

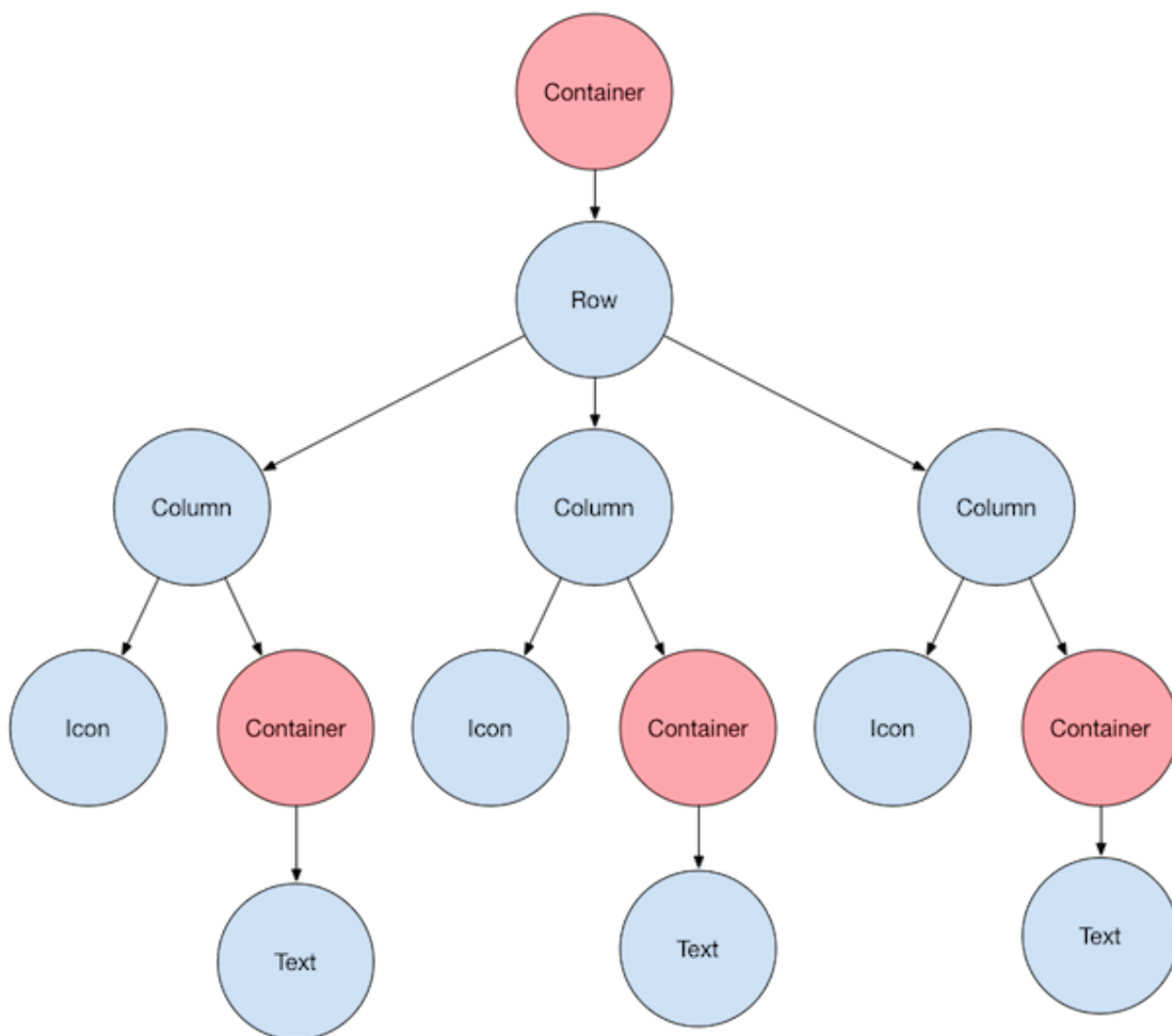
Xây dựng tư duy thiết kế giao diện

Cơ chế cốt lõi để xây dựng nên layout trong Flutter là Widget. Trong Flutter, hầu như tất cả mọi thứ đều là Widget - thậm chí là model layout cũng là Widget. Ảnh, icon, text bạn thấy trong 1 ứng dụng Flutter tất cả đều là widget. Nhưng những thứ bạn không thấy cũng là widget, ví dụ như rows, columns, và grid,... những widget này giúp cho việc sắp xếp, ràng buộc, căn chỉnh những visible widget (những widget mà bạn thấy nhìn là thấy). (Từ đây mình sẽ gọi widget nhìn thấy được - visible widget, và widget không nhìn thấy được - invisible widget; để các bạn có thể tiện theo dõi).

Bạn có thể tạo layout bằng cách kết hợp những widget với nhau để tạo nên những widget phức tạp hơn. Ví dụ dưới đây tạo ra 3 icon với nhãn dán nằm dưới:



Dưới đây là diagram của widget tree cho UI trên:



Hầu như mọi thứ bạn thấy trên ảnh có vẻ giống như bạn hình dung đúng không? Nhưng bạn thử để ý về những hình Container (màu hồng trong ảnh) xem. Container là 1 widget class cho phép bạn có thể tùy chỉnh widget con bên trong nó. Sử dụng Container khi bạn muốn thêm padding, margin, border, màu nền...

Trong ví dụ này, mỗi Text widget được đặt bên trong 1 Container để chúng ta có thể thêm margin, cách 1 khoảng so với Icon phía trên nó. Toàn bộ Row được đặt trong Container để thêm padding xung quanh Row.

Phần còn lại của UI trong ví dụ này được điều khiển bằng properties. Ví dụ: Thay đổi màu Icon bằng cách sử dụng color property, sử dụng Text.style property để tạo Font chữ, màu, độ đậm nhạt... Column và row có những property cho phép bạn xác định cách những widget con được căn chỉnh dọc hay ngang, hay xác định độ giãn cách giữa các widget con này.

Làm thế nào để bố trí 1 widget trong Flutter

Làm thế nào để bố trí 1 widget trong Flutter? Bài viết này sẽ chỉ ra cách bạn có thể tạo và hiển thị 1 widget đơn giản và chúng ta sẽ cùng nhau code 1 ví dụ Hello world đơn giản nhất nhé!

Trong Flutter, chúng ta sẽ mất 1 vài bước để đặt text, icon, hoặc Image vào màn hình.

1. Cách lựa chọn layout widget

Việc lựa chọn layout widget từ 1 tập rất nhiều layout widget hiện tại dựa trên cách mà bạn muốn căn chỉnh hay ràng buộc visible widget, vì những đặc điểm này thường được chuyển cho widget con.

Ví dụ ta sử dụng widget Center trong trường hợp muốn căn giữa content trong nó theo cả chiều dọc và chiều ngang.

2. Tạo 1 visible widget

Ví dụ, để tạo 1 Text widget:

```
1 | Text('Hello World'),
```

Tạo 1 Image widget:

```
1 | Image.asset(  
2 | 'images/lake.jpg',  
3 | fit: BoxFit.cover,  
4 | ),
```

Tạo 1 Icon widget:

```
1 | Icon(  
2 | Icons.star,  
3 | color: Colors.red[500],  
4 | ),
```

3. Thêm visible widget vào layout widget

Tất cả những layout widget đều có những thứ sau:

- Một thuộc tính (property) child nếu nó cần 1 widget con, ví dụ: Center, Container,
- Một thuộc tính (property) children nếu nó cần 1 tập danh sách các widget con, ví dụ: Row, Column, ListView, Stack,...

Thêm Text widget vào Center widget:

```
1  const Center(  
2  child: Text('Hello World'),  
3  ),
```

4. Thêm layout widget vào trong trang (page) UI

Bản thân 1 app Flutter cũng là 1 widget, và hầu như đều có method `build()`. Khởi tạo và trả về 1 widget trong method `build()` sẽ giúp hiển thị widget.

Material apps

Đối với 1 Material app, bạn có thể sử dụng `scaffold` widget, widget này cung cấp 1 banner mặc định, màu nền page, và có API cho việc thêm drawer, snackbar, bottom sheet... Sau đó bạn có thể thêm `Center` widget trực tiếp vào `body` property để căn giữa nội dung hiển thị trong page.

```
1  class MyApp extends StatelessWidget {  
2  const MyApp({Key? key}) : super(key: key);  
3  
4  @override  
5  Widget build(BuildContext context) {  
6    return MaterialApp(  
7      title: 'Flutter layout demo',  
8      home: Scaffold(  
9        appBar: AppBar(  
10         title: const Text('Flutter layout demo'),  
11       ),  
12       body: const Center(  
13         child: Text('Hello World'),  
14       ),  
15     ),  
16   );  
17 }  
18 }
```

Chú ý: Thư viện `Material` triển khai những widget dựa theo nguyên tắc `Material Design`. Khi thiết kế UI của bạn, bạn có thể sử dụng những thư viện widget tiêu chuẩn, hoặc bạn cũng có thể sử dụng widget từ thư viện `Material`.

Non-Material apps

Đối với một non-Material app, bạn có thể thêm Center widget vào method `build()`:

```
1 class MyApp extends StatelessWidget {
2   const MyApp({Key? key}) : super(key: key);
3
4   @override
5   Widget build(BuildContext context) {
6     return Container(
7       decoration: const BoxDecoration(color: Colors.white),
8       child: const Center(
9         child: Text(
10           'Hello World',
11           textDirection: TextDirection.ltr,
12           style: TextStyle(
13             fontSize: 32,
14             color: Colors.black87,
15           ),
16         ),
17       ),
18     );
19   }
20 }
```

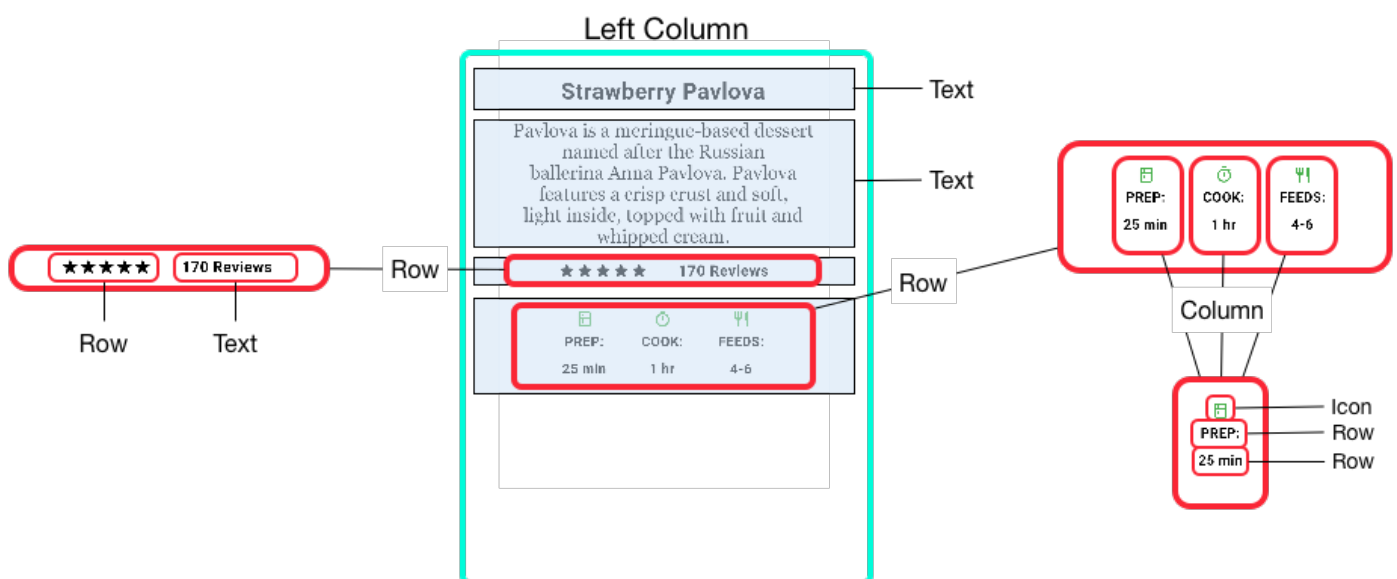
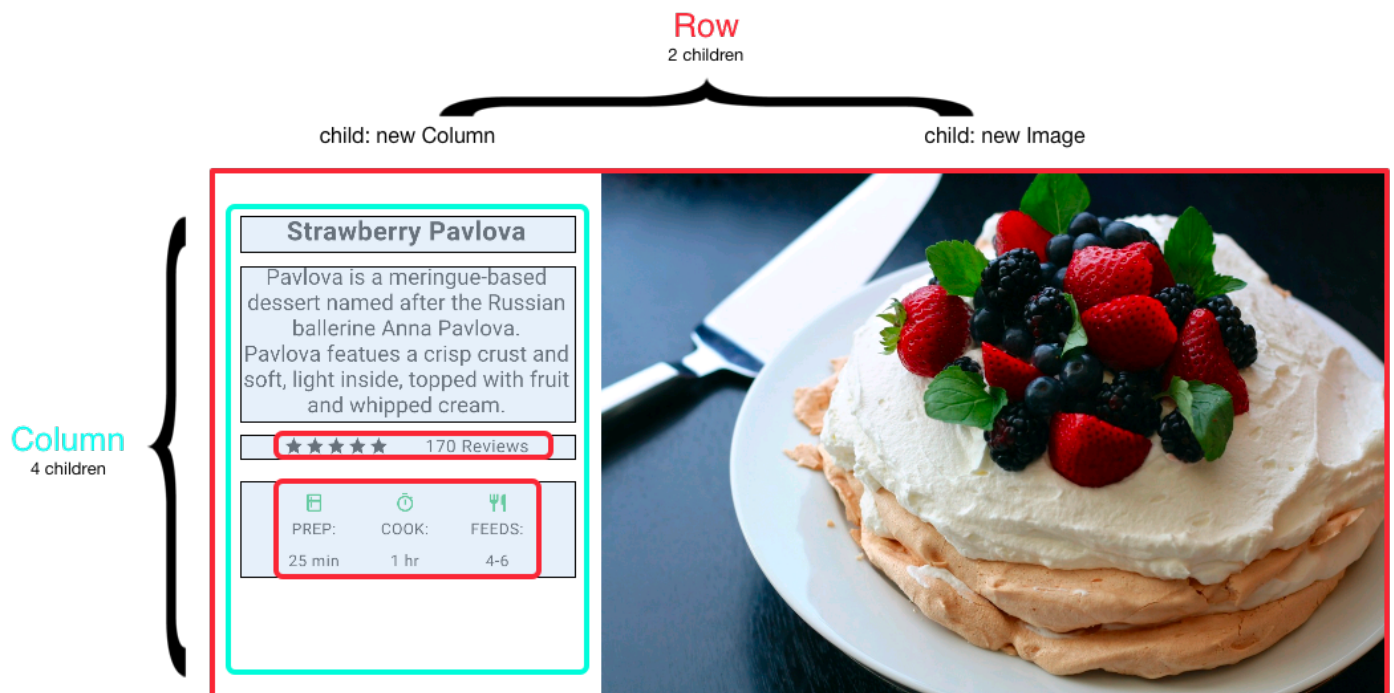
Mặc định, non-Material không bao gồm AppBar, tiêu đề hay màu nền. Nếu bạn muốn những tính năng này trong non-Material, bạn phải tự làm.

Bố trí các widget theo chiều ngang và dọc

Một trong những layout pattern phổ biến nhất là sắp xếp widget theo chiều dọc và ngang. Bạn có thể sử dụng `Row` widget để sắp xếp widget theo chiều ngang, và dùng `Column` để sắp xếp theo chiều dọc.

- `Row` và `Column` là 2 trong số những layout pattern được sử dụng phổ biến nhất.
- `Row` và `Column` cần 1 danh sách các widget con.
- Mỗi widget con có thể là `Row` hoặc `Column`, hoặc 1 widget phức tạp nào khác.
- Bạn có thể xác định cách `Row` và `Column` căn chỉnh widget con của nó, theo cả chiều ngang và dọc.
- Bạn có thể nới rộng hoặc ràng buộc widget con.
- Bạn có thể xác định cách những widget con sử dụng khoảng trống có sẵn của `Row` hoặc `Column`.

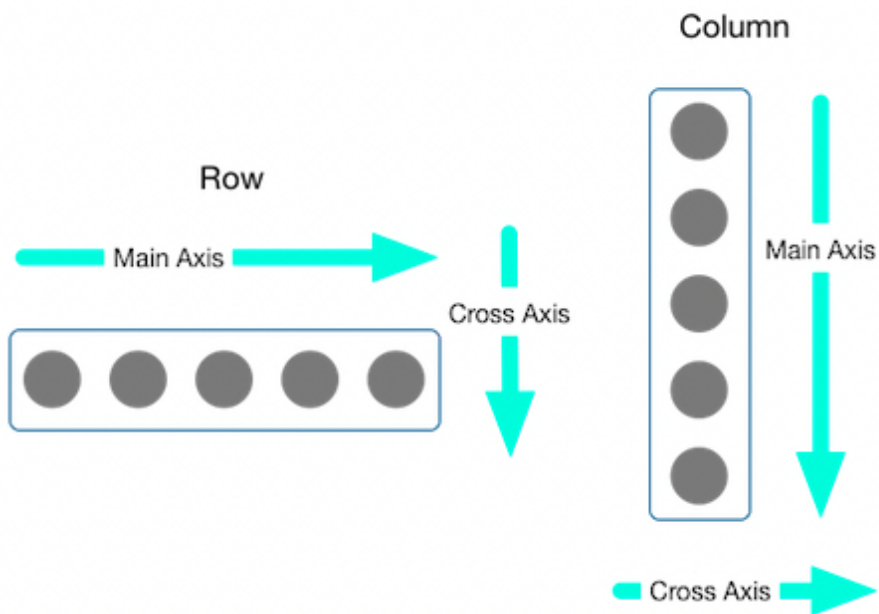
Ví dụ sau đây chỉ ra cách 1 `Row` hoặc `Column` nằm trong `Row` hoặc `Column`.



`Row` và `Column` là các widget nguyên thủy cơ bản cho bố cục ngang và dọc — những widget con cấp thấp này cho phép tùy chỉnh tối đa. Flutter cũng cung cấp các widget chuyên biệt, cấp cao hơn có thể đủ cho nhu cầu của bạn. Ví dụ: thay vì `Row`, bạn có thể thích `ListTile`, một tiện ích dễ sử dụng với các thuộc tính cho các biểu tượng đầu và cuối, và tối đa 3 dòng văn bản. Thay vì `Column`, bạn có thể thích `ListView`, một bố cục giống như cột nhưng tự động cuộn nếu nội dung của nó quá dài để phù hợp với không gian có sẵn.

Aligning widgets (căn chỉnh widget)

Bạn có thể kiểm soát cách một `Row` hoặc `Column` căn chỉnh con của nó bằng cách sử dụng các thuộc tính `mainAxisAlignment` và `crossAxisAlignment`. Đối với một `Row`, trục chính chạy theo chiều ngang và trục chéo chạy theo chiều dọc. Đối với một `Column`, trục chính chạy theo chiều dọc và trục chéo chạy theo chiều ngang. Bạn có thể nhìn hình vẽ sau để hiểu rõ hơn:

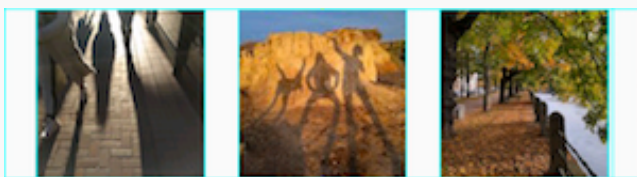


Một ví dụ khác `Row` và `Column` sử dụng với `Image`:

```

1 Row(
2   mainAxisAlignment: MainAxisAlignment.spaceEvenly,
3   children: [
4     Image.asset('images/pic1.jpg'),
5     Image.asset('images/pic2.jpg'),
6     Image.asset('images/pic3.jpg'),
7   ],
8 );


```



```


1 Column(
2   mainAxisAlignment: MainAxisAlignment.spaceEvenly,
3   children: [
4     Image.asset('images/pic1.jpg'),
5     Image.asset('images/pic2.jpg'),
6     Image.asset('images/pic3.jpg'),
7   ],
8 );

```

Sizing widgets

Khi bố cục quá lớn để vừa với một thiết bị, một hình sọc màu vàng và đen sẽ xuất hiện dọc theo cạnh bị ảnh hưởng. Đây là một ví dụ về `Row` quá rộng:

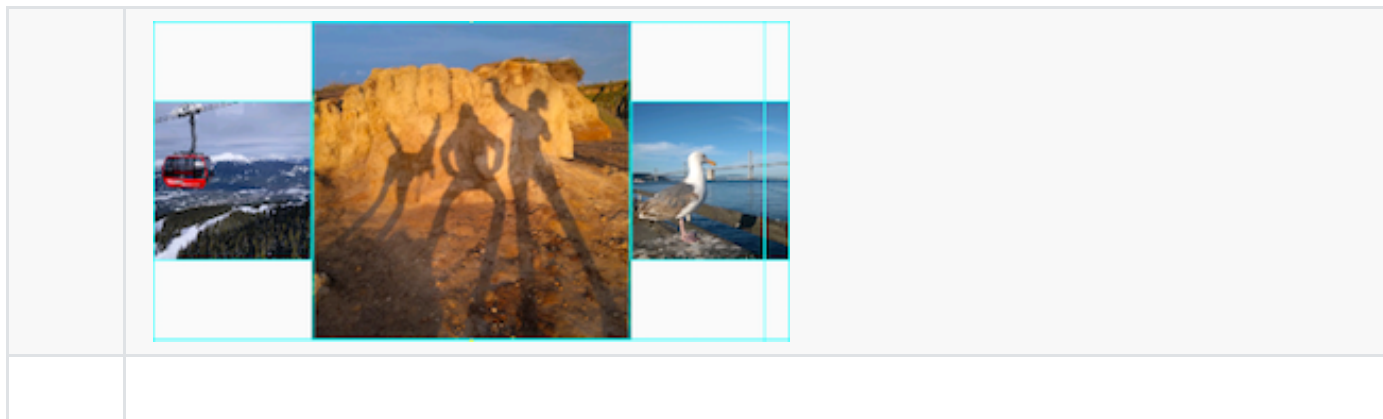
Các widget con có thể được định kích thước để vừa với một `Row` hoặc `Column` bằng cách sử dụng `Expanded` widget. Để sửa ví dụ trước trong đó hàng hình ảnh quá rộng đối với mức hiển thị của nó, hãy bọc mỗi hình ảnh bằng một `Expanded` widget.

```
1 Row(  
2   crossAxisAlignment: CrossAxisAlignment.center,  
3   children: [  
4     Expanded(  
5       child: Image.asset('images/pic1.jpg'),  
6     ),  
7     Expanded(  
8       child: Image.asset('images/pic2.jpg'),  
9     ),  
10    Expanded(  
11      child: Image.asset('images/pic3.jpg'),  
12    ),  
13  ],  
14 );
```



Nếu bạn muốn một widget chiếm không gian gấp đôi so với các widget anh chị em của nó. Để làm điều này, hãy sử dụng thuộc tính `flex` trong `Expanded` widget, một số nguyên xác định hệ số flex cho widget con. Hệ số flex mặc định là 1. Đoạn mã sau đặt hệ số flex của hình ảnh giữa thành 2:

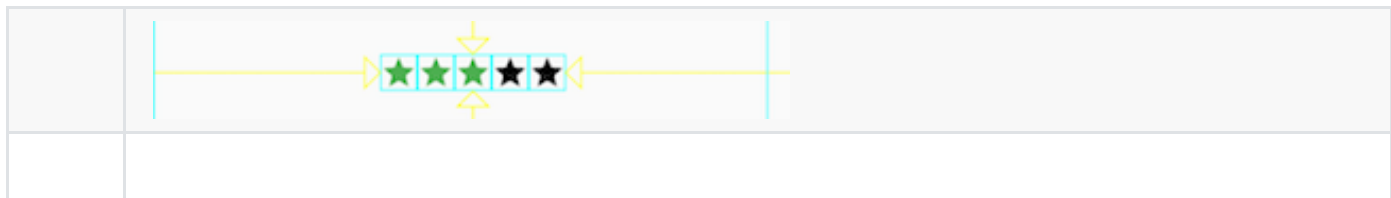
```
1 Row(  
2   crossAxisAlignment: CrossAxisAlignment.center,  
3   children: [  
4     Expanded(  
5       child: Image.asset('images/pic1.jpg'),  
6     ),  
7     Expanded(  
8       flex: 2,  
9       child: Image.asset('images/pic2.jpg'),  
10    ),  
11    Expanded(  
12      child: Image.asset('images/pic3.jpg'),  
13    ),  
14  ],  
15 );
```

Packing widgets

Theo mặc định, một `Row` hoặc `Column` chiếm càng nhiều không gian dọc theo trục chính của nó càng tốt, nhưng nếu bạn muốn kéo các widget con lại gần nhau, hãy đặt `mainAxisSize` của nó thành `MainAxisSize.min`. Ví dụ sau sử dụng thuộc tính này để đóng gói các hình ngôi sao lại với nhau.

```
1 Row(  
2   mainAxisSize: MainAxisSize.min,  
3   children: [  
4     Icon(Icons.star, color: Colors.green[500]),  
5     Icon(Icons.star, color: Colors.green[500]),  
6     Icon(Icons.star, color: Colors.green[500]),  
7     const Icon(Icons.star, color: Colors.black),  
8     const Icon(Icons.star, color: Colors.black),  
9   ],  
10  )
```



Nesting rows and columns

Layout framework cho phép bạn lồng các hàng và cột vào bên trong các hàng và cột tùy thích. Hãy xem đoạn code cho phần được phác thảo của bố cục sau:

Strawberry Pavlova

Pavlova is a meringue-based dessert named after the Russian ballerina Anna Pavlova. Pavlova features a crisp crust and soft, light inside, topped with fruit and whipped cream.

★★★★★

170 Reviews

🕒

PREP:

25 min

🕒

COOK:

1 hr

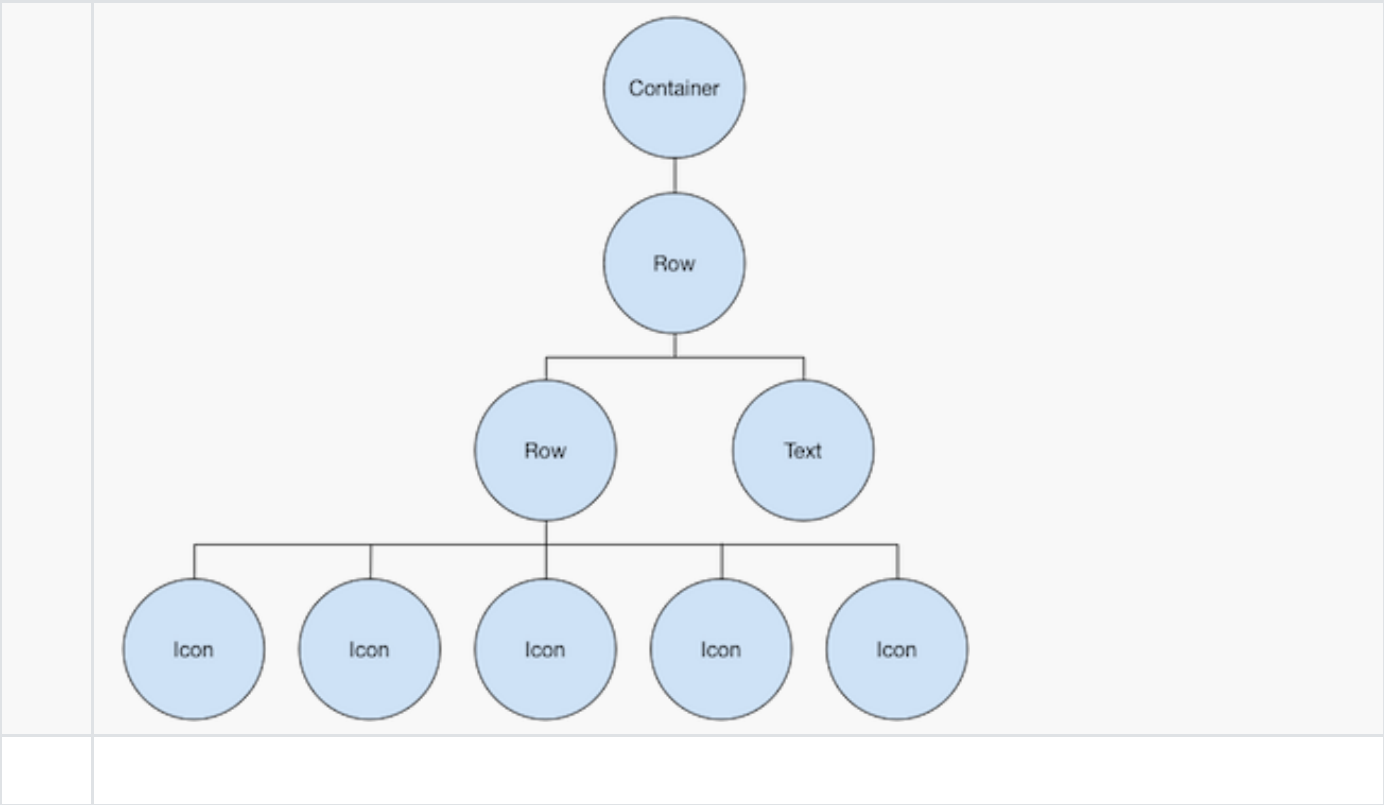
🍴

FEEDS:

4-6

Phần được phác thảo được thực hiện thành hai hàng. Hàng xếp hạng chứa năm sao và số lượng đánh giá. Hàng biểu tượng chứa ba cột biểu tượng và văn bản.

Widget tree cho Row chứa năm sao như sau:



Và mã code cho Row chứa năm sao và số lượng đánh giá như sau:

```
1 var stars = Row(
```

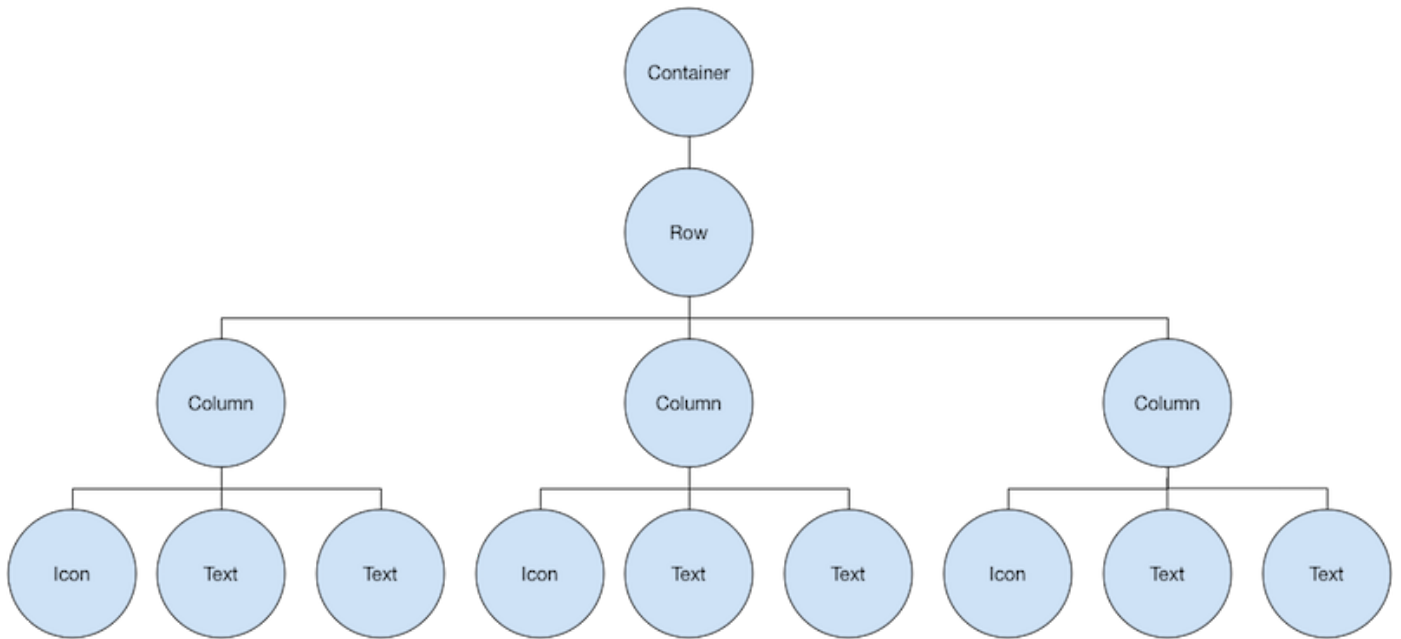
```

2  mainAxisSize: MainAxisSize.min,
3  children: [
4      Icon(Icons.star, color: Colors.green[500]),
5      Icon(Icons.star, color: Colors.green[500]),
6      Icon(Icons.star, color: Colors.green[500]),
7      const Icon(Icons.star, color: Colors.black),
8      const Icon(Icons.star, color: Colors.black),
9  ],
10 );
11
12 final ratings = Container(
13   padding: const EdgeInsets.all(20),
14   child: Row(
15     mainAxisAlignment: MainAxisAlignment.spaceEvenly,
16     children: [
17       stars,
18       const Text(
19         '170 Reviews',
20         style: TextStyle(
21           color: Colors.black,
22           fontWeight: FontWeight.w800,
23           fontFamily: 'Roboto',
24           letterSpacing: 0.5,
25           fontSize: 20,
26         ),
27       ),
28     ],
29   ),
30 );

```

Để giảm thiểu sự nhầm lẫn trực quan có thể do code các layout lồng nhau nhiều, hãy viết các phần của giao diện người dùng trong các biến và hàm.

Hàng biểu tượng, bên dưới hàng xếp hạng, có 3 cột, mỗi cột chứa một biểu tượng và hai dòng văn bản, như bạn có thể thấy trong widget tree con của nó:



Mã code như sau:

```
1  const descTextStyle = TextStyle(  
2  color: Colors.black,  
3  fontWeight: FontWeight.w800,  
4  fontFamily: 'Roboto',  
5  letterSpacing: 0.5,  
6  fontSize: 18,  
7  height: 2,  
8  );  
9  
10 // DefaultTextStyle.merge() allows you to create a default text  
11 // style that is inherited by its child and all subsequent children.  
12 final iconList = DefaultTextStyle.merge(  
13 style: descTextStyle,  
14 child: Container(  
15   padding: const EdgeInsets.all(20),  
16   child: Row(  
17     mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
18     children: [  
19       Column(  
20         children: [  
21           Icon(Icons.kitchen, color: Colors.green[500]),  
22           const Text('PREP:'),  
23           const Text('25 min'),  
24         ],  
25       ),  
26       Column(  
27         children: [  
28           Icon(Icons.timer, color: Colors.green[500]),  
29           const Text('COOK:'),  
30           const Text('1 hr'),  
31         ],
```

```

32     ),
33     Column(
34         children: [
35             Icon(Icons.restaurant, color: Colors.green[500]),
36             const Text('FEEDS:'),
37             const Text('4-6'),
38         ],
39     ),
40 ],
41 ),
42 ),
43 );

```

Ta có mã code cho phần hình bên trái như sau:

```

1  final leftColumn = Container(
2  padding: const EdgeInsets.fromLTRB(20, 30, 20, 20),
3  child: Column(
4      children: [
5          titleText,
6          subTitle,
7          ratings,
8          iconList,
9      ],
10 ),
11 );

```

Mã code cho toàn bộ UI của chúng ta như sau:

```

1  body: Center(
2  child: Container(
3      margin: const EdgeInsets.fromLTRB(0, 40, 0, 30),
4      height: 600,
5      child: Card(
6          child: Row(
7              crossAxisAlignment: CrossAxisAlignment.start,
8              children: [
9                  SizedBox(
10                     width: 440,
11                     child: leftColumn,
12                 ),
13                 mainImage,
14             ],
15         ),
16     ),

```

```
17 | ),
18 | ),
```

Common layout widgets - Một số widget phổ biến thường gặp

Standard widgets

- Container: Có thể thêm padding, margins, borders, background color, hoặc decorations vào a widget.
- GridView: Tạo widgets dạng lưới có khả năng scroll.
- ListView: Tạo widgets dạng danh sách có khả năng scroll.
- Stack: Tạo widget có khả năng đè lên trên các widget khác.

Material widgets

- Card: Tạo 1 widget hình hộp được bo góc và có đổ bóng.
- ListTile: Tạo widget với 3 dạng text, có leading và trailing icon trong 1 dòng.

Container

Rất nhiều layout sử dụng `Container`, `Container` dùng để padding, thêm viền, margin... Bạn có thể thay màu nền của thiết bị bằng cách đổi toàn bộ layout thành `Container` và đổi màu nền hoặc ảnh.

Tóm tắt về `Container`

- Thêm padding, margins, borders
- Đổi màu background color hoặc image
- Bao gồm một child widget, nhưng child có thể là Row, Column, hoặc thậm chí là root của một widget tree



Examples (Container)

Ví dụ dưới tạo 1 `Column` với 2 `Row`, mỗi `Row` lại gồm 2 hình ảnh. `Container` được sử dụng để đổi màu của `Column` thành màu xám.

```
1  Widget _buildImageColumn() {
2    return Container(
3      decoration: const BoxDecoration(
4        color: Colors.black26,
5      ),
6      child: Column(
7        children: [
8          _buildImageRow(1),
9          _buildImageRow(3),
10       ],
11     ),
12   );
13 }
```




GridView

Sử dụng GridView để sắp xếp các widget dưới dạng danh sách hai chiều. GridView cung cấp hai danh sách tạo sẵn hoặc bạn có thể tạo lưới tùy chỉnh của riêng mình. Khi GridView phát hiện thấy nội dung của nó quá dài để vừa với render box, nó sẽ tự động cuộn.

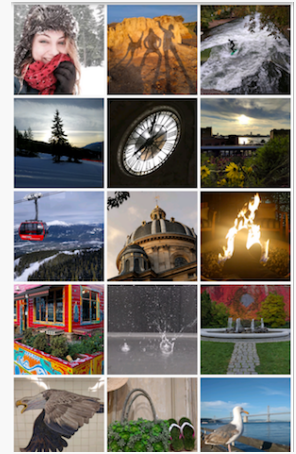
Tóm tắt (GridView)

- Đặt các widget trong một danh sách dạng lưới
- Phát hiện khi nội dung vượt quá render box và tự động cung cấp chức năng cuộn
- Tạo lưới tùy chỉnh của riêng bạn hoặc sử dụng một trong các lưới được cung cấp:
 - `GridView.count` cho phép bạn chỉ định số lượng cột
 - `GridView.extent` cho phép bạn chỉ định chiều rộng pixel tối đa của một ô

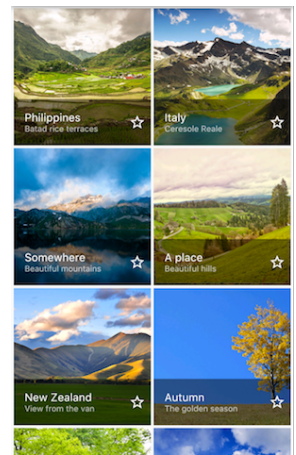
Lưu ý: Khi hiển thị danh sách hai chiều, trong đó điều quan trọng là hàng và cột mà ô chiếm dùng để chứa dữ liệu, hãy sử dụng Bảng hoặc Bảng dữ liệu.

Examples (GridView)

Sử dụng `GridView.extent` để tạo danh Sách lưới với các ô có chiều rộng tối đa là 150 pixel.



Sử dụng `GridView.count` để tạo danh sách dạng lưới với 2 ô ở chế độ dọc và 3 ô ở chế độ ngang. Tiêu đề được tạo bằng cách đặt thuộc tính footer cho mỗi `GridTile`.



```
1 Widget _buildGrid() => GridView.extent(  
2   maxCrossAxisExtent: 150,  
3   padding: const EdgeInsets.all(4),  
4   mainAxisSpacing: 4,  
5   crossAxisSpacing: 4,  
6   children: _buildGridTileList(30));  
7  
8 // The images are saved with names pic0.jpg, pic1.jpg...pic29.jpg.  
9 // The List.generate() constructor allows an easy way to create  
10 // a list when objects have a predictable naming pattern.  
11 List<Container> _buildGridTileList(int count) => List.generate(  
12   count, (i) => Container(child: Image.asset('images/pic$i.jpg')));
```

ListView










ListView, 1 widget giống như Column, nhưng tự có khả năng cuộn khi nội dung quá dài so với render box của nó.

Tóm tắt (ListView)

- Một Column được thiết kế cho việc tổ chức 1 danh sách các hộp, item.
- Có thể đặt theo chiều dọc hoặc ngang.
- Phát hiện khi nội dung không vừa và tự động cuộn.
- Ít phải cấu hình hơn Column, nhưng dễ sử dụng và trợ giúp việc scroll.

Examples (ListView)

Sử dụng `ListView` để hiển thị danh sách công việc kinh doanh với `ListTiles`. Một thanh ngăn `Divider` giúp chia tách các rạp phim với danh sách nhà hàng

-  **CineArts at the Empire**
85 W Portal Ave
-  **The Castro Theater**
429 Castro St
-  **Alamo Drafthouse Cinema**
2550 Mission St
-  **Roxie Theater**
3117 16th St
-  **United Artists Stonestown Twin**
501 Buckingham Way
-  **AMC Metreon 16**
135 4th St #3000
-  **K's Kitchen**
757 Monteneroy Blvd
-  **Emmy's Theater**
1923 Ocean Ave
-  **Chaiya Thai Restaurant**
272 Clarendon Blvd

Sử dụng ListView để hiện danh sách Colors từ Material Design palette.

DEEP PURPLE	INDIGO	BLUE	LIGHT BLUE	CYAN
50				#FFE32FD
100				#FFBDEFB
200				#FF90CAF9
300				#FF64B5F6
400				#FF42A5F5
500				#FF2196F3
600				#FF1E88E5
700				#FF197BD2
800				#FF1565C0
900				#FF0D47A1
A100				#FFB2B1FF
A200				#FF448AFF
A400				#FF2979FF

```

1 widget_buildList() {
2   return ListView(
3     children: [
4       _tile('CineArts at the Empire', '85 W Portal Ave', Icons.theaters),
5       _tile('The Castro Theater', '429 Castro St', Icons.theaters),
6       _tile('Alamo Drafthouse Cinema', '2550 Mission St', Icons.theaters),
7       _tile('Roxie Theater', '3117 16th St', Icons.theaters),
8       _tile('United Artists Stonestown Twin', '501 Buckingham Way',
9         Icons.theaters),
10      _tile('AMC Metreon 16', '135 4th St #3000', Icons.theaters),
11      const Divider(),
12      _tile('K's Kitchen', '757 Monterey Blvd', Icons.restaurant),
13      _tile('Emmy's Restaurant', '1923 Ocean Ave', Icons.restaurant),
14      _tile(
15        'Chaiya Thai Restaurant', '272 Claremont Blvd', Icons.restaurant),
16      tile('La Ciccia', '291 30th St', Icons.restaurant),

```

```

17     ],
18   );
19 }
20
21 ListTile _tile(String title, String subtitle, IconData icon) {
22   return ListTile(
23     title: Text(title,
24       style: const TextStyle(
25         fontWeight: FontWeight.w500,
26         fontSize: 20,
27       )),
28     subtitle: Text(subtitle),
29     leading: Icon(
30       icon,
31       color: Colors.blue[500],
32     ),
33   );
34 }

```


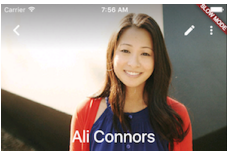
Stack

Sử dụng `stack` để sắp xếp các widget ở phía trên các widget khác - thường là 1 `Image`. `Stack` có thể chiếm toàn bộ widget phía dưới nó.

Tóm tắt (Stack)

- Sử dụng cho việc chồng các widget lên nhau.
- Widget đầu tiên trong danh sách của `stack` gọi là widget base, các widget tiếp sau trong danh sách widget sẽ được đặt đè lên trên widget Base.
- Content của `stack` không thể cuộn.
- Bạn có thể chọn cách cắt bớt các widget con để hiển thị quá render box.

Examples (Stack)

	<p>Sử dụng <code>stack</code> để đè 1 <code>Container</code> (Hiện 1 <code>Text</code> với background màu đen trong suốt) lên phía trên của một <code>CircleAvatar</code>. Vị trí của <code>Text</code> sử dụng <code>alignment</code> property hoặc <code>Alignments</code> widget.</p>
	<p>Sử dụng 'Stack' để đè 1 gradient lên phía trên hình ảnh. Gradient này đảm bảo cho việc hiển thị icon của toolbar phân tách rõ với hình ảnh bên dưới.</p>

```

1  Widget _buildStack() {
2    return Stack(
3      alignment: const Alignment(0.6, 0.6),
4      children: [
5        const CircleAvatar(
6          backgroundImage: AssetImage('images/pic.jpg'),
7          radius: 100,
8        ),
9        Container(
10       decoration: const BoxDecoration(
11         color: Colors.black45,
12       ),
13       child: const Text(
14         'Mia B',
15         style: TextStyle(
16           fontSize: 20,
17           fontWeight: FontWeight.bold,
18           color: Colors.white,
19         ),
20       ),
21     ),
22   ],
23 );
24 }

```

Card

`Card`, từ thư viện `Material`, chứa các thông tin liên quan và có thể được tạo từ hầu hết các widget, nhưng thường được sử dụng với `ListTile`. `Card` có một widget con, nhưng widget con của nó có thể là một `Colum`, `Row`, `ListView`, `GridView` hoặc các widget con khác hỗ trợ list widget. Theo mặc định, một `card` thu nhỏ kích thước của nó thành 0 x 0 pixel. Bạn có thể sử dụng `SizedBox` để giới hạn kích thước của thẻ.

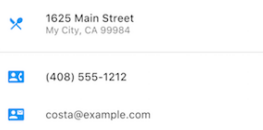
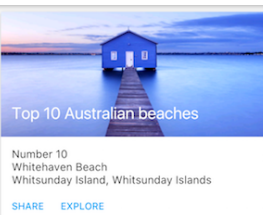
Trong Flutter, `Card` có các góc bo tròn và bóng đổ, tạo hiệu ứng 3D. Thay đổi thuộc tính `elevation` của `Card` cho phép bạn kiểm soát hiệu ứng đổ bóng. Ví dụ: đặt `elevation` lên 24, trực quan nâng `Card` lên khỏi bề mặt và làm cho bóng trở nên phân tán hơn. Để biết danh sách các giá trị độ cao được hỗ trợ, hãy xem `Elevation` trong nguyên lý `Material`. Việc chỉ định một giá trị không được hỗ trợ sẽ vô hiệu hóa hoàn toàn bóng đổ.

Tóm tắt (Card).

- Triển khai `Material card`
- Được sử dụng để trình bày các thông tin liên quan
- Chấp nhận một widget con, nhưng phần tử đó có thể là `Colum`, `Row`, `ListView`, `GridView` hoặc các widget con khác hỗ trợ list widget

- Được hiển thị với các góc bo tròn và bóng đổ
- Nội dung của `card` không thể cuộn
- Từ thư viện `Material`

Examples (Card)

	<p>Một <code>Card</code> bao gồm 3 <code>ListTiles</code> và giới hạn kích thước bằng các bọc trong một <code>SizedBox</code>. Một thanh ngăn <code>Divider</code> giúp phân tách 2 <code>ListTiles</code> với nhau.</p>
	<p>Một <code>card</code> bao gồm hình ảnh và text.</p>

```

1  Widget _buildCard() {
2    return SizedBox(
3      height: 210,
4      child: Card(
5        child: Column(
6          children: [
7            ListTile(
8              title: const Text(
9                '1625 Main Street',
10               style: TextStyle(fontWeight: FontWeight.w500),
11             ),
12             subtitle: const Text('My City, CA 99984'),
13             leading: Icon(
14               Icons.restaurant_menu,
15               color: Colors.blue[500],
16             ),
17           ],
18           const Divider(),
19           ListTile(
20             title: const Text(
21               '(408) 555-1212',
22             style: TextStyle(fontWeight: FontWeight.w500),
23           ),
24             leading: Icon(
25               Icons.contact_phone,
26               color: Colors.blue[500],
27             ),
28           ],
29           ListTile(
30             title: const Text('costa@example.com'),
31             leading: Icon(
32               Icons.contact_mail,
```

```
33         color: Colors.blue[500],
34     ),
35 ),
36 ],
37 ),
38 ),
39 );
40 }
```




ListTile

Sử dụng `ListTile`, một widget giống `Row` từ thư viện `Material`, để dễ dàng tạo một hàng chứa tối đa 3 dòng văn bản và các biểu tượng đầu và cuối tùy chọn. `ListTile` được sử dụng phổ biến nhất trong `Card` hoặc `ListView`, nhưng có thể được sử dụng ở những nơi khác.

Tóm tắt (ListTile)

- Một widget giống `Row` chứa tối đa 3 dòng văn bản và các biểu tượng tùy chọn
- Ít cấu hình hơn `Row`, nhưng dễ sử dụng hơn
- Từ thư viện `Material`

Examples (ListTile)

<div><div><div> 1625 Main Street My City, CA 99984</div><div> (408) 555-1212</div><div> costa@example.com</div></div></div>	<p>Một <code>card</code> bao gồm 3 <code>ListTiles</code>.</p>
<div><p>A dropdown button displays a menu that's used to select a value from a small set of values. The button displays the current value and a down arrow.</p><div><div>Simple dropdown:</div><div>Free ▾</div></div><div><div>Dropdown with a hint:</div><div>Choose ▾</div></div><div><div>Scrollable dropdown:</div><div>Four ▾</div></div></div>	<p>Sử dụng <code>ListTile</code> để tạo danh sách gồm 3 button dạng dropdown.</p>

** Tham khảo tài liệu tại Techmaster.vn