 'My TextFormField',

              ),

            ),

          ),

          const Padding(

            padding: EdgeInsets.symmetric(horizontal: 8, vertical: 16),

            child: TextField(

              obscureText:

                  true, // Whether to hide the text being edited (e.g., for passwords).

              decoration: InputDecoration(

                border: OutlineInputBorder(),

                labelText: 'Password',

              ),

            ),

          )

        ],

      ),

    );

  }

}

// Home2 - Retrieve the value of a text field

/// to retrieve the text a user has entered into a text field using the following steps:

// 1- Create a TextEditingController.

// 2- Supply the TextEditingController to a TextField

// Important: Call dispose of the TextEditingController when you’ve finished using it. This ensures that you discard any resources used by the object.

class Home2 extends StatefulWidget {

  const Home2({super.key});

  @override

  State<Home2> createState() => \_Home2State();

}

class \_Home2State extends State<Home2> {

  // Create a text controller and use it to retrieve the current value

  // of the TextField.

  final myController = TextEditingController();

  @override

  void dispose() {

    // Clean up the controller when the widget is disposed.

    myController.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Retrieve Text Input'),

      ),

      body: Padding(

        padding: const EdgeInsets.all(16),

        child: TextField(

          controller: myController,

          decoration: const InputDecoration(

            border: OutlineInputBorder(),

            labelText: 'Retrieve Text Input',

          ),

        ),

      ),

      floatingActionButton: FloatingActionButton(

        // When the user presses the button, show an alert dialog containing

        // the text that the user has entered into the text field.

        onPressed: () {

          if (myController.text.isNotEmpty) {

            showDialog(

              context: context,

              builder: (context) {

                return AlertDialog(

                  // Retrieve the text the that user has entered by using the

                  // TextEditingController.

                  content: Text(myController.text),

                );

              },

            );

          }

        },

        tooltip: 'Show value input!',

        child: const Icon(Icons.ac\_unit),

      ),

    );

  }

}

// Home3 - Handle changes to a text field

/// How do you run a callback function every time the text changes? With Flutter, you have two options:

// 1- Supply an onChanged() callback to a TextField or a TextFormField.

// 2- Use a TextEditingController.

class Home3 extends StatefulWidget {

  const Home3({super.key});

  @override

  State<Home3> createState() => \_Home3State();

}

class \_Home3State extends State<Home3> {

  // Create a text controller and use it to retrieve the current value

  // of the TextField.

  final myController = TextEditingController();

// Listen to the TextEditingController and call the \_printLatestValue() method when the text changes. Use the addListener() method for this purpose.

// Begin listening for changes when the \_Home3State class is initialized, and stop listening when the \_MyCustomFormState is disposed.

  @override

  void initState() {

    super.initState();

    // Start listening to changes.

    myController.addListener(\_printLatestValue);

  }

  @override

  void dispose() {

    // Clean up the controller when the widget is removed from the widget tree.

    // This also removes the \_printLatestValue listener.

    myController.dispose();

    super.dispose();

  }

  void \_printLatestValue() {

    final text = myController.text;

    debugPrint('Second text field: $text (${text.characters.length})');

    st2 = text;

    setState(() {}); // update UI

  }

  String st = '';

  String st2 = '';

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Retrieve Text Input'),

      ),

      body: Padding(

        padding: const EdgeInsets.all(16),

        child: Column(

          children: [

            // TextField uses onChanged

            TextField(

              decoration: const InputDecoration(

                border: OutlineInputBorder(),

                labelText: 'TextField with onChanged',

              ),

              onChanged: (text) {

                st = text;

                debugPrint(

                    'First text field: $text (${text.characters.length})');

                setState(() {}); // update UI

              },

            ),

            const SizedBox(

              height: 20,

            ),

            // TextFormField uses myController.addListener

            TextFormField(

              decoration: const InputDecoration(

                border: OutlineInputBorder(),

                labelText: 'TextFormField with myController.addListener',

              ),

              controller: myController,

            ),

            const Divider(

              height: 20,

            ),

            Text('TextField: $st'),

            Text('TextFormField: $st2'),

          ],

        ),

      ),

    );

  }

}

// Home4 - Focus and text fields (default switching focus with Tab key)

// - To give focus to a text field as soon as it’s visible, use the autofocus property.

// - Focus a text field when a button is tapped, following steps:

// 1- Create a FocusNode.

// 2- Pass the FocusNode to a TextField.

// 3- Give focus to the TextField when a button is tapped.

class Home4 extends StatefulWidget {

  const Home4({super.key});

  @override

  State<Home4> createState() => \_Home4State();

}

// Define a corresponding State class.

// This class holds data related to the form.

class \_Home4State extends State<Home4> {

  // Define the focus node. To manage the lifecycle, create the FocusNode in

  // the initState method, and clean it up in the dispose method.

  late FocusNode myFocusNode;

  @override

  void initState() {

    super.initState();

    myFocusNode = FocusNode();

  }

  @override

  void dispose() {

    // Clean up the focus node when the Form is disposed.

    myFocusNode.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Text Field Focus'),

      ),

      body: Padding(

        padding: const EdgeInsets.all(16),

        child: Column(

          children: [

            // The first text field is focused on as soon as the app starts.

            const TextField(

              autofocus: true,

            ),

            // The second text field is focused on when a user taps the

            // FloatingActionButton.

            TextField(

              focusNode: myFocusNode,

            ),

          ],

        ),

      ),

      floatingActionButton: FloatingActionButton(

        // When the button is pressed,

        // give focus to the text field using myFocusNode.

        onPressed: () => myFocusNode.requestFocus(),

        tooltip: 'Focus Second Text Field',

        child: const Icon(Icons.edit),

      ), // This trailing comma makes auto-formatting nicer for build methods.

    );

  }

}

// Home5 -  toggle the focus between the two text fields

class Home5 extends StatefulWidget {

  const Home5({super.key});

  @override

  State<Home5> createState() => \_Home5State();

}

class \_Home5State extends State<Home5> {

  // Define the focus node. To manage the lifecycle, create the FocusNode for each TextField

  late FocusNode firstFocusNode;

  late FocusNode secondFocusNode;

  @override

  void initState() {

    super.initState();

    firstFocusNode = FocusNode();

    secondFocusNode = FocusNode();

  }

  @override

  void dispose() {

    firstFocusNode.dispose();

    secondFocusNode.dispose();

    super.dispose();

  }

  bool isSecondTextFieldFocused = false;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Text Field Toggle Focus'),

      ),

      body: Padding(

        padding: const EdgeInsets.all(16),

        child: Column(

          children: [

            // The first text field is focused on as soon as the app starts.

            TextField(

              autofocus: true,

              focusNode: firstFocusNode, // set focusNode

            ),

            // The second text field is focused on when a user taps the FloatingActionButton.

            TextField(

              focusNode: secondFocusNode, // set focusNode

            ),

          ],

        ),

      ),

      floatingActionButton: FloatingActionButton(

        onPressed: () {

          setState(() {

            isSecondTextFieldFocused =

                !isSecondTextFieldFocused; // switch value of isSecondTextFieldFocused

          });

          // request focus base on isSecondTextFieldFocused

          if (isSecondTextFieldFocused) {

            FocusScope.of(context).requestFocus(secondFocusNode);

          } else {

            FocusScope.of(context).requestFocus(firstFocusNode);

          }

        },

        tooltip: 'Toggle Focus Between Text Fields',

        child: const Icon(Icons.edit),

      ),

    );

  }

}

## **showTimePicker function**

Shows a dialog containing a Material Design time picker.

The returned Future resolves to the time selected by the user when the user closes the dialog. If the user cancels the dialog, null is returned.

// Home1 - showTimePicker basic

class Home1 extends StatefulWidget {

  const Home1({super.key});

  @override

  State<Home1> createState() => \_Home1State();

}

class \_Home1State extends State<Home1> {

  TimeOfDay? selectedTime;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Show Time Picker'),

      ),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            ElevatedButton(

              onPressed: () async {

                final TimeOfDay? time = await showTimePicker(

                  context: context,

                  // helpText: 'Time of the day', // Default 'SELECT TIME' - label displayed at the top of the dialog (title)

                  confirmText: 'Get Time', // Default 'OK'

                  cancelText: 'Close', // Default 'Cancel'

                  hourLabelText:

                      'Giờ', // Default 'Hour' - in TimePickerEntryMode.input

                  minuteLabelText:

                      'Phút', // Default 'Minute' - TimePickerEntryMode.input

                  initialTime: TimeOfDay.now(),

                  // initialTime: const TimeOfDay(hour: 10, minute: 47),

                );

                setState(() {

                  // update UI with selectedTime

                  selectedTime = time;

                });

              },

              child: const Text('Show Time Picker'),

            ),

            if (selectedTime != null)

              Text('Selected time: ${selectedTime!.format(context)}'),

          ],

        ),

      ),

    );

  }

}

// Home1 - showTimePicker basic

class Home1 extends StatefulWidget {

  const Home1({super.key});

  @override

  State<Home1> createState() => \_Home1State();

}

class \_Home1State extends State<Home1> {

  TimeOfDay? selectedTime;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Show Time Picker'),

      ),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            ElevatedButton(

              onPressed: () async {

                final TimeOfDay? time = await showTimePicker(

                  context: context,

                  // helpText: 'Time of the day', // Default 'SELECT TIME' - label displayed at the top of the dialog (title)

                  confirmText: 'Get Time', // Default 'OK'

                  cancelText: 'Close', // Default 'Cancel'

                  hourLabelText:

                      'Giờ', // Default 'Hour' - in TimePickerEntryMode.input

                  minuteLabelText:

                      'Phút', // Default 'Minute' - TimePickerEntryMode.input

                  initialTime: TimeOfDay.now(),

                  // initialTime: const TimeOfDay(hour: 10, minute: 47),

                );

                setState(() {

                  // update UI with selectedTime

                  selectedTime = time;

                });

              },

              child: const Text('Show Time Picker'),

            ),

            if (selectedTime != null)

              Text('Selected time: ${selectedTime!.format(context)}'),

          ],

        ),

      ),

    );

  }

}

//--------------------------------------------------------------------

// Home1a - showTimePicker - set limit time range selected

// https://pub.dev/packages/time\_range\_picker

class Home1a extends StatefulWidget {

  const Home1a({super.key});

  @override

  State<Home1a> createState() => \_Home1aState();

}

class \_Home1aState extends State<Home1a> {

  TimeOfDay? selectedTime;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Show Time Picker 1a'),

      ),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            ElevatedButton(

              onPressed: () async {

                final TimeOfDay? time = await showTimePicker(

                  context: context,

                  initialTime: TimeOfDay.now(),

                  // initialTime: const TimeOfDay(hour: 10, minute: 47),

                  // anchorPoint: Offset.fromDirection(100), // chưa có tác dụng

                );

                setState(() {

                  // update UI with selectedTime

                  selectedTime = time;

                });

              },

              child: const Text('Show Time Picker'),

            ),

            if (selectedTime != null)

              Text('Selected time: ${selectedTime!.format(context)}'),

          ],

        ),

      ),

    );

  }

}

// Home1a - showTimePicker - set limit time range selected

// https://pub.dev/packages/time\_range\_picker

class Home1a extends StatefulWidget {

  const Home1a({super.key});

  @override

  State<Home1a> createState() => \_Home1aState();

}

class \_Home1aState extends State<Home1a> {

  TimeOfDay? selectedTime;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Show Time Picker 1a'),

      ),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            ElevatedButton(

              onPressed: () async {

                final TimeOfDay? time = await showTimePicker(

                  context: context,

                  initialTime: TimeOfDay.now(),

                  // initialTime: const TimeOfDay(hour: 10, minute: 47),

                  // anchorPoint: Offset.fromDirection(100), // chưa có tác dụng

                );

                setState(() {

                  // update UI with selectedTime

                  selectedTime = time;

                });

              },

              child: const Text('Show Time Picker'),

            ),

            if (selectedTime != null)

              Text('Selected time: ${selectedTime!.format(context)}'),

          ],

        ),

      ),

    );

  }

}

//--------------------------------------------------

// Home1A - showTimePicker with multi option

class Home1A extends StatefulWidget {

  const Home1A({super.key});

  @override

  State<Home1A> createState() => \_Home1AState();

}

class \_Home1AState extends State<Home1A> {

  ThemeMode themeMode = ThemeMode.dark;

  bool useMaterial3 = true;

  void setThemeMode(ThemeMode mode) {

    setState(() {

      themeMode = mode;

    });

  }

// switch between useMaterial3 or useMaterial2

  void setUseMaterial3(bool? value) {

    setState(() {

      useMaterial3 = value!;

    });

  }

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      theme: ThemeData.light(useMaterial3: useMaterial3),

      darkTheme: ThemeData.dark(useMaterial3: useMaterial3),

      themeMode: themeMode,

      home: TimePickerOptions(

        // TimePickerOptions defined below

        themeMode: themeMode,

        useMaterial3: useMaterial3,

        setThemeMode: setThemeMode,

        setUseMaterial3: setUseMaterial3,

      ),

    );

  }

}

class TimePickerOptions extends StatefulWidget {

  const TimePickerOptions({

    super.key,

    required this.themeMode,

    required this.useMaterial3,

    required this.setThemeMode,

    required this.setUseMaterial3,

  });

  final ThemeMode themeMode;

  final bool useMaterial3;

  final ValueChanged<ThemeMode> setThemeMode;

  final ValueChanged<bool?> setUseMaterial3;

  @override

  State<TimePickerOptions> createState() => \_TimePickerOptionsState();

}

class \_TimePickerOptionsState extends State<TimePickerOptions> {

  // Properties of showTimePicker

  // Allow the user to select, save the value for set to showTimePicker

  TimeOfDay? selectedTime;

  TimePickerEntryMode entryMode = TimePickerEntryMode.dial;

  Orientation? orientation;

  TextDirection textDirection = TextDirection.rtl;

  MaterialTapTargetSize tapTargetSize = MaterialTapTargetSize.padded;

  bool use24HourTime = false;

// switch between clock dial or text input

  void \_entryModeChanged(TimePickerEntryMode? value) {

    if (value != entryMode) {

      setState(() {

        entryMode = value!;

      });

    }

  }

// switch between portrait, landscape or from MediaQuery

  void \_orientationChanged(Orientation? value) {

    if (value != orientation) {

      setState(() {

        orientation = value;

      });

    }

  }

  void \_textDirectionChanged(TextDirection? value) {

    if (value != textDirection) {

      setState(() {

        textDirection = value!;

      });

    }

  }

  void \_tapTargetSizeChanged(MaterialTapTargetSize? value) {

    if (value != tapTargetSize) {

      setState(() {

        tapTargetSize = value!;

      });

    }

  }

// switch between 12H or 24H

  void \_use24HourTimeChanged(bool? value) {

    if (value != use24HourTime) {

      setState(() {

        use24HourTime = value!;

      });

    }

  }

  void \_themeModeChanged(ThemeMode? value) {

    widget.setThemeMode(value!);

  }

  @override

  Widget build(BuildContext context) {

    return Material(

      child: Column(

        children: <Widget>[

          Expanded(

            child: GridView(

              gridDelegate: SliverGridDelegateWithMaxCrossAxisExtent(

                maxCrossAxisExtent: 350,

                mainAxisSpacing: 4,

                mainAxisExtent: 200 \* MediaQuery.textScaleFactorOf(context),

                crossAxisSpacing: 4,

              ),

              children: <Widget>[

                EnumCard<TimePickerEntryMode>(

                  choices: TimePickerEntryMode

                      .values, // list the values of TimePickerEntryMode

                  value:

                      entryMode, // default option when loading time (defined above),

                  onChanged:

                      \_entryModeChanged, // // switch between clock dial or text input

                ),

                EnumCard<ThemeMode>(

                  choices: ThemeMode.values, // list the values of ThemeMode

                  value: widget

                      .themeMode, // default option when loading time (defined above),

                  onChanged: \_themeModeChanged,

                ),

                EnumCard<TextDirection>(

                  choices: TextDirection.values,

                  value: textDirection,

                  onChanged: \_textDirectionChanged,

                ),

                EnumCard<MaterialTapTargetSize>(

                  choices: MaterialTapTargetSize

                      .values, // list the values of MaterialTapTargetSize

                  value: tapTargetSize,

                  onChanged: \_tapTargetSizeChanged,

                ),

                ChoiceCard<Orientation?>(

                  // Orientation enum - Whether in portrait or landscape.

                  choices: const <Orientation?>[...Orientation.values, null],

                  value: orientation,

                  title: '$Orientation',

                  choiceLabels: <Orientation?, String>{

                    for (final Orientation choice in Orientation.values)

                      choice: choice.name,

                    null: 'from MediaQuery',

                  },

                  onChanged: \_orientationChanged,

                ),

                ChoiceCard<bool>(

                  choices: const <bool>[false, true],

                  value: use24HourTime,

                  onChanged: \_use24HourTimeChanged, // switch between 12H or 24H

                  title: 'Time Mode',

                  choiceLabels: const <bool, String>{

                    false: '12-hour am/pm time',

                    true: '24-hour time',

                  },

                ),

                ChoiceCard<bool>(

                  choices: const <bool>[false, true],

                  value: widget.useMaterial3,

                  onChanged: widget

                      .setUseMaterial3, // // switch between useMaterial3 or useMaterial2

                  title: 'Material Version',

                  choiceLabels: const <bool, String>{

                    false: 'Material 2',

                    true: 'Material 3',

                  },

                ),

              ],

            ),

          ),

          // Button 'Open time picker' and show selectedTime

          SingleChildScrollView(

            scrollDirection: Axis.horizontal,

            child: Row(

              mainAxisAlignment: MainAxisAlignment.center,

              children: <Widget>[

                Padding(

                  padding: const EdgeInsets.all(12.0),

                  child: ElevatedButton(

                    child: const Text('Open time picker'),

                    onPressed: () async {

                      final TimeOfDay? time = await showTimePicker(

                        // TimeOfDay? time return value selected on showTimePicker

                        context: context,

                        initialTime:

                            selectedTime ?? TimeOfDay.now(), // time initial

                        initialEntryMode:

                            entryMode, // initialEntryMode parameter can be used to determine the initial time entry selection of the picker (either a clock dial or text input).

                        orientation: orientation,

                        builder: (BuildContext context, Widget? child) {

                          // We just wrap these environmental changes around the

                          // child in this builder so that we can apply the

                          // options selected above. In regular usage, this is

                          // rarely necessary, because the default values are

                          // usually used as-is.

                          return Theme(

                            // theme of showTimePicker

                            data: Theme.of(context).copyWith(

                              materialTapTargetSize:

                                  tapTargetSize, // padding or shrinkWrap

                            ),

                            child: Directionality(

                              // Directionality - Creates a widget that determines the directionality of text and text-direction-sensitive render objects.

                              textDirection: textDirection,

                              child: MediaQuery(

                                data: MediaQuery.of(context).copyWith(

                                  alwaysUse24HourFormat:

                                      use24HourTime, // true or false

                                ),

                                child: child!,

                              ),

                            ),

                          );

                        },

                      );

                      setState(() {

                        // update UI with selectedTime

                        selectedTime = time;

                      });

                    },

                  ),

                ),

                if (selectedTime != null)

                  Text('Selected time: ${selectedTime!.format(context)}'),

              ],

            ),

          ),

        ],

      ),

    );

  }

}

// This is a simple card that presents a set of radio buttons (inside of a

// RadioSelection, defined below) for the user to select from.

// EnumCard or ChoiceCard show on screen.

// EnumCard return ChoiceCard

// RadioSelection is the child of ChoiceCard

class ChoiceCard<T extends Object?> extends StatelessWidget {

  const ChoiceCard({

    super.key,

    required this.value,

    required this.choices,

    required this.onChanged,

    required this.choiceLabels,

    required this.title,

  });

  final T value;

  final Iterable<T> choices;

  final Map<T, String> choiceLabels;

  final String title;

  final ValueChanged<T?> onChanged;

  @override

  Widget build(BuildContext context) {

    return Card(

      // If the card gets too small, let it scroll both directions.

      child: SingleChildScrollView(

        child: SingleChildScrollView(

          scrollDirection: Axis.horizontal,

          child: Padding(

            padding: const EdgeInsets.all(8.0),

            child: Column(

              crossAxisAlignment: CrossAxisAlignment.start,

              children: <Widget>[

                Padding(

                  padding: const EdgeInsets.all(8.0),

                  child: Text(title),

                ),

                for (final T choice in choices)

                  RadioSelection<T>(

                    // RadioSelection defined below

                    value: choice,

                    groupValue: value,

                    onChanged: onChanged,

                    child: Text(choiceLabels[choice]!),

                  ),

              ],

            ),

          ),

        ),

      ),

    );

  }

}

// This aggregates a ChoiceCard so that it presents a set of radio buttons for

// the allowed enum values for the user to select from.

// EnumCard return ChoiceCard (defined above)

class EnumCard<T extends Enum> extends StatelessWidget {

  const EnumCard({

    super.key,

    required this.value,

    required this.choices,

    required this.onChanged,

  });

  final T value;

  final Iterable<T> choices;

  final ValueChanged<T?> onChanged;

  @override

  Widget build(BuildContext context) {

    return ChoiceCard<T>(

        value: value,

        choices: choices,

        onChanged: onChanged,

        choiceLabels: <T, String>{

          for (final T choice in choices) choice: choice.name,

        },

        title: value.runtimeType.toString());

  }

}

// A button that has a radio button on one side and a label child. Tapping on

// the label or the radio button selects the item.

class RadioSelection<T extends Object?> extends StatefulWidget {

  const RadioSelection({

    super.key,

    required this.value,

    required this.groupValue,

    required this.onChanged,

    required this.child,

  });

  final T value;

  final T? groupValue;

  final ValueChanged<T?> onChanged;

  final Widget child;

  @override

  State<RadioSelection<T>> createState() => \_RadioSelectionState<T>();

}

class \_RadioSelectionState<T extends Object?> extends State<RadioSelection<T>> {

  @override

  Widget build(BuildContext context) {

    return Row(

      mainAxisSize: MainAxisSize.min,

      children: <Widget>[

        Padding(

          padding: const EdgeInsetsDirectional.only(end: 8),

          child: Radio<T>(

            groupValue: widget.groupValue,

            value: widget.value,

            onChanged: widget.onChanged,

          ),

        ),

        GestureDetector(

            onTap: () => widget.onChanged(widget.value), child: widget.child),

// GestureDetector: This is a widget that detects gestures, like taps, swipes, and more.

// onTap: () => widget.onChanged(widget.value): This is the callback that is executed when a tap gesture is detected on the widget wrapped by GestureDetector. It calls the onChanged callback provided as a parameter to the RadioSelection widget with the widget.value as an argument. This effectively triggers the change associated with the selected value.

// child: widget.child: This is the child widget inside the GestureDetector and typically represents the label or content that the user can tap on.

// In summary, this command allows users to select an option by tapping either the label or the radio button, and it updates the selected value based on the user's interaction.

      ],

    );

  }

}

//

## **showDatePicker function**

Shows a dialog containing a Material Design date picker.

The returned [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html) resolves to the date selected by the user when the user confirms the dialog. If the user cancels the dialog, null is returned.

// Home2 - showDatePicker - basic

class Home2 extends StatefulWidget {

  const Home2({super.key});

  @override

  State<Home2> createState() => \_Home2State();

}

class \_Home2State extends State<Home2> {

  DateTime selectedDate = DateTime.now();

  Future<void> \_selectDate(BuildContext context) async {

    final DateTime? picked = await showDatePicker(

      context: context,

      initialDate: selectedDate, // initial date

      firstDate: DateTime(2000), // min date

      lastDate: DateTime(2050), // max date

      // initialDatePickerMode: DatePickerMode.year, // set select day or year first, default is day

    );

    if (picked != selectedDate && picked != null) {

      setState(() {

        selectedDate = picked;

      });

    }

  }

  @override

  Widget build(BuildContext context) {

    // Parse the format  "yyyy-mm-dd" to "dd-mm-yyyy"

    String formattedDate = formatDate("${selectedDate.toLocal()}".split(' ')[

        0]); // "${selectedDate.toLocal()}".split(' ')[0] return "2023-10-27 14:30:00" is array 2 element, and return first element is "2023-10-27"

    return Scaffold(

      appBar: AppBar(

        title: const Text('Date Picker Example'),

      ),

      body: Center(

        child: Column(

          mainAxisSize: MainAxisSize.min,

          children: <Widget>[

            Text(

              formattedDate,

              style: const TextStyle(fontSize: 40, fontWeight: FontWeight.bold),

            ),

            const SizedBox(

              height: 20.0,

            ),

            ElevatedButton(

              onPressed: () => \_selectDate(context), // Refer step 3

              child: const Text('Select date'),

            ),

          ],

        ),

      ),

    );

  }

}

// Parse the format  "yyyy-mm-dd" to "dd-mm-yyyy"

String formatDate(String inputDate) {

  // Parse the inputDate in "yyyy-mm-dd" format to a DateTime object

  DateTime date = DateTime.parse(inputDate);

  // Format the DateTime object as a string in "dd-mm-yyyy" format

  String formattedDate =

      "${date.day.toString().padLeft(2, '0')}-${date.month.toString().padLeft(2, '0')}-${date.year.toString()}";

  return formattedDate;

}

// Home2A - showDatePicker - limit the number of days selected (include ipput mode) with firstDate and lastDate

class Home2A extends StatefulWidget {

  const Home2A({super.key});

  @override

  State<Home2A> createState() => \_Home2AState();

}

class \_Home2AState extends State<Home2A> {

  DateTime selectedDate = DateTime.now();

  Future<void> \_selectDate(BuildContext context) async {

    final DateTime currentDate = DateTime.now();

    final DateTime lastAllowedDate = currentDate.add(const Duration(

        days: 30)); // Limit selection to 30 days from the current date

    final DateTime? picked = await showDatePicker(

      context: context,

      initialDate: selectedDate, // initial date

      firstDate: currentDate, // min date

      lastDate: lastAllowedDate, // max date

    );

    if (picked != selectedDate && picked != null) {

      setState(() {

        selectedDate = picked;

      });

    }

  }

  @override

  Widget build(BuildContext context) {

    // Parse the format  "yyyy-mm-dd" to "dd-mm-yyyy"

    String formattedDate = formatDate("${selectedDate.toLocal()}".split(' ')[

        0]); // "${selectedDate.toLocal()}".split(' ')[0] return "2023-10-27 14:30:00" is array 2 element, and return first element is "2023-10-27"

    return Scaffold(

      appBar: AppBar(

        title: const Text('Date Picker Example 2A'),

      ),

      body: Center(

        child: Column(

          mainAxisSize: MainAxisSize.min,

          children: <Widget>[

            Text(

              formattedDate,

              style: const TextStyle(fontSize: 40, fontWeight: FontWeight.bold),

            ),

            const SizedBox(

              height: 20.0,

            ),

            ElevatedButton(

              onPressed: () => \_selectDate(context), // Refer step 3

              child: const Text('Select date'),

            ),

          ],

        ),

      ),

    );

  }

}

//-----------------------------------------------------------

// Home2A1 - showDatePicker - selectableDayPredicate: limit the number of days selected with set range day with condition predicates

class Home2A1 extends StatefulWidget {

  const Home2A1({super.key});

  @override

  State<Home2A1> createState() => \_Home2A1State();

}

class \_Home2A1State extends State<Home2A1> {

  DateTime selectedDate = DateTime.now();

  bool \_decideWhichDayToEnable(DateTime day) {

    // range is after now - 3 day(include now day) and before now + 10 day(not include now day)

    if ((day.isAfter(DateTime.now().subtract(const Duration(days: 3))) &&

        day.isBefore(DateTime.now().add(const Duration(days: 10))))) {

      return true;

    }

    return false;

  }

  Future<void> \_selectDate(BuildContext context) async {

    // final DateTime currentDate = DateTime.now();

    final DateTime? picked = await showDatePicker(

      context: context,

      initialDate: selectedDate, // initial date

      firstDate: DateTime(2023), // min date

      lastDate: DateTime(2024), // max date

      selectableDayPredicate: (day) => \_decideWhichDayToEnable(

          day), // set range day with condition predicates

    );

    if (picked != selectedDate && picked != null) {

      setState(() {

        selectedDate = picked;

      });

    }

  }

  @override

  Widget build(BuildContext context) {

    // Parse the format  "yyyy-mm-dd" to "dd-mm-yyyy"

    String formattedDate = formatDate("${selectedDate.toLocal()}".split(' ')[

        0]); // "${selectedDate.toLocal()}".split(' ')[0] return "2023-10-27 14:30:00" is array 2 element, and return first element is "2023-10-27"

    return Scaffold(

      appBar: AppBar(

        title: const Text('Date Picker Example 2A1 - selectableDayPredicate'),

      ),

      body: Center(

        child: Column(

          mainAxisSize: MainAxisSize.min,

          children: <Widget>[

            Text(

              formattedDate,

              style: const TextStyle(fontSize: 40, fontWeight: FontWeight.bold),

            ),

            const SizedBox(

              height: 20.0,

            ),

            ElevatedButton(

              onPressed: () => \_selectDate(context), // Refer step 3

              child: const Text('Select date'),

            ),

          ],

        ),

      ),

    );

  }

}

## **showDateRangePicker function**

Shows a full screen modal dialog containing a Material Design date range picker.

The returned [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html) resolves to the [DateTimeRange](https://api.flutter.dev/flutter/material/DateTimeRange-class.html) selected by the user when the user saves their selection. If the user cancels the dialog, null is returned.

// Home3 - To demonstrate the `showDateRangePicker` widget in Flutter, you can create a simple Flutter app that allows the user to select a date range. Here's a basic example:

class Home3 extends StatefulWidget {

  const Home3({super.key});

  @override

  State<Home3> createState() => \_Home3State();

}

class \_Home3State extends State<Home3> {

  // The range includes the [start] and [end] dates. The [start] and [end] dates may be equal to indicate a date range of a single day. The [start] date must not be after the [end] date.

  DateTimeRange? selectedDateRange; //

  Future<void> \_selectDateRange(BuildContext context) async {

    final DateTimeRange? picked = await showDateRangePicker(

      context: context,

      initialDateRange: DateTimeRange(

          start: DateTime.now(),

          end: DateTime.now().add(const Duration(days: 7))),

      firstDate: DateTime.now(),

      lastDate: DateTime(2024),

    );

    if (picked != null) {

      setState(() {

        selectedDateRange = picked;

      });

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Date Range Picker Example'),

      ),

      body: Center(

        child: Column(

          mainAxisSize: MainAxisSize.min,

          children: <Widget>[

            Text(

              selectedDateRange != null

                  ? 'Selected Range: ${selectedDateRange!.start} - ${selectedDateRange!.end}'

                  : 'No range selected',

              style: const TextStyle(fontSize: 20),

            ),

            const SizedBox(height: 20.0),

            ElevatedButton(

              onPressed: () => \_selectDateRange(context),

              child: const Text('Select Date Range'),

            ),

          ],

        ),

      ),

    );

  }

}

## **Time Range Picker**

## <https://pub.dev/packages/time_range_picker>

import 'package:flutter/foundation.dart';

import 'package:flutter/material.dart';

import 'package:time\_range\_picker/time\_range\_picker.dart';

import 'package:flutter/cupertino.dart';

class MyTimeRangePicker extends StatelessWidget {

  const MyTimeRangePicker({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      theme: ThemeData(

        primarySwatch: Colors.blue,

      ),

      home: const TimeRangePicker2A(),

    );

  }

}

/// https://pub.dev/packages/time\_range\_picker

/// Xem diến giải các tham số tại link trang chủ trên luôn

/// Demo 01

class TimeRangePicker1 extends StatelessWidget {

  const TimeRangePicker1({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        backgroundColor: Colors.blueGrey,

        body: Center(

          child: ElevatedButton(

            onPressed: () async {

              TimeRange? result = await showTimeRangePicker(

                context: context,

              );

              // Kiểm tra xem người dùng đã chọn thời gian hay không

              if (result != null) {

                debugPrint("Selected time range: $result");

              } else {

                // Người dùng đã nhấn nút "Close" hoặc hủy bỏ, không chọn thời gian nào

                debugPrint("No time range selected");

              }

            },

            child: const Text("Open Time Range Picker"),

          ),

        ));

  }

}

/// Demo 02

class TimeRangePicker2 extends StatefulWidget {

  const TimeRangePicker2({super.key});

  @override

  State<TimeRangePicker2> createState() => \_TimeRangePicker2State();

}

class \_TimeRangePicker2State extends State<TimeRangePicker2> {

  TimeRange? result;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        backgroundColor: Colors.blueGrey,

        body: Center(

            child:

                Column(mainAxisAlignment: MainAxisAlignment.center, children: [

          ElevatedButton(

            onPressed: () async {

              TimeRange? kq = await showTimeRangePicker(

                start: const TimeOfDay(hour: 8, minute: 0),

                end: const TimeOfDay(hour: 17, minute: 0),

                // disabledTime ko cho phép chọn TimeRange đi xuyên qua nó. Vd: disabledTime từ 12h đến 13h thì ko thể chọn TimeRange với startTime trước 12h và endTime sau 13h

                disabledTime: TimeRange(

                    startTime: const TimeOfDay(hour: 12, minute: 0),

                    endTime: const TimeOfDay(hour: 13, minute: 0)),

                context: context,

              );

              // Kiểm tra xem người dùng đã chọn thời gian hay không

              if (kq != null) {

                setState(() {

                  result = kq;

                });

              }

            },

            child: const Text("Open Time Range Picker"),

          ),

          const SizedBox(height: 20.0),

          Text(

            result != null ? 'Selected Range: $result' : 'No range selected',

            style: const TextStyle(fontSize: 20),

          ),

        ])));

  }

}

/// Demo 2A - Set ticks: 12 & labels xoay theo độ clockRotation(labels sẽ luôn phù hợp với mốc 0h)

class TimeRangePicker2A extends StatefulWidget {

  const TimeRangePicker2A({super.key});

  @override

  State<TimeRangePicker2A> createState() => \_TimeRangePicker2AState();

}

class \_TimeRangePicker2AState extends State<TimeRangePicker2A> {

  TimeRange? result;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        backgroundColor: Colors.blueGrey,

        body: Center(

            child:

                Column(mainAxisAlignment: MainAxisAlignment.center, children: [

          ElevatedButton(

            onPressed: () async {

              TimeRange? kq = await showTimeRangePicker(

                start: const TimeOfDay(hour: 8, minute: 0),

                end: const TimeOfDay(hour: 17, minute: 0),

                // disabledTime ko cho phép chọn TimeRange đi xuyên qua nó. Vd: disabledTime từ 12h đến 13h thì ko thể chọn TimeRange với startTime trước 12h và endTime sau 13h

                // disabledTime: TimeRange(

                //     startTime: const TimeOfDay(hour: 12, minute: 0),

                //     endTime: const TimeOfDay(hour: 13, minute: 0)),

                // clockRotation:0(default) - labels list sẽ map với ticks: labels list từ trên xuống, ticks 0h đầu tiên tại đáy đồng hồ.

                // clockRotation:180 - ticks 0h đầu tiên xoay lên đỉnh đồng hồ

                clockRotation: 180,

                ticks: 12,

                ticksColor: Colors.green,

                ticksLength: 20,

                ticksOffset: 5,

                labels: [

                  ClockLabel.fromTime(

                      time: const TimeOfDay(hour: 0, minute: 0), text: "0h"),

                  ClockLabel.fromTime(

                      time: const TimeOfDay(hour: 3, minute: 0), text: "3h"),

                  ClockLabel.fromTime(

                      time: const TimeOfDay(hour: 6, minute: 0), text: "6h"),

                  ClockLabel.fromTime(

                      time: const TimeOfDay(hour: 9, minute: 0), text: "9h"),

                  ClockLabel.fromTime(

                      time: const TimeOfDay(hour: 12, minute: 0), text: "12h"),

                  ClockLabel.fromTime(

                      time: const TimeOfDay(hour: 15, minute: 0), text: "15h"),

                  ClockLabel.fromTime(

                      time: const TimeOfDay(hour: 18, minute: 0), text: "18h"),

                  ClockLabel.fromTime(

                      time: const TimeOfDay(hour: 21, minute: 0), text: "21h"),

                  ClockLabel.fromTime(

                      time: const TimeOfDay(hour: 24, minute: 0), text: "0h")

                ],

                context: context,

              );

              // Kiểm tra xem người dùng đã chọn thời gian hay không

              if (kq != null) {

                setState(() {

                  result = kq;

                });

              }

            },

            child: const Text("Open Time Range Picker"),

          ),

          const SizedBox(height: 20.0),

          Text(

            result != null ? 'Selected Range: $result' : 'No range selected',

            style: const TextStyle(fontSize: 20),

          ),

        ])));

  }

}

// Demo 03

class TimeRangePicker3 extends StatefulWidget {

  const TimeRangePicker3({super.key, required this.title});

  // This widget is the home page of your application. It is stateful, meaning

  // that it has a State object (defined below) that contains fields that affect

  // how it looks.

  // This class is the configuration for the state. It holds the values (in this

  // case the title) provided by the parent (in this case the App widget) and

  // used by the build method of the State. Fields in a Widget subclass are

  // always marked "final".

  final String title;

  @override

  State<TimeRangePicker3> createState() => \_TimeRangePicker3State();

}

class \_TimeRangePicker3State extends State<TimeRangePicker3> {

  TimeOfDay \_startTime = TimeOfDay.now();

  TimeOfDay \_endTime =

      TimeOfDay.fromDateTime(DateTime.now().add(const Duration(hours: 3)));

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(widget.title),

      ),

      body: ListView(children: [

        ElevatedButton(

          onPressed: () async {

            TimeRange? result = await showTimeRangePicker(

              context: context,

            );

            if (kDebugMode) {

              print("result $result");

            }

          },

          child: const Text("Pure"),

        ),

        ElevatedButton(

          onPressed: () {

            showTimeRangePicker(

              context: context,

              start: const TimeOfDay(hour: 22, minute: 9),

              onStartChange: (start) {

                if (kDebugMode) {

                  print("start time $start");

                }

              },

              onEndChange: (end) {

                if (kDebugMode) {

                  print("end time $end");

                }

              },

              interval: const Duration(hours: 1),

              minDuration: const Duration(hours: 1),

              use24HourFormat: false,

              padding: 30,

              strokeWidth: 20, // độ dày viền clock

              handlerRadius: 10, // bán kính icon change value

              strokeColor: const Color.fromARGB(

                  255, 140, 0, 255), // color of range time selection

              handlerColor: Colors.orange[700], // color of icon handler

              selectedColor: const Color.fromARGB(

                  255, 255, 7, 69), // color of icon handler selected

              backgroundColor: Colors.black.withOpacity(0.3),

              ticks: 12,

              ticksColor: Colors.white,

              snap: true,

              labels: [

                "12 am",

                "3 am",

                "6 am",

                "9 am",

                "12 pm",

                "3 pm",

                "6 pm",

                "9 pm"

              ].asMap().entries.map((e) {

                return ClockLabel.fromIndex(

                    idx: e.key, length: 8, text: e.value);

              }).toList(),

              labelOffset: -30,

              labelStyle: const TextStyle(

                  fontSize: 22,

                  color: Colors.grey,

                  fontWeight: FontWeight.bold),

              timeTextStyle: const TextStyle(

                  // timeTextStyle - text [\_startTime] and [\_endTime]

                  color: Colors.orange,

                  fontSize: 24,

                  fontStyle: FontStyle.italic,

                  fontWeight: FontWeight.bold),

              activeTimeTextStyle: const TextStyle(

                  // activeTimeTextStyle - text [\_startTime] and [\_endTime] nhưng đang được change value

                  color: Color.fromARGB(255, 47, 255, 0),

                  fontSize: 26,

                  fontStyle: FontStyle.italic,

                  fontWeight: FontWeight.bold),

            );

          },

          child: const Text("Interval"),

        ),

        ElevatedButton(

          onPressed: () async {

            TimeRange? result = await showTimeRangePicker(

                context: context,

                start: const TimeOfDay(hour: 9, minute: 0),

                end: const TimeOfDay(hour: 12, minute: 0),

                disabledTime: TimeRange(

                    startTime: const TimeOfDay(hour: 22, minute: 0),

                    endTime: const TimeOfDay(hour: 5, minute: 0)),

                disabledColor: Colors.red.withOpacity(0.5),

                strokeWidth: 4,

                ticks: 24,

                ticksOffset: -7,

                ticksLength: 15,

                ticksColor: Colors.grey,

                labels: [

                  "12 am",

                  "3 am",

                  "6 am",

                  "9 am",

                  "12 pm",

                  "3 pm",

                  "6 pm",

                  "9 pm"

                ].asMap().entries.map((e) {

                  return ClockLabel.fromIndex(

                      idx: e.key, length: 8, text: e.value);

                }).toList(),

                labelOffset: 35,

                rotateLabels:

                    false, // rotateLabels: true - label rotate theo đường kính clock; false - label luôn hiển thị ngang

                padding: 60);

            if (kDebugMode) {

              print("result $result");

            }

          },

          child: const Text("Disabled Times"),

        ),

        ElevatedButton(

          onPressed: () async {

            TimeRange? result = await showTimeRangePicker(

              context: context,

              paintingStyle: PaintingStyle.fill,

              backgroundColor: Colors.grey.withOpacity(0.2),

              labels: [

                ClockLabel.fromTime(

                    time: const TimeOfDay(hour: 7, minute: 0),

                    text: "Start Work"),

                ClockLabel.fromTime(

                    time: const TimeOfDay(hour: 18, minute: 0), text: "Go Home")

              ],

              start: const TimeOfDay(hour: 10, minute: 0),

              end: const TimeOfDay(hour: 13, minute: 0),

              ticks: 8,

              strokeColor: Theme.of(context)

                  .primaryColor

                  .withOpacity(0.5), // color of khu vực chọn

              ticksColor: Theme.of(context).primaryColor,

              labelOffset: 15,

              padding: 60,

              disabledTime: TimeRange(

                  startTime: const TimeOfDay(hour: 18, minute: 0),

                  endTime: const TimeOfDay(hour: 7, minute: 0)),

              disabledColor: Colors.red.withOpacity(0.5),

            );

            if (kDebugMode) {

              print("result $result");

            }

          },

          child: const Text("Filled Style"),

        ),

        ElevatedButton(

          onPressed: () async {

            TimeRange? result = await showTimeRangePicker(

              context: context,

              strokeColor: Colors.teal,

              handlerColor: Colors.teal[200],

              selectedColor: Colors.tealAccent,

              strokeWidth: 16,

              handlerRadius: 18,

              backgroundWidget: Image.asset(

                // set background image

                "assets/images/day-night.png",

                height: 300,

                width: 300,

              ),

              labels: [

                ClockLabel.fromTime(

                    time: const TimeOfDay(hour: 6, minute: 0), text: "Get up"),

                ClockLabel.fromTime(

                    time: const TimeOfDay(hour: 9, minute: 0),

                    text: "Coffee time"),

                ClockLabel.fromTime(

                    time: const TimeOfDay(hour: 15, minute: 0),

                    text: "Afternoon"),

                ClockLabel.fromTime(

                    time: const TimeOfDay(hour: 18, minute: 0),

                    text: "Time for a beer"),

                ClockLabel.fromTime(

                    time: const TimeOfDay(hour: 22, minute: 0),

                    text: "Go to Sleep"),

                ClockLabel.fromTime(

                    time: const TimeOfDay(hour: 2, minute: 0),

                    text: "Go for a pee"),

                ClockLabel.fromTime(

                    time: const TimeOfDay(hour: 12, minute: 0),

                    text: "Lunchtime!")

              ],

              ticks: 12,

              ticksColor: Colors.black,

              labelOffset: 40,

              padding: 55,

              labelStyle: const TextStyle(fontSize: 18, color: Colors.black),

            );

            if (kDebugMode) {

              print("result $result");

            }

          },

          child: const Text("Background Widget"),

        ),

        ElevatedButton(

          onPressed: () async {

            TimeRange? result = await showTimeRangePicker(

              context: context,

              strokeWidth: 4,

              ticks: 24,

              ticksOffset: 2,

              ticksLength: 8,

              handlerRadius: 8,

              ticksColor: Colors.grey,

              rotateLabels: false,

              labels: [

                "24 h",

                "3 h",

                "6 h",

                "9 h",

                "12 h",

                "15 h",

                "18 h",

                "21 h"

              ].asMap().entries.map((e) {

                return ClockLabel.fromIndex(

                    idx: e.key, length: 8, text: e.value);

              }).toList(),

              labelOffset: 30,

              padding: 55,

              labelStyle: const TextStyle(fontSize: 18, color: Colors.black),

              start: const TimeOfDay(hour: 12, minute: 0),

              end: const TimeOfDay(hour: 15, minute: 0),

              disabledTime: TimeRange(

                  startTime: const TimeOfDay(hour: 6, minute: 0),

                  endTime: const TimeOfDay(hour: 10, minute: 0)),

              // clockRotation:0(default) - labels list sẽ map với ticks: labels list từ trên xuống, ticks 0h đầu tiên tại đáy đồng hồ.

              // clockRotation:180 - ticks 0h đầu tiên xoay lên đỉnh đồng hồ

              clockRotation: 0.0,

            );

            if (kDebugMode) {

              print("result $result");

            }

          },

          child: const Text("Rotated Clock"),

        ),

        ElevatedButton(

          onPressed: () async {

            TimeRange? result = await showTimeRangePicker(

              context: context,

              rotateLabels: false,

              ticks: 12,

              ticksColor: Colors.grey,

              ticksOffset: -12,

              labels: [

                "24 h",

                "3 h",

                "6 h",

                "9 h",

                "12 h",

                "15 h",

                "18 h",

                "21 h"

              ].asMap().entries.map((e) {

                return ClockLabel.fromIndex(

                    idx: e.key, length: 8, text: e.value);

              }).toList(),

              labelOffset: -30,

              padding: 55,

              start: const TimeOfDay(hour: 12, minute: 0),

              end: const TimeOfDay(hour: 18, minute: 0),

              disabledTime: TimeRange(

                startTime: const TimeOfDay(hour: 4, minute: 0),

                endTime: const TimeOfDay(hour: 10, minute: 0),

              ),

              maxDuration: const Duration(hours: 6),

            );

            if (kDebugMode) {

              print("result $result");

            }

          },

          child: const Text("Max duration"),

        ),

        ElevatedButton(

          onPressed: () async {

            TimeRange? result = await showTimeRangePicker(

              context: context,

              rotateLabels: false,

              ticks: 12,

              ticksColor: Colors.grey,

              ticksOffset: -12,

              labels: [

                "24 h",

                "3 h",

                "6 h",

                "9 h",

                "12 h",

                "15 h",

                "18 h",

                "21 h"

              ].asMap().entries.map((e) {

                return ClockLabel.fromIndex(

                    idx: e.key, length: 8, text: e.value);

              }).toList(),

              labelOffset: -30,

              padding: 55,

              start: const TimeOfDay(hour: 12, minute: 0),

              end: const TimeOfDay(hour: 18, minute: 0),

              disabledTime: TimeRange(

                startTime: const TimeOfDay(hour: 4, minute: 0),

                endTime: const TimeOfDay(hour: 10, minute: 0),

              ),

              minDuration: const Duration(hours: 3),

            );

            if (kDebugMode) {

              print("result $result");

            }

          },

          child: const Text("Min duration"),

        ),

        ElevatedButton(

          onPressed: () async {

            TimeRange? result = await showTimeRangePicker(

                context: context, barrierDismissible: false);

            if (kDebugMode) {

              print("result $result");

            }

          },

          child: const Text("No barrier dismissable"),

        ),

        const Divider(),

        Text(

          'As a regular widget:',

          style: Theme.of(context).textTheme.titleLarge,

          textAlign: TextAlign.center,

        ),

        const SizedBox(

          height: 20,

        ),

        Row(

          mainAxisAlignment: MainAxisAlignment.spaceAround,

          children: [

            Text("Start: ${\_startTime.format(context)}"),

            Text("End: ${\_endTime.format(context)}"),

          ],

        ),

        SizedBox(

          height: 300,

          child: TimeRangePicker(

            hideButtons: true, //  show/hide 2 buttons Cancel and OK

            hideTimes:

                true, // show/hide time texts From [\_startTime] and To [\_endTime]

            paintingStyle: PaintingStyle.fill, // fill time range selected

            backgroundColor: Colors.grey.withOpacity(0.2),

            labels: [

              // set labels for time ranges

              ClockLabel.fromTime(

                  time: const TimeOfDay(hour: 7, minute: 0),

                  text: "Start Work"),

              ClockLabel.fromTime(

                  time: const TimeOfDay(hour: 18, minute: 0), text: "Go Home")

            ],

            start: \_startTime,

            end: \_endTime,

            ticks: 12, // vạch khoảng cách cho clock

            ticksOffset: -5, // position of the ticks within đường kính clock

            strokeColor: Theme.of(context).primaryColor.withOpacity(0.5),

            ticksColor: Theme.of(context).primaryColorDark,

            labelOffset: 20, // spacing giữa label and đường kính clock

            padding: 30,

            onStartChange: (start) {

              // event khi change start value

              setState(() {

                \_startTime = start;

              });

            },

            onEndChange: (end) {

              // event khi change end value

              setState(() {

                \_endTime = end;

              });

            },

          ),

        ),

        ElevatedButton(

          onPressed: () async {

            TimeRange? result = await showDialog(

              context: context,

              builder: (BuildContext context) {

                TimeOfDay startTime = TimeOfDay.now();

                TimeOfDay endTime = TimeOfDay.now();

                return AlertDialog(

                  contentPadding: EdgeInsets.zero,

                  title: const Text("Choose a nice timeframe"),

                  content: SizedBox(

                    width: MediaQuery.of(context).size.width,

                    height: 450,

                    child: TimeRangePicker(

                      hideButtons: true,

                      onStartChange: (start) {

                        setState(() {

                          startTime = start;

                        });

                      },

                      onEndChange: (end) {

                        setState(() {

                          endTime = end;

                        });

                      },

                    ),

                  ),

                  actions: <Widget>[

                    TextButton(

                        child: const Text('My custom cancel'),

                        onPressed: () {

                          Navigator.of(context).pop();

                        }),

                    TextButton(

                      child: const Text('My custom ok'),

                      onPressed: () {

                        Navigator.of(context).pop(

                            TimeRange(startTime: startTime, endTime: endTime));

                      },

                    ),

                  ],

                );

              },

            );

            if (kDebugMode) {

              print(result.toString());

            }

          },

          child: const Text("Custom Dialog"),

        ),

        ElevatedButton(

            onPressed: () async {

              TimeRange? result = await showCupertinoDialog(

                barrierDismissible: true,

                context: context,

                builder: (BuildContext context) {

                  TimeOfDay startTime = TimeOfDay.now();

                  TimeOfDay endTime = TimeOfDay.now();

                  return CupertinoAlertDialog(

                    content: SizedBox(

                        width: MediaQuery.of(context).size.width,

                        height: 340,

                        child: Column(

                          children: [

                            TimeRangePicker(

                              padding: 22,

                              hideButtons: true,

                              handlerRadius: 8,

                              strokeWidth: 4,

                              ticks: 12,

                              activeTimeTextStyle: const TextStyle(

                                  fontWeight: FontWeight.normal,

                                  fontSize: 22,

                                  color: Colors.white),

                              timeTextStyle: const TextStyle(

                                  fontWeight: FontWeight.normal,

                                  fontSize: 22,

                                  color: Colors.white70),

                              onStartChange: (start) {

                                setState(() {

                                  startTime = start;

                                });

                              },

                              onEndChange: (end) {

                                setState(() {

                                  endTime = end;

                                });

                              },

                            ),

                          ],

                        )),

                    actions: <Widget>[

                      CupertinoDialogAction(

                          isDestructiveAction: true,

                          child: const Text('Cancel'),

                          onPressed: () {

                            Navigator.of(context).pop();

                          }),

                      CupertinoDialogAction(

                        child: const Text('Ok'),

                        onPressed: () {

                          Navigator.of(context).pop(

                            TimeRange(startTime: startTime, endTime: endTime),

                          );

                        },

                      ),

                    ],

                  );

                },

              );

              if (kDebugMode) {

                print(result.toString());

              }

            },

            child: const Text("Cupertino style"))

      ]),

    );

  }

}

## Progress bar

Progress bar(Thanh tiến trình) là một phần tử điều khiển đồ họa được sử dụng để hiển thị tiến trình của một tác vụ như tải xuống, tải lên, cài đặt, truyền tệp, v.v. Trong phần này, chúng ta sẽ hiểu cách hiển thị progress bar trong một ứng dụng rung.

Flutter có thể hiển thị progress bar với sự trợ giúp của hai widget con, được đưa ra dưới đây:

### LinearProgressIndicator

[LinearProgressIndicator](https://api.flutter.dev/flutter/material/LinearProgressIndicator/LinearProgressIndicator.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? value, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [Animation](https://api.flutter.dev/flutter/animation/Animation-class.html)<[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?>? valueColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? minHeight, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticsLabel, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticsValue, [BorderRadiusGeometry](https://api.flutter.dev/flutter/painting/BorderRadiusGeometry-class.html) borderRadius = BorderRadius.zero})

Creates a linear progress indicator.

// Example 1 - LinearProgressIndicator not status %

class LinearProgress01 extends StatefulWidget {

  const LinearProgress01({super.key});

  @override

  State<StatefulWidget> createState() {

    return LinearProgressIndicatorAppState();

  }

}

class LinearProgressIndicatorAppState extends State<LinearProgress01> {

  bool \_loading = false; // flag to switch to loading state

  @override

  void initState() {

    super.initState();

    \_loading = false;

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text("Flutter Linear Progress Bar"),

      ),

      body: Center(

        child: Container(

          padding: const EdgeInsets.all(12.0),

          child: \_loading

              ? const LinearProgressIndicator()

              : const Text("Press button for downloading",

                  style: TextStyle(fontSize: 25)),

        ),

      ),

      floatingActionButton: FloatingActionButton(

        onPressed: () {

          setState(() {

            \_loading = !\_loading;

          });

        },

        tooltip: 'Download',

        child: const Icon(Icons.cloud\_download),

      ),

    );

  }

}

// Example 2 -  LinearProgressIndicator with status %

class LinearProgress02 extends StatefulWidget {

  const LinearProgress02({super.key});

  @override

  State<StatefulWidget> createState() {

    return LinearProgressIndicatorAppState2();

  }

}

class LinearProgressIndicatorAppState2 extends State<LinearProgress02> {

  bool \_loading = false;

  double \_progressValue = 0;

  @override

  void initState() {

    super.initState();

    \_loading = false;

    \_progressValue = 0.0;

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text("Flutter Linear Progress Bar With Progress Value"),

      ),

      body: Center(

        child: Container(

          padding: const EdgeInsets.all(12.0),

          child: \_loading

              ? Column(

                  mainAxisAlignment: MainAxisAlignment.center,

                  children: <Widget>[

                    LinearProgressIndicator(

                      backgroundColor: Colors.cyanAccent,

                      valueColor:

                          const AlwaysStoppedAnimation<Color>(Colors.red),

                      value: \_progressValue,

                    ),

                    Text('${(\_progressValue \* 100).round()}%'),

                  ],

                )

              : const Text("Press button for downloading",

                  style: TextStyle(fontSize: 25)),

        ),

      ),

      floatingActionButton: FloatingActionButton(

        onPressed: () {

          setState(() {

            \_loading = !\_loading;

            \_updateProgress();

          });

        },

        tooltip: 'Download',

        child: const Icon(Icons.cloud\_download),

      ),

    );

  }

  // this function updates the progress value

  void \_updateProgress() {

    const oneSec = Duration(milliseconds: 300);

    Timer.periodic(oneSec, (Timer t) {

      // import 'dart:async'

      setState(() {

        \_progressValue += 0.1;

        // we "finish" downloading here

        if (\_progressValue.toStringAsFixed(1) == '1.0') {

          \_loading = false;

          \_progressValue = 0;

          t.cancel();

          return;

        }

      });

    });

  }

}

// Example 3 -  click button to stop progress indicator

class LinearProgress03 extends StatefulWidget {

  const LinearProgress03({super.key});

  @override

  State<LinearProgress03> createState() => \_LinearProgress03State();

}

class \_LinearProgress03State extends State<LinearProgress03> {

  bool \_isPaused = false;

  double \_progressValue = 0.0;

  /// \_updateProgress là một hàm đệ quy (gọi chính nó) để liên tục cập nhật giá trị tiến trình với khoảng thời gian nhất định, và việc này được thực hiện trong một vòng lặp không dừng nếu \_isPaused không được đặt là true.

//  const duration = Duration(milliseconds: 500);:

// Đây là một hằng số duration được sử dụng để xác định khoảng thời gian giữa các bước tiến trình.

// Trong ví dụ này, khoảng thời gian là 500 milliseconds (0.5 giây).

// Future.delayed(duration, \_updateProgress);:

// Future.delayed được sử dụng để tạo một Future sẽ hoàn thành sau một khoảng thời gian nhất định (duration).

// Sau khi thời gian đó kết thúc, \_updateProgress sẽ được gọi lại để cập nhật tiến trình.

// Điều này tạo ra một loạt các bước cập nhật tiến trình với khoảng thời gian nhất định giữa chúng.

  void \_updateProgress() {

    const duration = Duration(milliseconds: 500);

    Future.delayed(duration, \_updateProgress);

    setState(() {

      if (!\_isPaused) {

        \_progressValue += 0.2;

        if (\_progressValue > 1.0) {

          \_progressValue = 0.0;

        }

      }

    });

  }

  @override

  void initState() {

    super.initState();

    \_updateProgress();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Progress Indicator Demo'),

      ),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            LinearProgressIndicator(

              value: \_progressValue,

              backgroundColor: const Color.fromARGB(255, 231, 251, 185),

              valueColor: const AlwaysStoppedAnimation<Color>(

                  Color.fromARGB(255, 15, 5, 206)),

            ),

            const SizedBox(height: 20),

            ElevatedButton(

              onPressed: () {

                setState(() {

                  \_isPaused = !\_isPaused;

                });

              },

              child: Text(\_isPaused ? 'Resume' : 'Pause'),

            ),

          ],

        ),

      ),

    );

  }

}

// Example 4 -   The progress indicator is animated using an AnimationController, click Switch to stop controller

class LinearProgress04 extends StatefulWidget {

  const LinearProgress04({super.key});

  @override

  State<LinearProgress04> createState() => \_ProgressIndicatorExampleState();

}

/// TickerProviderStateMixin is a mixin class in Flutter that provides a vsync (Vertical Synchronization) property to a class. This mixin is commonly used with animation-related classes that require a vsync parameter, such as AnimationController.

///In Flutter, animations are typically synchronized with the device's vertical refresh rate to ensure smooth and visually pleasing animations. The vsync parameter, which stands for vertical synchronization, is used to achieve this synchronization.

class \_ProgressIndicatorExampleState extends State<LinearProgress04>

    with TickerProviderStateMixin {

  late AnimationController controller;

  bool determinate = false;

  @override

  void initState() {

    controller = AnimationController(

      /// [AnimationController]s can be created with `vsync: this` because of

      /// [TickerProviderStateMixin].

      vsync: this,

      duration: const Duration(seconds: 2),

    )..addListener(() {

        setState(() {});

      });

    controller.repeat();

    super.initState();

  }

  @override

  //  disposes of the AnimationController.

  void dispose() {

    controller.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Padding(

        padding: const EdgeInsets.all(20.0),

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: <Widget>[

            const Text(

              'Linear progress indicator',

              style: TextStyle(fontSize: 20),

            ),

            const SizedBox(height: 30),

            LinearProgressIndicator(

              value: controller.value,

              semanticsLabel: 'Linear progress indicator',

            ),

            const SizedBox(height: 10),

            Row(

              children: <Widget>[

                Expanded(

                  child: Text(

                    'determinate Mode',

                    style: Theme.of(context).textTheme.titleSmall,

                  ),

                ),

                Switch(

                  value: determinate,

                  onChanged: (bool value) {

                    setState(() {

                      determinate = value;

                      if (determinate) {

                        controller.stop();

                      } else {

                        controller

                          ..forward(from: controller.value)

                          ..repeat();

                      }

                    });

                  },

                ),

              ],

            ),

          ],

        ),

      ),

    );

  }

}

### CircularProgressIndicator

### RefreshIndicator

## Animations

## Form

[Form](https://api.flutter.dev/flutter/widgets/Form/Form.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child, [WillPopCallback](https://api.flutter.dev/flutter/widgets/WillPopCallback.html)? onWillPop, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onChanged, [AutovalidateMode](https://api.flutter.dev/flutter/widgets/AutovalidateMode.html)? autovalidateMode})

Creates a container for form fields.

*const*

An optional container for grouping together multiple form field widgets (e.g. [TextField](https://api.flutter.dev/flutter/material/TextField-class.html) widgets).

Each individual form field should be wrapped in a [FormField](https://api.flutter.dev/flutter/widgets/FormField-class.html) widget, with the [Form](https://api.flutter.dev/flutter/widgets/Form-class.html) widget as a common ancestor of all of those. Call methods on [FormState](https://api.flutter.dev/flutter/widgets/FormState-class.html) to save, reset, or validate each [FormField](https://api.flutter.dev/flutter/widgets/FormField-class.html) that is a descendant of this [Form](https://api.flutter.dev/flutter/widgets/Form-class.html). To obtain the [FormState](https://api.flutter.dev/flutter/widgets/FormState-class.html), you may use [Form.of](https://api.flutter.dev/flutter/widgets/Form/of.html) with a context whose ancestor is the [Form](https://api.flutter.dev/flutter/widgets/Form-class.html), or pass a [GlobalKey](https://api.flutter.dev/flutter/widgets/GlobalKey-class.html) to the [Form](https://api.flutter.dev/flutter/widgets/Form-class.html) constructor and call [GlobalKey.currentState](https://api.flutter.dev/flutter/widgets/GlobalKey/currentState.html).

Some important notes

### Key property

Là là một GlobalKey<FormState> , có vai trò quan trọng trong việc xác định và điều khiển trạng thái của biểu mẫu và các trường nhập liệu trong đó.

Dưới đây là một số lý do khiến key là bắt buộc cho Form:

* Quản lý trạng thái của biểu mẫu: Form phải theo dõi trạng thái của các trường nhập liệu bên trong nó, chẳng hạn như kiểm tra xem các trường nhập liệu có hợp lệ không. key được sử dụng để quản lý và truy cập trạng thái này.
* Gửi dữ liệu từ biểu mẫu: Khi người dùng gửi biểu mẫu, bạn cần sử dụng key để thu thập thông tin từ các trường nhập liệu và xử lý dữ liệu.
* Xác thực biểu mẫu: key cho phép bạn xác thực tính hợp lệ của biểu mẫu bằng cách gọi phương thức validate() trên đối tượng FormState. Điều này giúp bạn kiểm tra xem tất cả các trường nhập liệu đã được điền đầy đủ và đúng cách.
* Lưu trạng thái trường nhập liệu: key cũng được sử dụng để gọi phương thức save() trên đối tượng FormState, cho phép bạn lưu trạng thái của các trường nhập liệu sau khi biểu mẫu được gửi đi.
* Kiểm soát biểu mẫu từ bên ngoài: Bạn có thể cần kiểm soát và tương tác với biểu mẫu từ bên ngoài widget chứa nó, và key cho phép bạn truy cập đối tượng FormState để thực hiện các thao tác này.

### Child property

child chứa các trường nhập liệu và nút gửi trong biểu mẫu.

### TextFormField.onTap

Similar 1 lần lấy focus bằng mouse or chấm cảm ứng (đang có focus thì sẽ ko tác dụng)

### TextFormField.onChanged

When the user enters data into any form field and the data changes (e.g., a new character is typed), the onChanged callback is triggered. It then locates the nearest Form widget using the focused widget's context and calls the save() method on the Form to save the data entered in all form fields.

### TextFormField.onSaved

An optional method to call with the final value when the form is saved via FormState.save. (Thực hiện khi hàm [\_formKey].currentState!.save(); được gọi)

// Home1 - demo basic

class Home1 extends StatefulWidget {

  const Home1({super.key});

  @override

  State<Home1> createState() => \_Home1State();

}

class \_Home1State extends State<Home1> {

  final GlobalKey<FormState> \_formKey = GlobalKey<FormState>();

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: const Text('Form Sample')),

      body: Form(

        key:

            \_formKey, // \_formKey là một GlobalKey<FormState> defined above, a key that is unique across the entire app

        child: Column(

          crossAxisAlignment: CrossAxisAlignment.start,

          children: <Widget>[

            TextFormField(

              // This is a convenience widget that wraps a TextField widget in a FormField.

              // Creates a [FormField] that contains a [TextField].

              decoration: const InputDecoration(

                icon: Icon(Icons.email),

                labelText: "Input email address",

                hintText: 'Enter your email',

              ),

              validator: (String? value) {

                // validation values inputed

                if (value == null || value.isEmpty) {

                  return 'Please enter some text';

                }

                if (value.trim().isEmpty) {

                  return 'Cannot contain only spaces.';

                }

                return null;

              },

            ),

            Padding(

              padding: const EdgeInsets.symmetric(vertical: 16.0),

              child: ElevatedButton(

                onPressed: () {

                  // Validate will return true if the form is valid, or false if

                  // the form is invalid.

                  if (\_formKey.currentState!.validate()) {

                    //  the ! operator is used for null safety.

                    // Process data.

                  }

                },

                child: const Text('Submit'),

              ),

            ),

          ],

        ),

      ),

    );

  }

}

// Home2 - onChanged - auto update on screen change

class Home2 extends StatefulWidget {

  const Home2({super.key});

  @override

  State<Home2> createState() => \_Home2State();

}

class \_Home2State extends State<Home2> {

  final GlobalKey<FormState> \_formKey = GlobalKey<FormState>();

  String \_inputValue = '';

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Form onChanged Demo'),

      ),

      body: Padding(

        padding: const EdgeInsets.all(16.0),

        child: Form(

          key: \_formKey,

          child: Column(

            children: [

              TextFormField(

                decoration: const InputDecoration(

                  icon: Icon(Icons.person),

                  border: OutlineInputBorder(),

                  labelText:

                      "Nhập dữ liệu", // labelText and label only used 1 in 2

                  // label: Text('Enter your name'), // This is the label

                  hintText: 'hintText - Enter your name',

                ),

                // If you don't use onChanged,  must use  \_formKey.currentState!.save()

                onChanged: (value) {

                  // update value on UI when the value changes

                  setState(() {

                    // update value on UI and value of \_inputValue

                    \_inputValue = value;

                  });

                },

              ),

              const SizedBox(height: 20),

              Text('Giá trị nhập vào: $\_inputValue'),

            ],

          ),

        ),

      ),

    );

  }

}

// Home2A - get value and show on UI when click button

class Home2A extends StatefulWidget {

  const Home2A({super.key});

  @override

  Home2AState createState() {

    return Home2AState();

  }

}

class Home2AState extends State<Home2A> {

  final GlobalKey<FormState> \_formKey = GlobalKey<FormState>();

  String \_name = '';

  void \_handleFormSubmit() {

    if (\_formKey.currentState!.validate()) {

      \_formKey.currentState!

          .save(); // must be called \_formKey.currentState!.save() to save values (if don't use TextFormField.onChanged)

      // Do something with the collected data, for example, print it.

      debugPrint("Name: $\_name");

      setState(() {}); // update UI when called function \_handleFormSubmit

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Form Demo'),

      ),

      body: Form(

        key: \_formKey,

        child: Padding(

          padding: const EdgeInsets.all(16.0),

          child: Column(

            children: <Widget>[

              TextFormField(

                decoration: const InputDecoration(labelText: 'Name'),

                validator: (value) {

                  if (value == null || value.isEmpty) {

                    return 'Please enter your name';

                  }

                  return null;

                },

                onSaved: (value) {

                  \_name = value!;

                },

              ),

              const SizedBox(height: 20),

              ElevatedButton(

                onPressed: \_handleFormSubmit,

                child: const Text('Submit'),

              ),

              const SizedBox(height: 20),

              Text('Name: $\_name'),

            ],

          ),

        ),

      ),

    );

  }

}

// Home2B - similar 2A but onPressed: dỉrect code ...

class Home2B extends StatefulWidget {

  const Home2B({super.key});

  @override

  Home2BState createState() {

    return Home2BState();

  }

}

class Home2BState extends State<Home2B> {

  final GlobalKey<FormState> \_formKey = GlobalKey<FormState>();

  late String \_name = '';

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: const Text('Form Demo 2B'),

        ),

        body: Form(

          key: \_formKey,

          child: Padding(

            padding: const EdgeInsets.all(16.0),

            child: Column(

              children: <Widget>[

                TextFormField(

                  decoration: const InputDecoration(labelText: 'Name'),

                  validator: (value) {

                    if (value == null || value.isEmpty) {

                      return 'Please enter your name';

                    }

                    return null;

                  },

                  // onTab: similar 1 lần lấy focus bằng mouse or chấm cảm ứng (đang có focus thì sẽ ko tác dụng)

                  onTap: () => {

                    \_formKey.currentState!.save(),

                    debugPrint('onTab: $\_name')

                  },

                  onSaved: (value) {

                    // Thực hiện khi hàm  \_formKey.currentState!.save(); được gọi

                    // An optional method to call with the final value when the form is saved via FormState.save.

                    // This function is automatically called when the form is saved.

                    // You can capture the value here and perform any processing you need.

                    \_name = value!;

                  },

                ),

                const SizedBox(height: 20),

                ElevatedButton(

                  onPressed: () {

                    // Form saving is automatically handled by the Form widget.

                    if (\_formKey.currentState!.validate()) {

                      // Do any other necessary actions.

                      \_formKey.currentState!

                          .save(); // must be called \_formKey.currentState!.save() to save values (if don't use TextFormField.onChanged)

                      debugPrint("Name: $\_name");

                    }

                  },

                  child: const Text('Submit'),

                ),

              ],

            ),

          ),

        ));

  }

}

// Home3 -  SingleActivator(LogicalKeyboardKey.space): NextFocusIntent(),

class Home3 extends StatefulWidget {

  const Home3({super.key});

  @override

  State<Home3> createState() => \_Home3State();

}

class \_Home3State extends State<Home3> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Form LogicalKeyboardKey.space NextFocusIntent'),

      ),

      body: Center(

        child: Shortcuts(

          shortcuts: const <ShortcutActivator, Intent>{

            // Pressing space in the field will now move to the next field.

            SingleActivator(LogicalKeyboardKey.space): NextFocusIntent(),

          },

          child: FocusTraversalGroup(

            // FocusTraversalGroup widget: This widget groups together elements that can receive focus and ensures that they can be traversed in a specific order. It's used to manage the focus behavior.

            child: Form(

              autovalidateMode: AutovalidateMode.always,

              // when the user enters data into any form field and the data changes (e.g., a new character is typed), the onChanged callback is triggered. It then locates the nearest Form widget using the focused widget's context and calls the save() method on the Form to save the data entered in all form fields.

              onChanged: () {

                Form.of(primaryFocus!.context!).save();

              },

              child: Wrap(

                children: List<Widget>.generate(5, (int index) {

                  return Padding(

                    padding: const EdgeInsets.all(8.0),

                    child: ConstrainedBox(

                      constraints: BoxConstraints.tight(const Size(200, 50)),

                      child: TextFormField(

                        // The onSaved property of each TextFormField is set to a function that will be called when the form is saved.q

                        onSaved: (String? value) {

                          debugPrint(

                              'Value for field $index saved as "$value"');

                        },

                      ),

                    ),

                  );

                }),

              ),

            ),

          ),

        ),

      ),

    );

  }

}

## Ink – InkWell - InkResponse

[Ink](https://api.flutter.dev/flutter/material/Ink/Ink.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [Decoration](https://api.flutter.dev/flutter/painting/Decoration-class.html)? decoration, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? height, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Paints a decoration (which can be a simple color) on a [Material](https://api.flutter.dev/flutter/material/Material-class.html).

[Ink.image](https://api.flutter.dev/flutter/material/Ink/Ink.image.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, required [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)> image, [ImageErrorListener](https://api.flutter.dev/flutter/painting/ImageErrorListener.html)? onImageError, [ColorFilter](https://api.flutter.dev/flutter/dart-ui/ColorFilter-class.html)? colorFilter, [BoxFit](https://api.flutter.dev/flutter/painting/BoxFit.html)? fit, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html) alignment = Alignment.center, [Rect](https://api.flutter.dev/flutter/dart-ui/Rect-class.html)? centerSlice, [ImageRepeat](https://api.flutter.dev/flutter/painting/ImageRepeat.html) repeat = ImageRepeat.noRepeat, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) matchTextDirection = false, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? height, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a widget that shows an image (obtained from an [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)) on a [Material](https://api.flutter.dev/flutter/material/Material-class.html).

Widget Ink trong Flutter được sử dụng để tạo ra hiệu ứng nước rơi (ink splash) hoặc hiệu ứng nổi bật (ink highlight) khi một tương tác người dùng xảy ra, như khi bạn chạm vào một phần tử của giao diện người dùng. Ink thường được sử dụng kết hợp với các widget như InkWell hoặc InkResponse để tạo ra các hiệu ứng này.

Ink tạo hiệu ứng giọt nước loang khi click chọn object, có thể set màu nền, height/width

InkWell tạo hiệu ứng khi tương tác với object, có thể set màu các hiệu ứng các events, gọi callback xử lý events

InkResponse tương tự như InkWell.

Khác biệt:

InkWell: đổ bóng khi rê chuột vào hình vuông, splash loang màu toả ra tại điểm click

InkResponse: đổ bóng khi rê chuột vào hình tròn, splash loang màu toả ra tại tâm object

## BuildContext

"Mọi thứ trong Flutter đều là Widget" - Widget chính là một bản vẽ, một bản thiết kế, Flutter sẽ dựa vào bản thiết kế này và render ra hình ảnh UI hiển thị lên màn hình. Nhiều widget kết hợp lại gọi là widget tree. Từng cái Text chúng ta đọc, từng cái Button để chúng ta click, cái Icon, cái Logo được hiển thị trên màn hình đó không phải là Widget mà chính xác là nó được dựng lên từ bản thiết kế Widget.

Một BuildContext như là một tham chiếu (reference) đến cái vị trí của widget (widget's location) trong widget tree. Như chúng ta đã biết mỗi loại widget đều có hàm build(), mỗi hàm build đều nhận 1 BuildContext làm argument. Như vậy mỗi Widget đều có 1 BuildContext đại diện cho vị trí của chính Widget đó trên widget tree.

Context giữ thông tin vị trí các widget tại nơi nó được khai báo (chứ ko phải vị trí truyền vào hàm) ngược lên trên các widget ancestor (ngang level và descendant ko giữ). Nếu có hơn 1 widget ancestor trùng tên thì sẽ lấy context của ancestor gần nhất

Để gọi được các hàm với tham số context tới widget nào đó (như Theme.of, Scaffold.of, MediaQuery.of, Navigator.of, Provider.of,... và InheritedWidget cũng sử dụng hàm of này)thì phải khai báo context ở vị trí phù hợp quy tắc nói trên. Nếu ko đáp ứng quy tắc khi run app thực hiện tác vụ liên quan sẽ báo lỗi liên quan về context

Example

// Click button sẽ báo lỗi ….context… ancestor… do context được khai báo phía trên Scaffold

class MyHomePage extends StatelessWidget {

  const MyHomePage({super.key});

  @override

  Widget build(BuildContext context) { // context khai báo trên Scaffold

    return Scaffold(

      floatingActionButton: FloatingActionButton(onPressed: () {

        Scaffold.of(context).showBottomSheet(

          (context) => const Text('Flutter From Zero to Hero'),

        );

      }),

    );

  }

}

// Fix lỗi C1: Extract Widget FloatingActionButton ra ngoài

class MyHomePage extends StatelessWidget {

  const MyHomePage({super.key});

  @override

  Widget build(BuildContext context) {

    return const Scaffold(

      floatingActionButton: MyButtonWidget(),

    );

  }

}

// Extract button ra thành widget ngoài

class MyButtonWidget extends StatelessWidget {

  const MyButtonWidget({super.key});

  @override

  Widget build(BuildContext context) {

    return FloatingActionButton(onPressed: () {

      Scaffold.of(context).showBottomSheet(

        (context) => const Text('Flutter From Zero to Hero'),

      );

    });

  }

}

// Fix lỗi C2: dùng widget Builder wrap widget tương tác gây lỗi

class MyHomePage extends StatelessWidget {

  const MyHomePage({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      floatingActionButton: Builder(builder: (context) { // Builder wrap button

        return FloatingActionButton(onPressed: () {

          Scaffold.of(context).showBottomSheet(

            (context) => const Text('Flutter From Zero to Hero'),

          );

        });

      }),

    );

  }

}

Example 2

// 2 hàm ScaffoldMessenger.of, Scaffold.of báo lỗi cùng Theme.of không tác dụng

// do context được khai báo trên các widget này

class MyContext extends StatelessWidget {

  const MyContext({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Flutter Demo',

      theme: ThemeData(

        primarySwatch: Colors.red,

        visualDensity: VisualDensity.adaptivePlatformDensity,

        textTheme: const TextTheme(

          displayLarge: TextStyle(

              fontSize: 72, fontWeight: FontWeight.bold, color: Colors.green),

        ),

      ),

      home: Scaffold(

          appBar: AppBar(

            title: const Text('Flutter BuildContext Demo'),

          ),

          body: Center(

              child: Column(

                  mainAxisAlignment: MainAxisAlignment.center,

                  children: <Widget>[

                Text(

                  'Text 1',

                  style: Theme.of(context).textTheme.displayLarge,

                ),

                const Text('Text 2', style: TextStyle(fontSize: 40)),

                ElevatedButton(

                    onPressed: () {

                      ScaffoldMessenger.of(context).showSnackBar(const SnackBar(

                        content: Text('Hahahaha'),

                      ));

                    },

                    child: const Text('SHOW SNACKBAR')),

                FloatingActionButton(onPressed: () {

                  Scaffold.of(context).showBottomSheet((context) => Text(

                      'Flutter From Zero to Hero',

                      style: Theme.of(context).textTheme.titleLarge));

                })

              ]))),

    );

  }

}

// Fix lỗi: dùng Builder(builder: (context) wrap ngoài các vị trí gây lỗi

….

   home: Scaffold(

          appBar: AppBar(

            title: const Text('Flutter BuildContext Demo'),

          ),

          body: Center(child: Builder(builder: (context) { // wrap with Builder

            return Column(

                mainAxisAlignment: MainAxisAlignment.center,

                children: <Widget>[

                  Text(

                    'Text 1',

                    style: Theme.of(context).textTheme.displayLarge,

                  ),

                  const Text('Text 2', style: TextStyle(fontSize: 40)),

                  ElevatedButton(

                      onPressed: () {

                        ScaffoldMessenger.of(context)

                            .showSnackBar(const SnackBar(

                          content: Text('Hahahaha'),

                        ));

                      },

                      child: const Text('SHOW SNACKBAR')),

                  FloatingActionButton(onPressed: () {

                    Scaffold.of(context).showBottomSheet((context) => Text(

                        'Flutter From Zero to Hero',

                        style: Theme.of(context).textTheme.titleLarge));

                  })

                ]);

          }))),

    );

  }

}

Ngoài 2 cách fix lỗi Extract Widget và Wrap with Builder widget ở trên còn 1 cách nữa là dùng GlobalKey

Example 3

// demo context OngBa - ChaMe - Con1 - Con2

class MyContext1 extends StatelessWidget {

  const MyContext1({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

        title: 'Flutter Demo',

        theme: ThemeData(

            primarySwatch: Colors.red,

            textTheme: const TextTheme(

                titleLarge: TextStyle(

                    fontSize: 22,

                    fontWeight: FontWeight.bold,

                    color: Colors.green))),

        home: Scaffold(

            appBar: AppBar(

              title: const Text('Flutter BuildContext Demo'),

            ),

            body: const Center(

              child: OngBa(

                  child: ChaMe(

                      child: Column(

                children: [Con1(), Con2()],

              ))),

            )));

  }

}

class OngBa extends StatelessWidget {

  final Widget child;

  const OngBa({super.key, required this.child});

  @override

  Widget build(BuildContext context) {

    return Container(

      child: child,

    );

  }

  static OngBa? of(BuildContext context) {

    return context.findAncestorWidgetOfExactType();

  }

  getOngBa() {

    return 'getOngBa';

  }

}

class ChaMe extends StatelessWidget {

  final Widget child;

  const ChaMe({super.key, required this.child});

  @override

  Widget build(BuildContext context) {

    return Container(

      child: child,

    );

  }

  static ChaMe? of(BuildContext context) {

    return context.findAncestorWidgetOfExactType();

  }

  getChaMe() {

    return 'getChaMe';

  }

}

class Con1 extends StatelessWidget {

  const Con1({super.key});

  @override

  Widget build(BuildContext context) {

    OngBa ofOngBa = OngBa.of(context) as OngBa;

    String ch = ofOngBa.getOngBa();

    return Text('Is Con 1 - $ch',

        style: Theme.of(context).textTheme.titleLarge);

  }

}

class Con2 extends StatelessWidget {

  const Con2({super.key});

  @override

  Widget build(BuildContext context) {

    ChaMe ofChaMe = ChaMe.of(context) as ChaMe;

    String ch = ofChaMe.getChaMe();

    return Text('Is Con 2 - $ch',

        style: Theme.of(context).textTheme.titleLarge);

  }

}

Một số thuộc tính của context object:

// context.widget - tên widget hiện tại

// context.runtimeType - kiểu runtime của widget hiện tại (StatelessWidget or StatefulWidget)

// context.describeElement('Description') - trả về một chuỗi mô tả của widget hiện tại (là chuỗi truyền vào phương thức describeElement)

// context.findAncestorWidgetOfExactType<Center>()) - trả về widget cha gần nhất (widget chứa context)có kiểu là Center (và có thể cùng với các thuộc tính của nó)

// print(context.widget);

// print(context.runtimeType);

// print(context.describeElement('Description'));

// print(context.findRootAncestorStateOfType Function() Function );

## Phân loại widget theo khả năng định dạng

Container

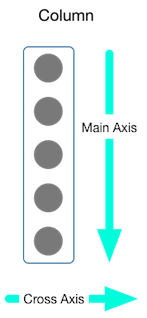
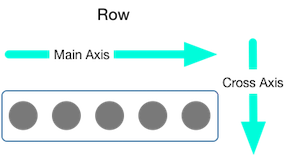
## Cách lựa chọn, phối hợp widget tối ưu cho layout

(tổng hợp nội dung các widget ở trên đưa vào đây)

* Row and Column are basic primitive widgets for horizontal and vertical layouts—these low-level widgets allow for maximum customization. Flutter also offers specialized, higher level widgets that might be sufficient for your needs. For example, instead of Row you might prefer ListTile, an easy-to-use widget with properties for leading and trailing icons, and up to 3 lines of text. Instead of Column, you might prefer ListView, a column-like layout that automatically scrolls if its content is too long to fit the available space. For more information, see Common layout widgets.
* Aligning widgets

You control how a row or column aligns its children using the mainAxisAlignment and crossAxisAlignment properties. For a row, the main axis runs horizontally and the cross axis runs vertically. For a column, the main axis runs vertically and the cross axis runs horizontally.

The MainAxisAlignment and CrossAxisAlignment enums offer a variety of constants for controlling alignment.



* Sizing widgets

When a layout is too large to fit a device, a yellow and black striped pattern appears along the affected edge.

Widgets can be sized to fit within a row or column by using the Expanded widget.

* Packing widgets
* By default, a row or column occupies as much space along its main axis as possible, but if you want to pack the children closely together, set its mainAxisSize to MainAxisSize.min.

https://docs.flutter.dev/ui/layout/constraints

## Tầm vực áp dụng định dạng các object và mức độ ưu tiên trong app

Trong ứng dụng Flutter, để định dạng đối tượng nào đó ta có một số tùy chọn tầm vực áp dụng khác nhau:

* 1. Áp dụng toàn cục với Theme:

Bạn có thể sử dụng Theme để định nghĩa các kiểu, màu sắc, phông chữ và các thuộc tính khác cho toàn bộ ứng dụng. Điều này giúp bạn duy trì tính nhất quán trong giao diện.

* 1. Áp dụng cho một loại đối tượng cụ thể:

Bạn có thể tạo các phong cách tùy chỉnh cho một loại đối tượng cụ thể, ví dụ như nút, văn bản, hoặc hình ảnh. Điều này giúp bạn áp dụng kiểu đó cho tất cả các trường hợp sử dụng đối tượng loại đó.

* 1. Áp dụng cho một trạng thái cụ thể:

Một số thuộc tính như màu sắc, vị trí, hoặc kích thước có thể thay đổi dựa trên trạng thái của đối tượng. Bạn có thể sử dụng các thuộc tính như MaterialStateProperty để xác định các giá trị dựa trên trạng thái.

* 1. Áp dụng tùy chỉnh cho từng đối tượng riêng lẻ:

Bạn có thể áp dụng các thuộc tính tùy chỉnh cho từng đối tượng riêng lẻ. Điều này thường dùng khi bạn cần định dạng đối tượng một cách riêng biệt so với các đối tượng khác.

* 1. Sử dụng thuộc tính cụ thể cho từng widget:

Một số widget cung cấp các thuộc tính cụ thể để tùy chỉnh giao diện của chính widget đó, chẳng hạn như style cho các nút (TextButton, ElevatedButton, ...) hoặc textStyle cho Text.

Bạn có thể kết hợp nhiều tùy chọn để định dạng đối tượng một cách chi tiết. Ví dụ, bạn có thể sử dụng ButtonStyle hoặc TextStyle kết hợp với Theme để áp dụng kiểu chung nhưng vẫn có thể điều chỉnh từng trường hợp cụ thể.

Mức độ ưu tiên khi các giá trị định dạng bị ghi đè, từ mức ưu tiên cao nhất đến thấp nhất:

1. Giá trị cụ thể cho widget cụ thể:

Các thuộc tính được cung cấp trực tiếp cho một widget sẽ được ưu tiên cao nhất và ghi đè lên tất cả các giá trị định dạng khác. Ví dụ, nếu bạn cung cấp thuộc tính style cho một nút (TextButton, ElevatedButton, ...), nó sẽ ghi đè lên bất kỳ giá trị nào được xác định từ các tùy chọn khác.

1. Thuộc tính được định nghĩa bởi các widget con:

Nếu một thuộc tính được xác định bởi các widget con bên trong một widget cha, giá trị của widget con gần nhất về mặt cây widget sẽ được sử dụng.

1. Theme:

Các giá trị được định nghĩa trong Theme sẽ áp dụng cho toàn bộ ứng dụng, trừ khi các giá trị cụ thể được cung cấp cho widget hoặc widget con.

1. Trạng thái đối tượng:

Các thuộc tính có thể thay đổi dựa trên trạng thái (ví dụ như trạng thái hovered, pressed, disabled, ...) có thể được xác định sử dụng MaterialStateProperty. Giá trị của trạng thái được xác định trong thời điểm thực thi của widget.

1. Giá trị mặc định của thuộc tính:

Nếu không có giá trị nào được cung cấp, giá trị mặc định của thuộc tính sẽ được sử dụng.

## Các loại hộp thoại

### showDialog<T> function

[Future](https://api.flutter.dev/flutter/dart-async/Future-class.html)<T?> showDialog<T>(

{required [BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context,

required [WidgetBuilder](https://api.flutter.dev/flutter/widgets/WidgetBuilder.html) builder,

[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) barrierDismissible = true,

[Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? barrierColor = Colors.black54,

[String](https://api.flutter.dev/flutter/dart-core/String-class.html)? barrierLabel,

[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) useSafeArea = true,

[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) useRootNavigator = true,

[RouteSettings](https://api.flutter.dev/flutter/widgets/RouteSettings-class.html)? routeSettings,

[Offset](https://api.flutter.dev/flutter/dart-ui/Offset-class.html)? anchorPoint,

[TraversalEdgeBehavior](https://api.flutter.dev/flutter/widgets/TraversalEdgeBehavior.html)? traversalEdgeBehavior})

showDialog là một phương thức tĩnh trong Flutter được sử dụng để hiển thị một hộp thoại trên màn hình. Nó cho phép bạn hiển thị một Dialog, AlertDialog, hoặc SimpleDialog trên màn hình.

Bạn có thể sử dụng showDialog để hiển thị một hộp thoại thông báo, yêu cầu xác nhận từ người dùng, hoặc hiển thị một danh sách các tùy chọn cho người dùng chọn.

Phương thức showDialog có hai tham số chính là context và builder. Tham số context là BuildContext hiện tại và tham số builder là một hàm trả về một widget, thường là một Dialog, AlertDialog, hoặc SimpleDialog.

Trong Flutter, Dialog, AlertDialog, và SimpleDialog đều là các widgets có thể được sử dụng để hiển thị một hộp thoại trên màn hình. Tuy nhiên, chúng có một số khác biệt về giao diện và chức năng.

* Dialog là một hộp thoại thông thường có thể được tùy chỉnh để phù hợp với nhu cầu của bạn. Nó có nền trong suốt và có thể có bất kỳ hình dạng hoặc kích thước nào. Bạn có thể thêm bất kỳ widgets nào bạn muốn vào Dialog để tạo một hộp thoại tùy chỉnh.
* AlertDialog là một hộp thoại được xây sẵn với thiết kế và chức năng cụ thể. Nó có tiêu đề, nội dung và các hành động, và thường được sử dụng để hiển thị thông tin quan trọng hoặc yêu cầu xác nhận từ người dùng. Nó có nền màu trắng và các góc được làm tròn.

To close an alert dialog, one of its actions must be selected.

* SimpleDialog tương tự như AlertDialog, nhưng có thiết kế đơn giản hơn và thường được sử dụng để hiển thị danh sách tùy chọn cho người dùng. Nó cũng có nền màu trắng và các góc được làm tròn và có thể có một tiêu đề và một danh sách tùy chọn.

SimpleDialog cho phép các tương tác sau:

* + Nhấn vào một hành động để chọn nó và đóng hộp thoại
  + Nhấn vào bên ngoài hộp thoại để đóng hộp thoại mà không thực hiện hành động nào

Không nên đặt các button trong SimpleDialog, bản thân sự lựa chọn tự gọi action được khi chạm vào.

### Dialog class

[Dialog](https://api.flutter.dev/flutter/material/Dialog/Dialog.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? elevation, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? shadowColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? surfaceTintColor, [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html) insetAnimationDuration = const Duration(milliseconds: 100), [Curve](https://api.flutter.dev/flutter/animation/Curve-class.html) insetAnimationCurve = Curves.decelerate, [EdgeInsets](https://api.flutter.dev/flutter/painting/EdgeInsets-class.html)? insetPadding = \_defaultInsetPadding, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)? shape, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html)? alignment, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a dialog.

*const*

[Dialog.fullscreen](https://api.flutter.dev/flutter/material/Dialog/Dialog.fullscreen.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html) insetAnimationDuration = Duration.zero, [Curve](https://api.flutter.dev/flutter/animation/Curve-class.html) insetAnimationCurve = Curves.decelerate, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a fullscreen dialog.

Example Dialog and Dialog.fullscreen

class Dialog1 extends StatelessWidget {

  const Dialog1({super.key});

  @override

  Widget build(BuildContext context) {

    return Center(

      child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        children: <Widget>[

          TextButton(

            onPressed: () => showDialog(

              // showDialog is a static method that takes a builder parameter

              context:

                  context, // context is the current BuildContext - required

              builder: (BuildContext context) => Dialog(

                //  builder is a function that returns a widget - required

                backgroundColor: Colors.amber,

                shape: RoundedRectangleBorder(

                    borderRadius: BorderRadius.circular(20)),

                shadowColor: Colors.blue,

                child: Padding(

                  padding: const EdgeInsets.all(8.0),

                  child: Column(

                    mainAxisSize: MainAxisSize.min,

                    mainAxisAlignment: MainAxisAlignment.center,

                    children: <Widget>[

                      const Text('This is a typical dialog.'),

                      const SizedBox(height: 15),

                      TextButton(

                        onPressed: () {

                          Navigator.pop(

                              context); // dismiss dialog - đóng dialog

                        },

                        child: const Text('Close'),

                      ),

                    ],

                  ),

                ),

              ),

            ),

            child: const Text('Show Dialog'),

          ),

          const SizedBox(height: 10),

          TextButton(

            onPressed: () => showDialog(

              context: context,

              builder: (BuildContext context) => Dialog.fullscreen(

                // fullscreen dialog - bung chiếm hết screen

                child: Column(

                  mainAxisSize: MainAxisSize.min,

                  mainAxisAlignment: MainAxisAlignment.center,

                  children: <Widget>[

                    const Text('This is a fullscreen dialog.'),

                    const SizedBox(height: 15),

                    TextButton(

                      onPressed: () {

                        Navigator.pop(context);

                      },

                      child: const Text('Close'),

                    ),

                  ],

                ),

              ),

            ),

            child: const Text('Show Fullscreen Dialog'),

          ),

        ],

      ),

    );

  }

}

### AlertDialog

[AlertDialog](https://api.flutter.dev/flutter/material/AlertDialog/AlertDialog.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? icon, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? iconPadding, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? iconColor, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? title, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? titlePadding, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? titleTextStyle, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? content, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? contentPadding, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? contentTextStyle, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)>? actions, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? actionsPadding, [MainAxisAlignment](https://api.flutter.dev/flutter/rendering/MainAxisAlignment.html)? actionsAlignment, [OverflowBarAlignment](https://api.flutter.dev/flutter/widgets/OverflowBarAlignment.html)? actionsOverflowAlignment, [VerticalDirection](https://api.flutter.dev/flutter/painting/VerticalDirection.html)? actionsOverflowDirection, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? actionsOverflowButtonSpacing, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? buttonPadding, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? elevation, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? shadowColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? surfaceTintColor, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticLabel, [EdgeInsets](https://api.flutter.dev/flutter/painting/EdgeInsets-class.html) insetPadding = \_defaultInsetPadding, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)? shape, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html)? alignment, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) scrollable = false})

Creates an alert dialog.

Example AlertDialog

* actions: thuộc tính của widget AlertDialog trong được sử dụng để xác định danh sách các hành động (nút hoặc lựa chọn) mà người dùng có thể thực hiện khi họ tương tác với hộp thoại. Mỗi phần tử trong danh sách actions là một widget, thường là các nút nhấn hoặc các lựa chọn để người dùng chọn.

Example 1

class ActionChip2 extends StatelessWidget {

  const ActionChip2({super.key});

  @override

  Widget build(BuildContext context) {

    return ActionChip(

      label: const Text('Open Dialog'),

      onPressed: () {

        showDialog(

          context: context,

          builder: (BuildContext context) {

            return AlertDialog(

              title: const Text('Dialog Title'), // title dialog

              content: const Text('This is a dialog content.'), // content dialog

              actions: [ // các hành động, thường là button

                TextButton(

                  onPressed: () {

                    Navigator.pop(context);

                  },

                  child: const Text('Close'),

                ),

              ],

            );

          },

        );

      },

    );

  }

}

Example 2

 ActionChip(

            label: const Text('Open Dialog'),

            onPressed: () {

              showDialog(

                context: context,

                builder: (BuildContext context) {

                  return AlertDialog(

                    icon: const Icon(Icons.warning),

                    iconColor: Colors.red,

                    title: const Text('Dialog Title'), // tiêu đề

                    content:

                        const Text('This is a dialog content.'), //  nội dung

                    actions: [

                      // nút

                      TextButton(

                        onPressed: () {

                          Navigator.pop(context);

                        },

                        child: const Text('Close'),

                      ),

                    ],

                  );

                },

              );

            },

          ),

Example 3

 Widget build(BuildContext context) {

    return ElevatedButton(

      onPressed: () {

        showDialog(

          context: context,

          builder: (BuildContext context) {

            return AlertDialog(

              title: const Text('Xác nhận'),

              content: const Text('Bạn có chắc muốn xóa mục này?'),

              actions: [

                TextButton(

                  onPressed: () {

                    Navigator.of(context).pop(); // Đóng hộp thoại

                  },

                  child: const Text('Hủy'),

                ),

                TextButton(

                  onPressed: () {

                    // Thực hiện xóa mục ở đây

                    Navigator.of(context).pop(); // Đóng hộp thoại sau khi xong

                  },

                  child: const Text('Xóa'),

                ),

              ],

            );

          },

        );

      },

      child: const Text('Xóa Mục'),

    );

### SimpleDialog

[SimpleDialog](https://api.flutter.dev/flutter/material/SimpleDialog/SimpleDialog.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? title, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html) titlePadding = const EdgeInsets.fromLTRB(24.0, 24.0, 24.0, 0.0), [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? titleTextStyle, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)>? children, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html) contentPadding = const EdgeInsets.fromLTRB(0.0, 12.0, 0.0, 16.0), [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? elevation, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? shadowColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? surfaceTintColor, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticLabel, [EdgeInsets](https://api.flutter.dev/flutter/painting/EdgeInsets-class.html) insetPadding = \_defaultInsetPadding, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)? shape, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html)? alignment})

Creates a simple dialog.

Example

  showDialog(

    context: context,

    builder: (BuildContext context) {

      return SimpleDialog(

        title: const Text(

          'Chọn một tùy chọn',

          style: TextStyle(

            fontSize: 20.0,

            fontWeight: FontWeight.bold,

            color: Colors.blue,

          ),

        ),

        contentPadding: const EdgeInsets.all(16.0),

        backgroundColor: Colors.yellow[100],

        shape: RoundedRectangleBorder(

          borderRadius: BorderRadius.circular(10.0),

        ),

        children: <Widget>[

          SimpleDialogOption( // An option used in a SimpleDialog.

              padding: const EdgeInsets.all(40), // Set padding cho option

              onPressed: () {

                // Xử lý tùy chọn 1

                Navigator.of(context).pop(); // Đóng hộp thoại

              },

              child: const Text('Tùy chọn 1',

                  style: TextStyle(

                    color: Colors.red,

                  ) // Set màu sắc ở đây.

                  )),

          SimpleDialogOption( // An option used in a SimpleDialog.

              onPressed: () {

                // Xử lý tùy chọn 2

                Navigator.of(context).pop(); // Đóng hộp thoại

              },

              child: const Text('Tùy chọn 2',

                  style: TextStyle(

                    color: Colors.red,

                  ) // Set màu sắc ở đây

                  )),

        ],

      );

    },

  );

Example 2

  @override

  Widget build(BuildContext context) {

    return Center(

      child: ElevatedButton(

        onPressed: () {

          // Khi nút được nhấn, hiển thị SimpleDialog

          \_showDialog3(context);

        },

        child: const Text('Mở SimpleDialog'),

      ),

    );

  }

}

void \_showDialog3(BuildContext context) {

  showDialog(

      context: context,

      builder: (BuildContext context) {

        return SimpleDialog(

          title: const Text('Set backup account'),

          children: [

            SimpleDialogItem( // call hàm SimpleDialogItem với các tham số kèm theo

              icon: Icons.account\_circle,

              color: Colors.orange,

              text: 'user01@gmail.com',

              onPressed: () {

                Navigator.pop(context, 'user01@gmail.com');

              },

            ),

            SimpleDialogItem(

              icon: Icons.account\_circle,

              color: Colors.green,

              text: 'user02@gmail.com',

              onPressed: () {

                Navigator.pop(context, 'user02@gmail.com');

              },

            ),

          ],

        );

      });

}

// Tạo item SimpleDialogOption cho SimpleDialog

class SimpleDialogItem extends StatelessWidget {

  const SimpleDialogItem(

      {Key? key, this.icon, this.color, this.text, this.onPressed})

      : super(key: key);

  final IconData? icon;

  final Color? color;

  final String? text;

  final VoidCallback? onPressed;

  @override

  Widget build(BuildContext context) {

    return SimpleDialogOption(

      onPressed: onPressed,

      child: Row(

        mainAxisAlignment: MainAxisAlignment.start,

        crossAxisAlignment: CrossAxisAlignment.center,

        children: [

          Icon(icon, size: 36.0, color: color),

          Padding(

            padding: const EdgeInsetsDirectional.only(start: 16.0),

            child: Text(text != null ? text.toString() : ''),

          ),

        ],

      ),

    );

  }

}

### Full-screen dialog

Full-screen dialogs group a series of tasks, such as creating a calendar entry with the event title, date, location, and time. Because they take up the entire screen, full-screen dialogs are the only dialogs over which other dialogs can appear.

Cách setup

* Dùng MaterialPageRoute để gọi hàm show dialog và set thuộc tính fullscreenDialog: true
* Tại hàm show dialog: Phải có Scaffold để hiển thị AppBar

Example 1

class FullScreenDialog1 extends StatelessWidget {

  const FullScreenDialog1({super.key});

  @override

  Widget build(BuildContext context) {

    return const MyHomePage();

  }

}

class MyHomePage extends StatelessWidget {

  const MyHomePage({super.key});

  @override

  Widget build(BuildContext context) {

    return Center(

      child: TextButton(

        style: TextButton.styleFrom(

          backgroundColor: const Color.fromARGB(255, 182, 244, 249),

          foregroundColor: const Color.fromARGB(255, 210, 5, 203),

          padding: const EdgeInsets.all(15.0),

          shape: const RoundedRectangleBorder(

            borderRadius: BorderRadius.all(Radius.circular(16.0)),

          ),

        ),

        onPressed: () {

          // Future<T?> push<T extends Object?>(BuildContext context, Route<T> route)

          /\* Dùng MaterialPageRoute để gọi FullScreenDialog() và set fullscreenDialog: true

          \*/

          Navigator.push(

            context,

            MaterialPageRoute<void>(

              builder: (BuildContext context) => const FullScreenDialog(),

              fullscreenDialog: true,

            ),

          );

        },

        child: const Text("SHOW DIALOG"),

      ),

    );

  }

}

class FullScreenDialog extends StatelessWidget {

  const FullScreenDialog({super.key});

  @override

  Widget build(BuildContext context) {

    // Phải có Scaffold để hiển thị AppBar

    return Scaffold(

      appBar: AppBar(

        backgroundColor: const Color(0xFF6200EE),

        title: const Text('Full-screen Dialog'),

      ),

      body: const Center(

        child: Text("Full-screen dialog"),

      ),

    );

  }

}

### Get data from dialog

Example 1

// .then((result) { ... }) để lấy giá trị trả về từ hộp thoại

class MyWidget extends StatefulWidget {

  const MyWidget({super.key});

  @override

  State<MyWidget> createState() => \_MyWidgetState();

}

class \_MyWidgetState extends State<MyWidget> {

  String \_inputValue = ''; // Biến để lưu trữ giá trị nhập từ hộp thoại

  void \_openDialog() {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: const Text('Enter a Value'),

          content: TextField(

            onChanged: (value) {

              setState(() {

                \_inputValue = value;

              });

            },

          ),

          actions: <Widget>[

            TextButton(

              child: const Text('OK'),

              onPressed: () {

                Navigator.of(context).pop(\_inputValue);

                debugPrint(\_inputValue);

              },

            ),

            TextButton(

              child: const Text('Cancel'),

              onPressed: () {

                Navigator.of(context).pop();

              },

            ),

          ],

        );

      },

    ).then((result) { // .then((result) { ... }) để lấy giá trị trả về từ hộp thoại

      setState(() {

        \_inputValue = result ?? '';

      });

    });

  }

  @override

  Widget build(BuildContext context) {

    return Center(

      child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        children: <Widget>[

          Text('Entered Value: $\_inputValue'), // Hiển thị giá trị nhập vào

          const SizedBox(height: 20),

          ElevatedButton(

            onPressed: \_openDialog,

            child: const Text('Open Dialog'),

          ),

        ],

      ),

    );

  }

}

Example 2

//// Get input value from dialog dùng async/await - sử dụng trong trường hợp cần xử lý bất đồng bộ

class MyHomePage extends StatefulWidget {

  const MyHomePage({super.key});

  @override

  State<MyHomePage> createState() => \_MyHomePageState();

}

class \_MyHomePageState extends State<MyHomePage> {

  String \_inputValue = ''; // Biến để lưu trữ giá trị nhập từ hộp thoại

  void \_openDialog() async {

    String? result = await showDialog(

      // async/await để chờ kết quả trả về từ hộp thoại. sử dụng trong trường hợp cần xử lý bất đồng bộ

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: const Text('Enter a Value'),

          content: TextField(

            onChanged: (value) {

              setState(() {

                \_inputValue = value;

              });

            },

          ),

          actions: <Widget>[

            TextButton(

              child: const Text('OK'),

              onPressed: () {

                Navigator.of(context).pop(\_inputValue);

                debugPrint(\_inputValue);

              },

            ),

            TextButton(

              child: const Text('Cancel'),

              onPressed: () {

                Navigator.of(context).pop();

              },

            ),

          ],

        );

      },

    );

    setState(() {

      \_inputValue = result ??

          ''; // Nếu người dùng không nhập giá trị thì gán giá trị rỗng

    });

  }

  @override

  Widget build(BuildContext context) {

    return Center(

      child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        children: <Widget>[

          Text('Entered Value: $\_inputValue'), // Hiển thị giá trị nhập vào

          const SizedBox(height: 20),

          ElevatedButton(

            onPressed: \_openDialog,

            child: const Text('Open Dialog'),

          ),

        ],

      ),

    );

  }

}

Example 3

// Get values from CheckboxListTile and RadioListTile from dialog

class MyHomePage2 extends StatefulWidget {

  const MyHomePage2({super.key});

  @override

  State<MyHomePage2> createState() => \_MyHomePage2State();

}

class \_MyHomePage2State extends State<MyHomePage2> {

  List<String> \_selectedValues = [];

  String \_selectedRadioValue = '';

  void \_openCheckboxDialog() async {

    List<String>? result = await showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: const Text('Select values'),

          content: StatefulBuilder(

              // StatefulBuilder để cập nhật lại giao diện khi click trên checkbox

              builder: (BuildContext context, StateSetter setState) {

            return Column(

              mainAxisSize: MainAxisSize.min,

              children: <Widget>[

                CheckboxListTile(

                  title: const Text('Value 1'),

                  value: \_selectedValues.contains(

                      'Value 1'), // nếu list chứa 'Value 1' thì value = true, ngược lại value = false

                  onChanged: (bool? value) {

                    setState(() {

                      if (value ?? false) {

                        // nếu value = true thì add 'Value 1' vào list, ngược lại remove 'Value 1' khỏi list

                        \_selectedValues.add('Value 1');

                      } else {

                        \_selectedValues.remove('Value 1');

                      }

                    });

                  },

                ),

                CheckboxListTile(

                  title: const Text('Value 2'),

                  value: \_selectedValues.contains('Value 2'),

                  onChanged: (bool? value) {

                    setState(() {

                      if (value ?? false) {

                        \_selectedValues.add('Value 2');

                      } else {

                        \_selectedValues.remove('Value 2');

                      }

                    });

                  },

                ),

                CheckboxListTile(

                  title: const Text('Value 3'),

                  value: \_selectedValues.contains('Value 3'),

                  onChanged: (bool? value) {

                    setState(() {

                      if (value ?? false) {

                        \_selectedValues.add('Value 3');

                      } else {

                        \_selectedValues.remove('Value 3');

                      }

                    });

                  },

                ),

              ],

            );

          }),

          actions: <Widget>[

            TextButton(

              child: const Text('OK'),

              onPressed: () {

                Navigator.of(context)

                    .pop(\_selectedValues); // result = .pop(value)

                debugPrint(\_selectedValues.toString());

              },

            ),

            TextButton(

              child: const Text('Cancel'),

              onPressed: () {

                Navigator.of(context).pop(); // result = null

                debugPrint(\_selectedValues.toString());

              },

            ),

          ],

        );

      },

    );

    if (result != null) {

      setState(() {

        \_selectedValues = result;

      });

    }

  }

  void \_openRadioDialog() async {

    String? result = await showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: const Text('Select a value'),

          content: StatefulBuilder(

              // StatefulBuilder để cập nhật lại giao diện khi click trên radio button

              builder: (BuildContext context, StateSetter setState) {

            return Column(

              mainAxisSize: MainAxisSize.min,

              children: [

                RadioListTile<String>(

                  title: const Text('Value 1'),

                  value: 'Value 1',

                  groupValue: \_selectedRadioValue,

                  onChanged: (String? value) {

                    setState(() {

                      \_selectedRadioValue = value!;

                    });

                  },

                ),

                RadioListTile<String>(

                  title: const Text('Value 2'),

                  value: 'Value 2',

                  groupValue: \_selectedRadioValue,

                  onChanged: (String? value) {

                    setState(() {

                      \_selectedRadioValue = value!;

                    });

                  },

                ),

                RadioListTile<String>(

                  title: const Text('Value 3'),

                  value: 'Value 3',

                  groupValue: \_selectedRadioValue,

                  onChanged: (String? value) {

                    setState(() {

                      \_selectedRadioValue = value!;

                    });

                  },

                ),

              ],

            );

          }),

          actions: <Widget>[

            TextButton(

              child: const Text('OK'),

              onPressed: () {

                Navigator.of(context)

                    .pop(\_selectedRadioValue); // result = .pop(value)

                debugPrint(\_selectedRadioValue);

              },

            ),

            TextButton(

              child: const Text('Cancel'),

              onPressed: () {

                Navigator.of(context).pop(); // result = null

                debugPrint(\_selectedRadioValue.toString());

              },

            ),

          ],

        );

      },

    );

    if (result != null) {

      setState(() {

        \_selectedRadioValue = result;

      });

    }

  }

  @override

  Widget build(BuildContext context) {

    return Center(

      child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        children: <Widget>[

          Text(

              'Selected Values: ${\_selectedValues.join(', ')}'), // Show the selected values from the checkbox list

          const SizedBox(height: 20),

          ElevatedButton(

            onPressed: \_openCheckboxDialog,

            child: const Text('Open Checkbox Dialog'),

          ),

          const SizedBox(height: 20),

          Text(

              'Selected Value: $\_selectedRadioValue'), // Show the selected value from the radio button list

          const SizedBox(height: 20),

          ElevatedButton(

            onPressed: \_openRadioDialog,

            child: const Text('Open Radio Dialog'),

          ),

        ],

      ),

    );

  }

}

Example 4

// Get 1 value in list item on SimpleDialog - click choose item là gọi action luôn

class MyHomePage3 extends StatefulWidget {

  const MyHomePage3({super.key});

  @override

  State<MyHomePage3> createState() => \_MyHomePage3State();

}

class \_MyHomePage3State extends State<MyHomePage3> {

  String \_selectedValue = '';

  void \_openDialog() async {

    String result = await showDialog(

      context: context,

      builder: (BuildContext context) {

        return SimpleDialog(

          title: const Text('Select a value'),

          children: <Widget>[

            SimpleDialogOption(

              onPressed: () {

                Navigator.of(context).pop('Value 1');

              },

              child: const Text('Value 1'),

            ),

            SimpleDialogOption(

              onPressed: () {

                Navigator.of(context).pop('Value 2');

              },

              child: const Text('Value 2'),

            ),

            SimpleDialogOption(

              onPressed: () {

                Navigator.of(context).pop('Value 3');

              },

              child: const Text('Value 3'),

            ),

          ],

        );

      },

    );

    setState(() {

      \_selectedValue = result;

    });

    debugPrint(\_selectedValue);

  }

  @override

  Widget build(BuildContext context) {

    return Center(

      child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        children: <Widget>[

          Text('Selected Value: $\_selectedValue'), // Show the selected value

          const SizedBox(height: 20),

          ElevatedButton(

            onPressed: \_openDialog,

            child: const Text('Open SimpleDialogOption'),

          ),

        ],

      ),

    );

  }

}

## Navigation and routing

Flutter cung cấp một hệ thống hoàn chỉnh để điều hướng giữa các màn hình và xử lý các liên kết sâu. Các ứng dụng nhỏ không có liên kết sâu phức tạp có thể sử dụng Navigator, trong khi các ứng dụng có yêu cầu điều hướng và liên kết sâu cụ thể cũng nên sử dụng Router để xử lý chính xác các liên kết sâu trên Android và iOS, và luôn đồng bộ hóa với thanh địa chỉ khi ứng dụng đang chạy trên web.

Để định cấu hình ứng dụng Android hoặc iOS của bạn nhằm xử lý các liên kết sâu, hãy xem Deep link.

Widget tree tổ chức các screen theo cấu trúc stack - LIFO. Hành động .push - đẩy screen vào stack (navigate to new page), .pop lấy screen ra khỏi stack (back to previous page).

Khi gọi phương thức .push lại với screen cũ (đã .push trước đó) thì widget tạo screen này vẫn được .push tiếp vào stack (là screen mới nhất trong stack)

Back button trên AppBar tương đương gọi .pop - back to previous page

Khi dùng Navigator.pushName(…) luồng code sẽ quay lên MaterialApp.router dò trước, nếu route ko tồn tại mới dò trong onGenerateRoute (nếu có sẽ bỏ qua onGenerateRoute)

Dùng Navigator.pushName(…) đi qua cấu hình MaterialApp.router ko cần gán RouteSettings; đi qua cấu hình onGenerateRoute phải gán RouteSettings với cú pháp

return MaterialPageRoute(

                builder: (context) => const ProfileScreen2(),

          settings: settings); // gán RouteSettings

Khi dùng Navigator.push không đi qua cấu hình routes hoặc onGenerateRoute của MaterialApp. Thay vào đó, Navigator.push được sử dụng để thêm một màn hình mới vào ngăn xếp (stack) của Navigator hiện tại mà không liên quan đến việc đăng ký các route trong MaterialApp.

Dùng Navigator.push gán RouteSettings như sau

 ElevatedButton(

              onPressed: () {

                Navigator.push(

                  context,

                  MaterialPageRoute(

                    builder: (context) => const ProfileScreen(),

                    settings: const RouteSettings(

                        name:

                            '/profile'), //truyền theo route name ở đây, để UI có thể get ra bằng ModalRoute.of(context)?.settings.name;

                  ),

                );

### Một số cách truyền data giữa các màn hình:

* Qua phương thức Navigator.Push với Constructor màn hình đích:
* class HomeScreen extends StatelessWidget {
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* appBar: AppBar(
* title: Text('Home Screen'),
* ),
* body: Center(
* child: ElevatedButton(
* onPressed: () {
* // Khi nút được nhấn, chuyển đến Màn hình chi tiết và truyền tham số.
* Navigator.push(
* context,
* MaterialPageRoute(
* builder: (context) => DetailScreen(data: 'Dữ liệu từ Màn hình gốc'), // gọi contructor DetailScreen
* ),
* );
* },
* child: Text('Chuyển đến Màn hình chi tiết'),
* ),
* ),
* );
* }
* }

// class DetailScreen với constructor có tham số

* class DetailScreen extends StatelessWidget {
* final String data;
* DetailScreen({required this.data}); // Nhận tham số data
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* appBar: AppBar(
* title: Text('Màn hình chi tiết'),
* ),
* body: Center(
* child: Column(
* mainAxisAlignment: MainAxisAlignment.center,
* children: [
* Text('Dữ liệu nhận được: $data'), // Hiển thị dữ liệu trên Màn hình chi tiết
* ElevatedButton(
* onPressed: () {
* Navigator.pop(context); // Quay trở lại Màn hình gốc
* },
* child: Text('Quay lại'),
* ),
* ],
* ),
* ),
* );
* }
* }
* Truyền từ screen 2 về screen 1 qua phương thức Navigator.push trong hàm async - await:
* class ScreenA extends StatefulWidget {
* const ScreenA({super.key});
* @override
* State<ScreenA> createState() => \_ScreenAState();
* }
* class \_ScreenAState extends State<ScreenA> {
* String \_displayText = ''; // Biến để lưu trữ giá trị result
* @override
* Widget build(BuildContext context) {
* return Center(
* child: Column(
* mainAxisAlignment: MainAxisAlignment.center,
* children: [
* const Text(
* 'Result:',
* style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
* ),
* Text(
* \_displayText, // Hiển thị giá trị result
* style: const TextStyle(fontSize: 20, color: Colors.red),
* ),
* ElevatedButton(
* onPressed: () {
* \_navigateToScreenB(
* context); // Navigate to the second screen when tapped.
* },
* child: const Text('Pick an option, any option!'),
* ),
* ],
* ),
* );
* }
* \_navigateToScreenB(BuildContext context) async {
* // Navigator.push returns a Future that completes after calling
* // Navigator.pop on the Selection Screen.
* final result = await Navigator.push(
* context,
* MaterialPageRoute(builder: (context) => const ScreenB()),
* );
* // Xử lý dữ liệu trả về từ Màn hình chi tiết
* if (result != null) {
* setState(() {
* \_displayText = result;
* });
* }
* }
* }
* class ScreenB extends StatelessWidget {
* const ScreenB({super.key});
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* appBar: AppBar(
* title: const Text('Pick an option'),
* ),
* body: Center(
* child: Column(
* mainAxisAlignment: MainAxisAlignment.center,
* children: <Widget>[
* Padding(
* padding: const EdgeInsets.all(8),
* child: ElevatedButton(
* onPressed: () {
* // Close the screen and return 'ABC!' as the result.
* Navigator.pop(context,
* 'ABC!'); // void pop<T extends Object?>(BuildContext context, [T? result])
* },
* child: const Text('ABC!'),
* ),
* ),
* Padding(
* padding: const EdgeInsets.all(8),
* child: ElevatedButton(
* onPressed: () {
* // Close the screen and return '123.' as the result.
* Navigator.pop(context,
* '123.'); // void pop<T extends Object?>(BuildContext context, [T? result])
* },
* child: const Text('123.'),
* ),
* )
* ],
* ),
* ),
* );
* }
* }
* Qua phương thức Navigator.pushName với tham số arguments

// Khai báo routes, truyền tham số qua route

class Navigator5 extends StatelessWidget {

* const Navigator5({super.key});
* @override
* Widget build(BuildContext context) {
* return MaterialApp(
* onGenerateRoute: (settings) {
* // {Route<dynamic>? Function(RouteSettings)? onGenerateRoute} .Settings chứa thông tin về route
* /// settings có 2 thuộc tính: const RouteSettings({String? name,Object? arguments})
* // Kiểm tra tên route và trả về màn hình tương ứng
* if (settings.name == '/') {
* return MaterialPageRoute(builder: (context) {
* return const HomeScreen5();
* });
* } else if (settings.name == '/profile') {
* // Lấy thông tin người dùng từ settings.arguments ( Navigator.pushNamed(context, '/profile', arguments: 'user123');)
* final userId = settings.arguments as String;
* return MaterialPageRoute(builder: (context) {
* return UserProfilePage(
* userId:
* userId); // Truyền thông tin người dùng vào màn hình UserProfilePage
* });
* . . . . .

// Điều hướng màn hình với Navigator.pushNamed

 ElevatedButton(

              onPressed: () {

                // Điều hướng đến route '/profile' truyền theo thông tin người dùng

                Navigator.pushNamed(context, '/profile', arguments: 'user123');

                /// Future<T?> pushNamed<T extends Object?>( BuildContext context, String routeName, {Object? arguments,})

              },

. . . . .

// Nhận tham số tại màn hình đích

class UserProfilePage extends StatelessWidget {

  final String userId;

  const UserProfilePage({super.key, required this.userId});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('User Profile'),

      ),

      body: Center(

        child: Text(

            'User ID: $userId'), // Hiển thị thông tin người dùng được truyền từ route

      ),

    );

  }

}

### Get route name trên giao diện:

Dùng ModalRoute.of(context) get được route name cho cả hai cách Navigator.push và .pushName. Cụ thể:

* Dùng Navigator.push: truyền route name ngay trong phương thức .push (ko liên quan MaterialApp.router và onGenerateRoute)
* Dùng Navigator.pushNamed: ngoài route truyền vào phương thức .pushName khi gọi, chỉ cần truyển route name trong onGenerateRoute

(Nói cách khác dùng .push phải truyền theo RouteSettings, dùng .pushNamed chỉ cần truyền RouteSettings trong onGenerateRoute thì sẽ get được route name bằng ModalRoute.of(context)?.settings.name)

* Truyền tham số routeName qua phương thức Navigator.push hoặc .pushNamed

ElevatedButton(

* onPressed: () {
* // Chuyển đến màn hình ProfileScreen
* Navigator.push(
* context,
* MaterialPageRoute(
* builder: (context) => const ProfileScreen(),
* settings: const RouteSettings(
* name:
* '/profile'), // bắt buộc truyền theo route name ở đây, để UI có thể get ra bằng ModalRoute.of(context)?.settings.name;
* ),
* );
* },
* child: const Text(
* 'Go to Profile - .push với MaterialPageRoute có settings'),
* ),

// OR

 ElevatedButton(

              onPressed: () {

                // Chuyển đến màn hình ProfileScreen

                Navigator.pushNamed(context,

                    '/profile'); // Nếu đi qua onGenerateRoute

Phải truyền route name, còn đi qua MaterialApp.routes ko cần truyền

              },

              child: const Text('Go to Profile - Navigator.pushNamed'),

            ),

* Nhận tại UI bằng phương thức ModalRoute.of(context)?.settings.name
* class ProfileScreen extends StatelessWidget {
* const ProfileScreen({super.key});
* @override
* Widget build(BuildContext context) {
* // Lấy tên của route từ settings
* final routeName = ModalRoute.of(context)?.settings.name;
* return Scaffold(
* appBar: AppBar(
* title: const Text('Profile'),
* ),
* body: Center(
* child: Column(
* mainAxisAlignment: MainAxisAlignment.center,
* children: [
* Text('Route Name: $routeName'), // Hiển thị tên của route
* const SizedBox(height: 20),
* ElevatedButton(
* onPressed: () {
* // Chuyển trở lại màn hình trước đó
* Navigator.pop(context);
* },
* child: const Text('Go Back'),
* ),
* ],
* ),
* ),
* );
* }
* }

https://docs.flutter.dev/cookbook/navigation/returning-data

Nên tổ chức routes tập trung vào một class để dễ quản lý, bảo trì

Example 1

class Routes {

  static const String screen1 = "/screen1";

  static const String screen2 = "/screen2";

}

class Navigator4 extends StatelessWidget {

  const Navigator4({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(routes: {

      '/': (context) => const NewWidget(), // route default khi start app

      Routes.screen1: (context) => const Screen1(),

      Routes.screen2: (context) => const Screen2(),

    });

  }

}

class NewWidget extends StatelessWidget {

  const NewWidget({

    super.key,

  });

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: const Text(' Home Screen '),

        ),

        body: Center(

          child: Column(

            children: [

              ElevatedButton(

                style: ElevatedButton.styleFrom(foregroundColor: Colors.amber),

                onPressed: () {

                  Navigator.pushNamed(context, Routes.screen1);

                },

                child: const Text('Goto Screen1'),

              ),

              const SizedBox(height: 10),

              ElevatedButton(

                style: ElevatedButton.styleFrom(foregroundColor: Colors.amber),

                onPressed: () {

                  Navigator.pushNamed(context, Routes.screen2);

                },

                child: const Text('Goto Screen2'),

              ),

            ],

          ),

        ));

  }

}

Example 2

class Routes2 {

  static const String home = "/home";

  static const String screen1 = "/screen1";

  static const String screen2 = "/screen2";

  static const String screen3 = "/screen3";

}

class Navigator1 extends StatelessWidget {

  const Navigator1({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      initialRoute: Routes2.home,

      home: const Scaffold(

          // body: Home(),

          ),

      onGenerateRoute: (settings) {

        switch (settings.name) {

          case Routes2.home:

            return MaterialPageRoute(

                // làm sao truyền route tại đây? (hiện tại truyền tại đay ko get được ở UI)

                builder: (\_) => const Home());

          case Routes2.screen1:

            return MaterialPageRoute(

                // làm sao truyền route tại đây? (hiện tại truyền tại đay ko get được ở UI)

                builder: (\_) => const Screen1());

          case Routes2.screen2:

            return MaterialPageRoute(

                // làm sao truyền route tại đây? (hiện tại truyền tại đay ko get được ở UI)

                builder: (\_) => const Screen22(items: []));

          case Routes2.screen3:

            return MaterialPageRoute(

                builder: (\_) => Screen3(data: settings.arguments.toString()));

          default:

            return MaterialPageRoute(builder: (context) => const Screen1());

        }

      },

    );

  }

}

class Home extends StatelessWidget {

  const Home({super.key});

  @override

  Widget build(BuildContext context) {

    List<String> itemList = ['Mục 1', 'Mục 2', 'Mục 3', 'Mục 4'];

    return Scaffold(

      appBar: AppBar(title: const Text('Home')),

      body: Padding(

        padding: const EdgeInsets.all(16),

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            ElevatedButton(

                onPressed: () {

                  Navigator.push(

                    context,

                    MaterialPageRoute(

                        builder: (context) => const Screen11(),

                        settings: const RouteSettings(name: Routes2.screen1)),

                  );

                },

                child: const Text('Screen1')),

            const SizedBox(

              height: 20,

            ),

            ElevatedButton(

              onPressed: () {

                Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => Screen22(items: itemList),

                      settings: const RouteSettings(name: Routes2.screen2)),

                );

              },

              child: const Text('Screen2'),

            ),

            const SizedBox(

              height: 20,

            ),

            ElevatedButton(

              onPressed: () {

                Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => const Screen3(

                            data: '123',

                          ),

                      settings: const RouteSettings(name: Routes2.screen3)),

                );

              },

              child: const Text('Screen3'),

            ),

          ],

        ),

      ),

    );

  }

}

### Phương thức .push và .pushReplacement

Dưới đây là giải thích chi tiết về sự khác biệt giữa chúng và cách chúng ảnh hưởng đến ngăn xếp chuyển hướng:

Navigator.push:

* Khi bạn sử dụng .push để chuyển hướng, một trang mới sẽ được thêm vào ngăn xếp chuyển hướng.
* Trang hiện tại vẫn giữ nguyên trong ngăn xếp và trở thành trang trước đó của trang mới.
* Khi người dùng nhấn nút "Back", ứng dụng sẽ trở lại trang trước đó trong ngăn xếp chuyển hướng.

Navigator.pushReplacement:

* Khi bạn sử dụng .pushReplacement, trang hiện tại sẽ được thay thế bằng trang mới trong ngăn xếp chuyển hướng.
* Trang hiện tại sẽ không còn tồn tại trong ngăn xếp và sẽ bị loại bỏ.

Với .push, mỗi lần bạn thực hiện chuyển hướng, một trang mới được thêm vào ngăn xếp và trang hiện tại trở thành trang trước đó của trang mới.

Với .pushReplacement, trang hiện tại bị loại bỏ khỏi ngăn xếp và thay thế bằng trang mới. Người dùng không thể quay lại trang hiện tại từ trang mới bằng cách sử dụng nút "Back".

## Title của MaterialApp khác với Title của AppBar thế nào

Trong Flutter, title của MaterialApp và title của AppBar có mục đích và vị trí sử dụng khác nhau:

* Title của MaterialApp:

Thuộc tính title của MaterialApp đặt tiêu đề của toàn bộ ứng dụng. Điều này thường được sử dụng để xác định tiêu đề của ứng dụng trên thanh tiêu đề của hệ điều hành hoặc trong danh sách các ứng dụng đang chạy.

Được đặt trong phần khởi tạo của MaterialApp.

* Title của AppBar

Thuộc tính title của AppBar đặt tiêu đề cho thanh tiêu đề nằm trong AppBar. Thường được sử dụng để hiển thị một tiêu đề đặc biệt cho từng màn hình hoặc trạng thái của ứng dụng.

Được đặt trong phần body của AppBar.

Dưới đây là một ví dụ để thấy sự khác biệt giữa hai thuộc tính này:

void main() {

  runApp(MaterialApp(

    title: 'My Awesome App', // Chưa rõ sẽ show lên trong trường hợp nào?

    home: Scaffold(

      appBar: AppBar(

        title: Text('Screen Title'), // Tiêu đề của ứng dụng

      ),

      body: TextWidget(),

    ),

  ));

}

## Widget có khả năng chứa nhiều widget con

Một số widget có khả năng chứa nhiều widget con như Column, Row, Stack, **ListView**,TextSpan,…

Example 1

child: Column(

            mainAxisAlignment: MainAxisAlignment.center,

            children: [

              PaddingAndMargin\_01(),

              PaddingAndMargin\_02(),

            ],

          ),

## Thuộc tính **children**

Thuộc tính children là một thuộc tính phổ biến trong các widget trong Flutter, đặc biệt là các widget nhóm như Column, Row, ListView, Stack và nhiều widget khác. Thuộc tính này cho phép bạn cung cấp một danh sách các widget con mà bạn muốn hiển thị hoặc xử lý trong một widget cha.

Cú pháp cơ bản của thuộc tính children là:

List<Widget> children = [

// Danh sách các widget con

Widget1(),

Widget2(),

// ...

];

Một số ví dụ triển khai

Column(

  children: [

    Text('Widget 1'),

    Text('Widget 2'),

    Text('Widget 3'),

  ],

)

//--------------------

Row(

  children: [

    Icon(Icons.star),

    Icon(Icons.star),

    Icon(Icons.star),

  ],

)

//--------------------

ListView(

  children: [

    ListTile(title: Text('Item 1')),

    ListTile(title: Text('Item 2')),

    ListTile(title: Text('Item 3')),

  ],

)

## Listener

const Listener(

{[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key,

[PointerDownEventListener](https://api.flutter.dev/flutter/rendering/PointerDownEventListener.html)? onPointerDown,

[PointerMoveEventListener](https://api.flutter.dev/flutter/rendering/PointerMoveEventListener.html)? onPointerMove,

[PointerUpEventListener](https://api.flutter.dev/flutter/rendering/PointerUpEventListener.html)? onPointerUp,

[PointerHoverEventListener](https://api.flutter.dev/flutter/services/PointerHoverEventListener.html)? onPointerHover,

[PointerCancelEventListener](https://api.flutter.dev/flutter/rendering/PointerCancelEventListener.html)? onPointerCancel,

[PointerPanZoomStartEventListener](https://api.flutter.dev/flutter/rendering/PointerPanZoomStartEventListener.html)? onPointerPanZoomStart,

[PointerPanZoomUpdateEventListener](https://api.flutter.dev/flutter/rendering/PointerPanZoomUpdateEventListener.html)? onPointerPanZoomUpdate,

[PointerPanZoomEndEventListener](https://api.flutter.dev/flutter/rendering/PointerPanZoomEndEventListener.html)? onPointerPanZoomEnd,

[PointerSignalEventListener](https://api.flutter.dev/flutter/rendering/PointerSignalEventListener.html)? onPointerSignal,

[HitTestBehavior](https://api.flutter.dev/flutter/rendering/HitTestBehavior.html) behavior = HitTestBehavior.deferToChild,

[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child}

)

A widget that calls callbacks in response to common pointer events.

It listens to events that can construct gestures, such as when the pointer is pressed, moved, then released or canceled.

It does not listen to events that are exclusive to mouse, such as when the mouse enters, exits or hovers a region without pressing any buttons. For these events, use MouseRegion.

Rather than listening for raw pointer events, consider listening for higher-level gestures using GestureDetector.

\*Lỗi ko set child - A RenderFlex overflowed by Infinity pixels on the bottom. (phải set child mới hết lỗi)

class Listener01 extends StatefulWidget {

  const Listener01({super.key});

  @override

  State<Listener01> createState() => \_Listener01State();

}

class \_Listener01State extends State<Listener01> {

  int \_downCounter = 0;

  int \_upCounter = 0;

  double x = 0.0;

  double y = 0.0;

  void \_incrementDown(PointerEvent details) {

    \_updateLocation(details);

    setState(() {

      \_downCounter++;

    });

  }

  void \_incrementUp(PointerEvent details) {

    \_updateLocation(details);

    setState(() {

      \_upCounter++;

    });

  }

  void \_updateLocation(PointerEvent details) {

    setState(() {

      x = details.position.dx;

      y = details.position.dy;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Listener(

        onPointerDown: \_incrementDown,

        onPointerMove: \_updateLocation,

        onPointerUp: \_incrementUp,

        child: Container(

          width: 400,

          height: 400,

          color: Colors.blue,

          child: ColoredBox(

            color: Colors.yellowAccent,

            child: Column(

              mainAxisAlignment: MainAxisAlignment.center,

              children: <Widget>[

                const Text('Tap, drag, or move your pointer here!'),

                Text(

                  '$\_downCounter presses\n$\_upCounter releases',

                  style: Theme.of(context).textTheme.headlineMedium,

                ),

                Text(

                  'The cursor is here: (${x.toStringAsFixed(2)}, ${y.toStringAsFixed(2)})',

                ),

              ],

            ),

          ),

        ));

  }

}

//----------------------------------------------------------------

class Listener01A extends StatefulWidget {

  const Listener01A({super.key});

  @override

  State<Listener01A> createState() => \_Listener01AState();

}

class \_Listener01AState extends State<Listener01A> {

  int \_downCounter = 0;

  int \_upCounter = 0;

  double x = 0.0;

  double y = 0.0;

  void \_incrementDown(PointerEvent details) {

    \_updateLocation(details);

    setState(() {

      \_downCounter++;

    });

  }

  void \_incrementUp(PointerEvent details) {

    \_updateLocation(details);

    setState(() {

      \_upCounter++;

    });

  }

  void \_updateLocation(PointerEvent details) {

    setState(() {

      x = details.position.dx;

      y = details.position.dy;

    });

  }

  @override

  Widget build(BuildContext context) {

    return ColoredBox(

      color: Colors.yellowAccent,

      child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        crossAxisAlignment: CrossAxisAlignment.center,

        children: <Widget>[

          const Text('Tap, drag, or move your pointer here!'),

          Text(

            '$\_downCounter presses\n$\_upCounter releases',

            style: Theme.of(context).textTheme.headlineMedium,

          ),

          Text(

            'The cursor is here: (${x.toStringAsFixed(2)}, ${y.toStringAsFixed(2)})',

          ),

          Listener(

            onPointerDown: \_incrementDown,

            onPointerMove: \_updateLocation,

            onPointerUp: \_incrementUp,

            child: Container(

              width: 100,

              height: 200,

              color: Colors.green,

            ),

          )

        ],

      ),

    );

  }

}

//---------------------------

class Listener02 extends StatefulWidget {

  const Listener02({super.key});

  @override

  State<Listener02> createState() => \_Listener02State();

}

class \_Listener02State extends State<Listener02> {

  int \_downCounter = 0;

  int \_upCounter = 0;

  double x = 0.0;

  double y = 0.0;

  void \_incrementDown(PointerEvent details) {

    \_updateLocation(details);

    setState(() {

      \_downCounter++;

    });

  }

  void \_incrementUp(PointerEvent details) {

    \_updateLocation(details);

    setState(() {

      \_upCounter++;

    });

  }

  void \_updateLocation(PointerEvent details) {

    setState(() {

      x = details.position.dx;

      y = details.position.dy;

    });

  }

  @override

  Widget build(BuildContext context) {

    return ConstrainedBox(

      constraints: BoxConstraints.tight(const Size(300.0, 300.0)),

      child: Listener(

        onPointerDown: \_incrementDown,

        onPointerMove: \_updateLocation,

        onPointerUp: \_incrementUp,

        child: ColoredBox(

          color: Colors.lightBlueAccent,

          child: Column(

            mainAxisAlignment: MainAxisAlignment.center,

            children: <Widget>[

              const Text(

                  'You have pressed or released in this area this many times:'),

              Text(

                '$\_downCounter presses\n$\_upCounter releases',

                style: Theme.of(context).textTheme.headlineMedium,

              ),

              Text(

                'The cursor is here: (${x.toStringAsFixed(2)}, ${y.toStringAsFixed(2)})',

              ),

            ],

          ),

        ),

      ),

    );

  }

}

## Gestures

Flutter widget hỗ trợ tương tác thông qua một widget đặc biệt gọi là GestureDetector. GestureDetector không hiển thị trên giao diện nhưng có khả năng nắm bắt các thao tác của người dùng như nhấp, kéo, vuốt, chạm.... Phần lớn widget gốc của Flutter hỗ trợ tương tác giao diện thông qua GestureDetector.

If this widget has a child, it defers to that child for its sizing behavior. If it does not have a child, it grows to fit the parent instead.

By default a GestureDetector with an invisible child ignores touches; this behavior can be controlled with behavior.

GestureDetector also listens for accessibility events and maps them to the callbacks. To ignore accessibility events, set excludeFromSemantics to true.

See flutter.dev/gestures/ for additional information.

Material design applications typically react to touches with ink splash effects. The InkWell class implements this effect and can be used in place of a GestureDetector for handling taps.

// GestureDetector set child property

class Gesture01 extends StatefulWidget {

  const Gesture01({super.key});

  @override

  State<Gesture01> createState() => \_Gesture01State();

}

class \_Gesture01State extends State<Gesture01> {

  bool \_lightIsOn = false;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Container(

        alignment: FractionalOffset.center,

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: <Widget>[

            Padding(

              padding: const EdgeInsets.all(8.0),

              child: Icon(

                Icons.lightbulb\_outline,

                color: \_lightIsOn ? Colors.yellow.shade600 : Colors.black,

                size: 60,

              ),

            ),

            GestureDetector(

              onTap: () {

                setState(() {

                  // Toggle light when tapped.

                  \_lightIsOn = !\_lightIsOn;

                });

              },

              child: Container(

                color: const Color.fromARGB(255, 172, 171, 161),

                padding: const EdgeInsets.all(8),

                // Change button text when light changes state.

                child: Text(\_lightIsOn ? 'TURN LIGHT OFF' : 'TURN LIGHT ON'),

              ),

            ),

          ],

        ),

      ),

    );

  }

}

//---------------------------

// GestureDetector not set child property

/// GestureDetector does not have a child, it takes on the size of its parent, making the entire area of the surrounding Container clickable.

class Gesture02 extends StatefulWidget {

  const Gesture02({super.key});

  @override

  State<Gesture02> createState() => \_Gesture02State();

}

class \_Gesture02State extends State<Gesture02> {

  Color \_color = Colors.white;

  @override

  Widget build(BuildContext context) {

    return Container(

      // color: \_color,

      height: 400,

      width: 400.0,

      decoration: BoxDecoration(

          color: \_color, border: Border.all(width: 2, color: Colors.black)),

      child: GestureDetector(

        onTap: () {

          setState(() {

            \_color == Colors.yellow

                ? \_color = Colors.white

                : \_color = Colors.yellow;

          });

        },

      ),

    );

  }

}

//----------------------------------------------------------------

// GestureDetectors nested

//  The key concept here is the "first come, first served" principle in gesture recognition. - Child GestureDetectors action is parent GestureDetectors not action

class Gesture03 extends StatefulWidget {

  const Gesture03({super.key});

  @override

  State<Gesture03> createState() => \_Gesture03State();

}

class \_Gesture03State extends State<Gesture03> {

  void \_handleParentTap() {

    print('Parent GestureDetector tapped!');

  }

  void \_handleChildTap() {

    print('Child GestureDetector tapped!');

  }

  @override

  Widget build(BuildContext context) {

    return GestureDetector(

      onTap: \_handleParentTap, // Parent GestureDetector onTap callback

      child: Container(

        width: 200,

        height: 200,

        color: Colors.blue,

        child: GestureDetector(

          onTap: \_handleChildTap, // Child GestureDetector onTap callback

          child: const Center(

            child: Text(

              'Tap me!',

              style: TextStyle(color: Colors.white),

            ),

          ),

        ),

      ),

    );

  }

}

//

// Gesture04

enum \_OnTapWinner { none, yellow, green }

class Gesture04 extends StatefulWidget {

  const Gesture04({super.key});

  @override

  State<Gesture04> createState() => \_Gesture04State();

}

class \_Gesture04State extends State<Gesture04> {

  bool \_isYellowTranslucent = false;

  \_OnTapWinner \_winner = \_OnTapWinner.none;

  final Border highlightBorder =

      Border.all(color: Colors.red, width: 5); // define Border object

  @override

  Widget build(BuildContext context) {

    return Column(

      children: <Widget>[

        Expanded(

          child: GestureDetector(

            onTap: () {

              debugPrint('Green onTap');

              setState(() {

                \_winner = \_OnTapWinner.green;

              });

            },

            onTapDown: (\_) => debugPrint('Green onTapDown'),

            onTapCancel: () => debugPrint('Green onTapCancel'),

            child: Container(

              alignment: Alignment.center,

              decoration: BoxDecoration(

                border: \_winner == \_OnTapWinner.green ? highlightBorder : null,

                color: Colors.green,

              ),

              child: GestureDetector(

                // Setting behavior to transparent or opaque as no impact on

                // parent-child hit testing. A tap on 'Yellow' is also in

                // 'Green' bounds. Both enter the gesture arena, 'Yellow' wins

                // because it is in front.

                behavior: \_isYellowTranslucent

                    ? HitTestBehavior.translucent

                    : HitTestBehavior.opaque,

                onTap: () {

                  debugPrint('Yellow onTap');

                  setState(() {

                    \_winner = \_OnTapWinner.yellow;

                  });

                },

                child: Container(

                  alignment: Alignment.center,

                  decoration: BoxDecoration(

                    border:

                        \_winner == \_OnTapWinner.yellow ? highlightBorder : null,

                    color: Colors.amber,

                  ),

                  width: 200,

                  height: 200,

                  child: Text(

                    'HitTextBehavior.${\_isYellowTranslucent ? 'translucent' : 'opaque'}',

                    textAlign: TextAlign.center,

                  ),

                ),

              ),

            ),

          ),

        ),

        Padding(

          padding: const EdgeInsets.all(8.0),

          child: Row(

            children: <Widget>[

              ElevatedButton(

                child: const Text('Reset'),

                onPressed: () {

                  setState(() {

                    \_isYellowTranslucent = false;

                    \_winner = \_OnTapWinner.none;

                  });

                },

              ),

              const SizedBox(width: 8),

              ElevatedButton(

                child: Text(

                  'Set Yellow behavior to ${\_isYellowTranslucent ? 'opaque' : 'translucent'}',

                ),

                onPressed: () {

                  setState(() => \_isYellowTranslucent = !\_isYellowTranslucent);

                },

              ),

            ],

          ),

        ),

      ],

    );

  }

  @override

  void dispose() {

    debugPrintGestureArenaDiagnostics = false;

    super.dispose();

  }

}

// Gesture05

class Gesture05 extends StatefulWidget {

  const Gesture05({super.key});

  @override

  State<Gesture05> createState() => \_Gesture05State();

}

class \_Gesture05State extends State<Gesture05> {

  bool \_isYellowTranslucent = false;

  \_OnTapWinner \_winner = \_OnTapWinner.none;

  String \_tapInfo = '';

  String \_hitTestBehaviorInfo = '';

  final Border highlightBorder = Border.all(color: Colors.red, width: 5);

  @override

  Widget build(BuildContext context) {

    return Column(

      children: <Widget>[

        Expanded(

          child: GestureDetector(

            onTap: () {

              setState(() {

                \_winner = \_OnTapWinner.green;

                \_tapInfo = 'Green tapped!';

              });

            },

            onTapDown: (\_) {

              setState(() {

                \_tapInfo = 'Green onTapDown';

              });

            },

            onTapCancel: () {

              setState(() {

                // để giữ value của biến gán trong GestureDetector con thì phải ko gán tại GestureDetector cha(gán tại GestureDetector luôn ghi đè lại dù HitTestBehavior có là giá trị nào)

                // \_tapInfo = 'Green onTapCancel';

              });

            },

            child: Container(

              alignment: Alignment.center,

              decoration: BoxDecoration(

                border: \_winner == \_OnTapWinner.green ? highlightBorder : null,

                color: Colors.green,

              ),

              child: GestureDetector(

                behavior: \_isYellowTranslucent

                    ? HitTestBehavior.translucent

                    : HitTestBehavior.opaque,

                onTap: () {

                  setState(() {

                    \_winner = \_OnTapWinner.yellow;

                    \_tapInfo = 'Yellow tapped!';

                  });

                },

                onTapDown: (\_) {

                  setState(() {

                    \_tapInfo = 'Yellow onTapDown';

                  });

                },

                onTapCancel: () {

                  setState(() {

                    \_tapInfo = 'Yellow onTapCancel';

                  });

                },

                child: Container(

                  alignment: Alignment.center,

                  decoration: BoxDecoration(

                    border:

                        \_winner == \_OnTapWinner.yellow ? highlightBorder : null,

                    color: Colors.amber,

                  ),

                  width: 250,

                  height: 200,

                  child: Column(

                    mainAxisAlignment: MainAxisAlignment.center,

                    children: [

                      Text(

                        'Tap Info: $\_tapInfo',

                        style: const TextStyle(fontSize: 16),

                      ),

                      Text(

                        'Winner: $\_winner',

                        style: const TextStyle(fontSize: 16),

                      ),

                      Text(

                        'HitTestBehavior: ${\_isYellowTranslucent ? 'translucent' : 'opaque'}',

                        style: const TextStyle(fontSize: 16),

                      ),

                    ],

                  ),

                ),

              ),

            ),

          ),

        ),

        Padding(

          padding: const EdgeInsets.all(8.0),

          child: Row(

            children: <Widget>[

              ElevatedButton(

                child: const Text('Reset'),

                onPressed: () {

                  setState(() {

                    \_isYellowTranslucent = false;

                    \_winner = \_OnTapWinner.none;

                    \_tapInfo = '';

                  });

                },

              ),

              const SizedBox(width: 8),

              ElevatedButton(

                child: Text(

                  'Set Yellow behavior to ${\_isYellowTranslucent ? 'opaque' : 'translucent'}',

                ),

                onPressed: () {

                  setState(() {

                    \_isYellowTranslucent = !\_isYellowTranslucent;

                    \_hitTestBehaviorInfo =

                        'HitTestBehavior: ${\_isYellowTranslucent ? 'translucent' : 'opaque'}';

                  });

                },

              ),

            ],

          ),

        ),

      ],

    );

  }

}

## Dismissible

A widget that can be dismissed by dragging in the indicated direction.

Dragging or flinging this widget in the DismissDirection causes the child to slide out of view. Following the slide animation, if resizeDuration is non-null, the Dismissible widget animates its height (or width, whichever is perpendicular to the dismiss direction) to zero over the resizeDuration.

[Dismissible](https://api.flutter.dev/flutter/widgets/Dismissible/Dismissible.html)({required [Key](https://api.flutter.dev/flutter/foundation/Key-class.html) key, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? background, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? secondaryBackground, [ConfirmDismissCallback](https://api.flutter.dev/flutter/widgets/ConfirmDismissCallback.html)? confirmDismiss, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onResize, [DismissUpdateCallback](https://api.flutter.dev/flutter/widgets/DismissUpdateCallback.html)? onUpdate, [DismissDirectionCallback](https://api.flutter.dev/flutter/widgets/DismissDirectionCallback.html)? onDismissed, [DismissDirection](https://api.flutter.dev/flutter/widgets/DismissDirection.html) direction = DismissDirection.horizontal, [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html)? resizeDuration = const Duration(milliseconds: 300), [Map](https://api.flutter.dev/flutter/dart-core/Map-class.html)<[DismissDirection](https://api.flutter.dev/flutter/widgets/DismissDirection.html), [double](https://api.flutter.dev/flutter/dart-core/double-class.html)> dismissThresholds = const <DismissDirection, double>{}, [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html) movementDuration = const Duration(milliseconds: 200), [double](https://api.flutter.dev/flutter/dart-core/double-class.html) crossAxisEndOffset = 0.0, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [HitTestBehavior](https://api.flutter.dev/flutter/rendering/HitTestBehavior.html) behavior = HitTestBehavior.opaque})

Creates a widget that can be dismissed.

Example

// Example 1 - Delete item from list

class Dismissible01 extends StatefulWidget {

  const Dismissible01({super.key});

  @override

  State<Dismissible01> createState() => \_DismissibleExampleState();

}

class \_DismissibleExampleState extends State<Dismissible01> {

  List<int> items = List<int>.generate(100, (int index) => index);

  @override

  Widget build(BuildContext context) {

    return ListView.builder(

      itemCount: items.length,

      padding: const EdgeInsets.symmetric(vertical: 15),

      itemBuilder: (BuildContext context, int index) {

        return Dismissible(

          background: Container(

            // set background when gesture on items

            color: Colors.green,

          ),

          key: ValueKey<int>(items[index]),

          onDismissed: (DismissDirection direction) {

            setState(() {

              items.removeAt(index);

            });

          },

          child: ListTile(

            title: Text(

              'Item ${items[index]}',

            ),

          ),

        );

      },

    );

  }

}

// Example 2 - Delete items from list 2

class Dismissible02 extends StatelessWidget {

  const Dismissible02({super.key});

  static List<String> items = [

    "Item 1",

    "Item 2",

    "Item 3",

    "Item 4",

    "Item 5",

    "Item 6",

    "Item 7",

    "Item 8",

    "Item 9"

  ];

  @override

  Widget build(BuildContext context) {

    return ListView.builder(

      itemCount: items.length,

      itemBuilder: (context, index) {

        final item = items[index];

        return Dismissible(

          key: Key(item),

          onDismissed: (direction) {

            // Remove the item from the data source

            items.removeAt(index);

            // Show a snackbar with the item name

            ScaffoldMessenger.of(context).showSnackBar(

              SnackBar(

                content: Text('$item dismissed'),

              ),

            );

          },

          background: Container(

            color: Colors.red,

            child: const Icon(Icons.delete, color: Colors.white),

          ),

          child: ListTile(

            title: Text(item),

          ),

        );

      },

    );

  }

}

// Example 3 - Chuyển trang khi Dismiss

class Dismissible03 extends StatelessWidget {

  const Dismissible03({super.key});

  static List<String> items = [

    "Item 1",

    "Item 2",

    "Item 3",

    "Item 4",

    "Item 5",

    "Item 6",

    "Item 7",

    "Item 8",

    "Item 9"

  ];

  @override

  Widget build(BuildContext context) {

    return ListView.builder(

      itemCount: items.length,

      itemBuilder: (context, index) {

        final item = items[index];

        return Dismissible(

          key: Key(item),

          onDismissed: (direction) {

            // Remove the item from the data source

            items.removeAt(index);

            // Navigate to a new page when dismissed

            // Với .pushReplacement, trang hiện tại bị loại bỏ khỏi ngăn xếp và thay thế bằng trang mới. Người dùng không thể quay lại trang hiện tại từ trang mới bằng cách sử dụng nút "Back".

            Navigator.of(context).pushReplacement(

              MaterialPageRoute(

                builder: (BuildContext context) => ReplacementPage(item),

              ),

            );

          },

          background: Container(

            color: Colors.red,

            child: const Icon(Icons.delete, color: Colors.white),

          ),

          child: ListTile(

            title: Text(item),

          ),

        );

      },

    );

  }

}

class ReplacementPage extends StatelessWidget {

  final String itemName;

  const ReplacementPage(this.itemName, {super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        automaticallyImplyLeading: true,

        title: const Text('Replacement Page'),

      ),

      body: Center(

        child: Text('$itemName dismissed. Do something else on this page.'),

      ),

    );

  }

}

// Example 4 - Chuyển trang khi Dismiss và back to the home page

class Dismissible04 extends StatelessWidget {

  const Dismissible04({super.key});

  static List<String> items = [

    "Item 1",

    "Item 2",

    "Item 3",

    "Item 4",

    "Item 5",

    "Item 6",

    "Item 7",

    "Item 8",

    "Item 9"

  ];

  @override

  Widget build(BuildContext context) {

    return ListView.builder(

      itemCount: items.length,

      itemBuilder: (context, index) {

        final item = items[index];

        return Dismissible(

          key: Key(item),

          onDismissed: (direction) {

            // Remove the item from the data source

            items.removeAt(index);

            // Navigate to a new page when dismissed

            Navigator.of(context).push(

              MaterialPageRoute(

                builder: (BuildContext context) => ReplacementPage(item),

              ),

            );

          },

          background: Container(

            color: Colors.red,

            child: const Icon(Icons.delete, color: Colors.white),

          ),

          child: ListTile(

            title: Text(item),

          ),

        );

      },

    );

  }

}

class ReplacementPage1 extends StatelessWidget {

  final String itemName;

  const ReplacementPage1(this.itemName, {super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        automaticallyImplyLeading: true,

        title: const Text('Replacement Page'),

      ),

      body: Center(

        child: Text('$itemName dismissed. Do something else on this page.'),

      ),

    );

  }

}

## MediaQuery

Trong Flutter, **MediaQuery** là một cách để truy cập thông tin về thiết bị và môi trường màn hình hiện tại. Đối tượng **MediaQuery** chứa thông tin về kích thước màn hình, hướng (portrait hoặc landscape), tỷ lệ điểm ảnh (pixel density), vùng safe area (nơi có thể đặt nội dung mà không bị cắt), và nhiều thông tin khác.

**MediaQuery** giúp ứng dụng Flutter dễ dàng thích nghi với nhiều loại thiết bị và môi trường màn hình khác nhau, tạo ra trải nghiệm người dùng tốt hơn trên mọi nền tảng.

Một số ví dụ về cách **MediaQuery** có thể được sử dụng bao gồm:

1. **Lấy thông tin về kích thước màn hình:** Bạn có thể sử dụng **MediaQuery.of(context).size** để lấy kích thước màn hình hiện tại. Điều này có thể hữu ích để điều chỉnh giao diện của ứng dụng dựa trên kích thước màn hình.
2. **Xác định hướng màn hình:** Sử dụng **MediaQuery.of(context).orientation** để biết xem màn hình đang ở chế độ portrait (dọc) hay landscape (ngang). Điều này giúp bạn tùy chỉnh giao diện cho cả hai chế độ.
3. **Tính toán tỷ lệ điểm ảnh (pixel density):** **MediaQuery.of(context).devicePixelRatio** cho biết tỷ lệ giữa kích thước điểm ảnh và kích thước thiết bị. Điều này có thể hữu ích khi bạn cần tải hình ảnh với chất lượng phù hợp với thiết bị.
4. **Thiết lập safe area:** **MediaQuery.of(context).padding** cho biết khoảng cách an toàn xung quanh màn hình, giúp bạn đặt nội dung sao cho không bị cắt bỏ.
5. **Lấy thông tin về ngôn ngữ và hệ điều hành:** Bạn có thể sử dụng **MediaQuery.of(context).platformBrightness** để biết màu sắc chế độ sáng hoặc tối của hệ điều hành và **MediaQuery.of(context).platform** để xác định hệ điều hành (iOS, Android, hoặc khác).
6. Lệnh **MediaQuery.of(context).copyWith(...)** trong Flutter được sử dụng để tạo một bản sao mới của đối tượng **MediaQueryData** (thường được gọi là bản sao của **MediaQuery**) với một số cài đặt tùy chỉnh. Điều này cho phép bạn tạo ra một **MediaQuery** mới với các giá trị được thay đổi so với **MediaQuery** gốc, mà sau đó có thể sử dụng để ảnh hưởng đến cách giao diện của ứng dụng hoạt động.

Dưới đây là một số ví dụ về việc sử dụng **MediaQuery.of(context).copyWith(...)** để tùy chỉnh **MediaQuery**:

* **Thay đổi hướng màn hình:** Bạn có thể thay đổi hướng màn hình của **MediaQuery** bằng cách sử dụng **.copyWith(orientation: ...)** để thiết lập hướng mới, dựa trên cách bạn muốn ứng dụng phản ánh hướng màn hình hiện tại.
* **Thay đổi tỷ lệ điểm ảnh (pixel density):** Bạn có thể điều chỉnh tỷ lệ pixel density bằng cách sử dụng **.copyWith(devicePixelRatio: ...)**, điều này hữu ích khi bạn cần kiểm soát việc tải hình ảnh với chất lượng khác nhau dựa trên tỷ lệ điểm ảnh.
* **Thay đổi khoảng cách an toàn (safe area):** Bạn có thể thay đổi khoảng cách an toàn bằng **.copyWith(padding: ...)**. Điều này giúp đặt nội dung sao cho không bị cắt bỏ trên các màn hình có safe area lớn hơn.

//---------------------------

class MediaQuery01 extends StatelessWidget {

  const MediaQuery01({super.key});

  @override

  Widget build(BuildContext context) {

    // Accessing MediaQuery to get screen properties

    final mediaQuery = MediaQuery.of(context);

    // Adjusting the widget based on screen width

    final screenWidth = mediaQuery.size.width;

    final isWideScreen = screenWidth > 600;

    return Scaffold(

      appBar: AppBar(

        title: const Text('Responsive App'),

      ),

      body: Center(

        child: Text(

          isWideScreen ? 'Wide Screen Layout' : 'Narrow Screen Layout',

          style: TextStyle(fontSize: isWideScreen ? 24.0 : 16.0),

        ),

      ),

    );

  }

}

//

class MediaQuery02 extends StatelessWidget {

  const MediaQuery02({super.key});

  @override

  Widget build(BuildContext context) {

    // Accessing MediaQuery to get text scale factor

    final textScaleFactor = MediaQuery.of(context).textScaler;

    return Scaffold(

      appBar: AppBar(

        title: const Text('Text Scaling Demo'),

      ),

      body: Center(

        child: Text(

          'Hello, Text Scaling! - $textScaleFactor',

          style: TextStyle(

            fontSize: 20 \* MediaQuery.of(context).devicePixelRatio,

          ),

        ),

      ),

    );

  }

}

//

class MediaQuery03 extends StatelessWidget {

  const MediaQuery03({super.key});

  @override

  Widget build(BuildContext context) {

    // getting the size of the window

    final size = MediaQuery.of(context).size;

    final height = size.height;

    final width = size.width;

    return Scaffold(

      appBar: AppBar(

        title: const Text("Geeks For Geeks"),

        backgroundColor: Colors.green,

      ),

      body: Container(

        color: Colors.yellow,

        height: height / 2, //half of the height size

        width: width / 2, //half of the width size

      ),

    );

  }

}

//

class MediaQuery04 extends StatelessWidget {

  const MediaQuery04({super.key});

  @override

  Widget build(BuildContext context) {

    // getting the orientation of the app

    var orientation = MediaQuery.of(context).orientation;

    Orientation currentOrientation = MediaQuery.of(context).orientation;

    debugPrint('Orientation - $currentOrientation');

    //size of the window

    final size = MediaQuery.of(context).size;

    final height = size.height;

    final width = size.width;

    return Scaffold(

        appBar: AppBar(

          title: const Text("Geeks For MediaQuery04"),

          backgroundColor: Colors.green,

        ),

        // checking the orientation

        body: orientation == Orientation.portrait

            ? Container(

                color: Colors.blue,

                height: height / 2,

                width: width / 4,

                child: Text("$currentOrientation"),

              )

            : Container(

                height: height / 4,

                width: width / 2,

                color: Colors.red,

                child: Text("$currentOrientation"),

              ));

  }

}

## OrientationBuilder

[OrientationBuilder](https://api.flutter.dev/flutter/widgets/OrientationBuilder/OrientationBuilder.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [OrientationWidgetBuilder](https://api.flutter.dev/flutter/widgets/OrientationWidgetBuilder.html) builder})

Builds a widget tree that can depend on the parent widget's orientation (distinct from the device orientation).

"Distinct from the device orientation":

The orientation mentioned here is specific to the parent widget's orientation, which may not necessarily be the same as the overall device orientation. The device orientation refers to whether the device is in a portrait or landscape mode globally. However, a specific part of the UI (represented by the parent widget) might have its own orientation, especially if it's contained within a flexible or rotating container.

In summary, OrientationBuilder is a widget that helps you build a responsive widget tree within the context of its parent widget's orientation. It gives you the flexibility to adapt and customize the UI based on the local orientation of a specific part of your app, independent of the global device orientation.

class MyOrientationDemo extends StatelessWidget {

  const MyOrientationDemo({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('OrientationBuilder Example'),

      ),

      body: Center(

        child: OrientationBuilder(

          builder: (context, orientation) {

            // Build UI elements based on device orientation

            return Column(

              mainAxisAlignment: MainAxisAlignment.center,

              children: <Widget>[

                Icon(

                  // Display different icons based on orientation

                  orientation == Orientation.portrait

                      ? Icons.horizontal\_rule

                      : Icons.phone\_iphone,

                  size: 100.0,

                ),

                const SizedBox(height: 20.0),

                Text(

                  // Display different text based on orientation

                  'Device is in ${orientation == Orientation.portrait ? 'Portrait' : 'Landscape'} mode',

                  style: const TextStyle(fontSize: 20.0),

                ),

              ],

            );

          },

        ),

      ),

    );

  }

}

**OrientationBuilder** and **MediaQuery.of(context).orientation** are both used in Flutter to handle changes in device orientation, but they serve slightly different purposes.

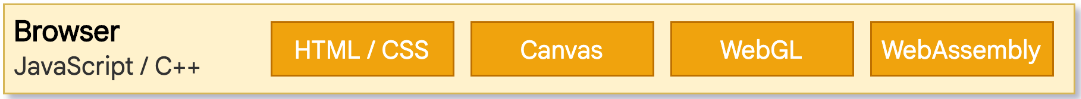
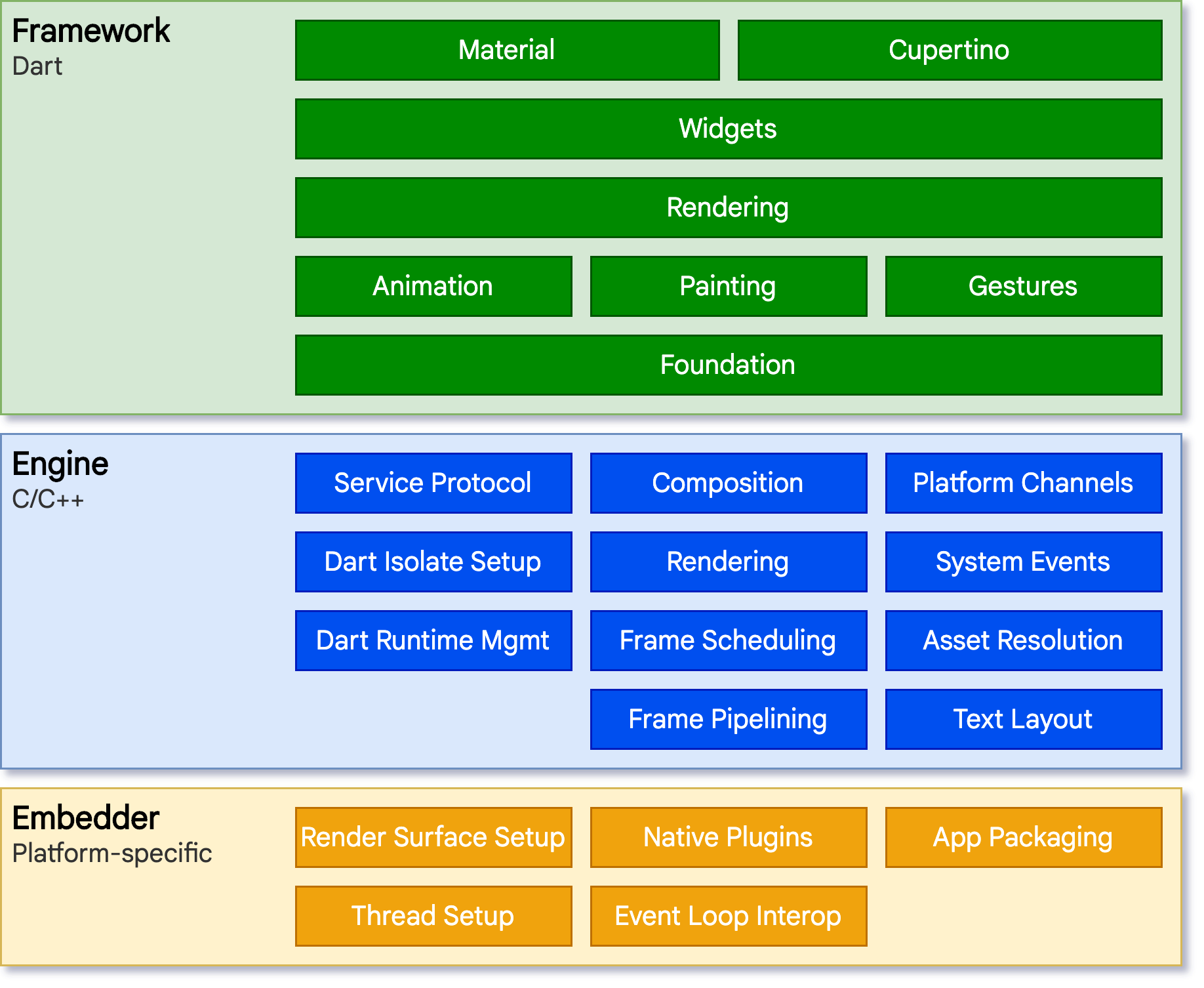
### Comparison:

* **Scope:**
  + **OrientationBuilder** is scoped to a specific part of the widget tree, allowing you to conditionally rebuild only that part.
  + **MediaQuery.of(context).orientation** can be used globally, and you can check the orientation anywhere in your code.
* **Flexibility:**
  + **OrientationBuilder** provides more flexibility in terms of building UI based on orientation, as it allows you to use the full power of the Flutter widget tree.
  + **MediaQuery.of(context).orientation** gives you a direct value and is often used for simple condition checks.
* **Use Case:**
  + Use **OrientationBuilder** when you need to conditionally build different parts of your UI based on orientation within the widget tree.
  + Use **MediaQuery.of(context).orientation** when you need to access the orientation value for global logic or conditions outside the widget tree.

In summary, while both are related to handling orientation, **OrientationBuilder** is a widget that allows for conditional building within the widget tree, while **MediaQuery.of(context).orientation** is a direct way to access the orientation value globally in your Flutter code.

## Layers

Layers are an important concept of the Flutter framework, which are grouped into multiple categories in terms of complexity and arranged in the top-down approach. The topmost layer is the UI of the application, which is specific to the Android and iOS platforms. The second topmost layer contains all the Flutter native widgets. The next layer is the rendering layer, which renders everything in the Flutter app. Then, the layers go down to Gestures, foundation library, engine, and finally, core platform-specific code. The following diagram specifies the layers in Flutter app development.



## Reference

https://docs.flutter.dev/ui/widgets

<https://docs.flutter.dev/cookbook>

<https://fonts.google.com> – fonts, icons và hướng dẫn sử dụng

## Set default size window app

Mở file windows\runner\main.cpp trong project flutter, edit values như bên dưới

Win32Window::Point origin(1000, 10); // set value vị trí góc trên-trái (lề trái, lề trên)

Win32Window::Size size(600, 800); // set value size window (width, height)

## Cài gói logger để in thông tin debug (để ko còn cảnh báo khi sử dụng lệnh print để debug)

(Ko nên dùng vì in ttin ra khá rối, nên dùng lệnh debugPrint(''))

* Edit file pubspec.yaml:
* dependencies:
* flutter:
* sdk: flutter
* logger: ^1.0.0-nullsafety.0 # ngang hàng với flutter

save để update dữ liệu.

* Sử dụng:
  + Import package: import 'package:logger/logger.dart';
  + Khai báo instance: final logger = Logger();
  + Gọi lệnh ở những vị trí cần in debug bằng 1 trong các cách sau:
* logger.d('Debug message'); // nên dùng .d

// logger.i('Info message');

// logger.w('Warning message');

// logger.e('Error message');

## Cấu hình dùng mouse cuộn màn hình app trên PC

C1: dùng ScrollConfiguration để cấu hình

// B1: Định nghĩa class cấu hình

class MyCustomScrollBehavior extends MaterialScrollBehavior {

  // Override behavior methods and getters like dragDevices

  @override

  Set<PointerDeviceKind> get dragDevices => {

        PointerDeviceKind.touch, // cảm ứng

        PointerDeviceKind.mouse, // vuốt màn hình bằng chuột. default ko có

        // etc.

      };

}

/\*

B2: Áp dụng cấu hình

Tại Widget build thêm như dưới đây

Widget build(BuildContext context) {

    return ScrollConfiguration(

      behavior: MyCustomScrollBehavior(),

child: GridView.count(……..

\*/

class GridView2 extends StatelessWidget {

  const GridView2({super.key});

  @override

  Widget build(BuildContext context) {

    return ScrollConfiguration(

      behavior: MyCustomScrollBehavior(),

      child: GridView.count(

        physics: const AlwaysScrollableScrollPhysics(),

        scrollDirection: Axis

            .vertical, // Axis.horizontal -> crossAxisCount là số row;  Axis.vertical -> crossAxisCount là số column

        crossAxisCount: 2, // Số cột trong grid

        children: List.generate(20, (index) {

          return Center(

            child: Container(

              width: 150,

              height: 150,

              color: Colors.blue, // Màu nền cho mỗi ô trong grid

              child: Text(

                'Item $index',

                style: const TextStyle(fontSize: 24, color: Colors.white),

              ),

            ),

          );

        }),

      ),

    );

  }

}