

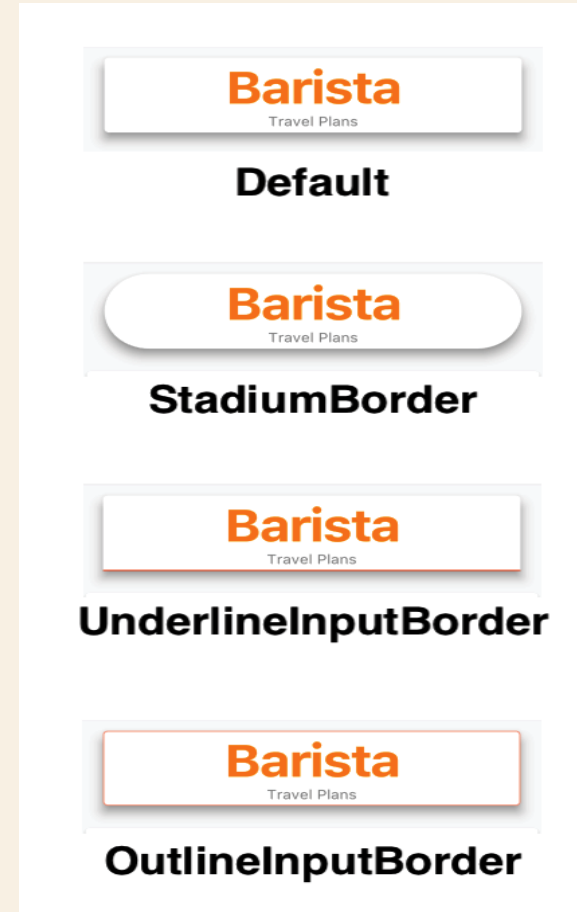
Creating Scrolling Lists and Gridview

USING THE CARD

- The **Card** widget is part of the Material Design and has minimal rounded corners and shadows. To group and lay out data,
- The Card widget is customizable with properties such as **elevation**, **shape**, **color**, **margin**, and others.
- The **elevation** property is a value of double, and the higher the number, the larger the shadow that is cast.
- To customize the shape and borders of the **Card** widget, you modify the shape property. Some of the **shape** properties are **StadiumBorder**, **UnderlineInputBorder**, **OutlineInputBorder**, and others.

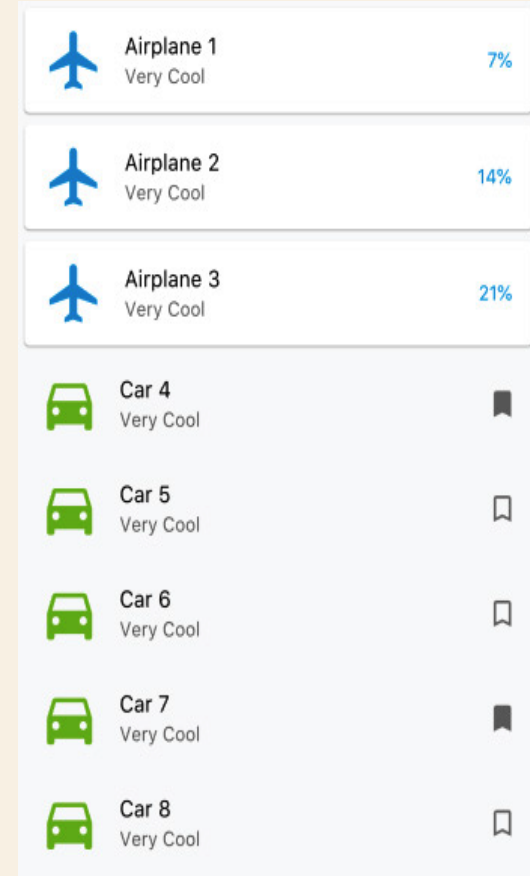
The following are a few ways to customize the Card's shape property

- **// Create a Stadium Border**
shape: StadiumBorder(),
- **// Create Square Corners Card with a Single Orange Bottom Border**
shape: UnderlineInputBorder(borderSide: BorderSide(color: Colors.deepOrange)),
- **// Create Rounded Corners Card with Orange Border**
shape: OutlineInputBorder(borderSide: BorderSide(color: Colors.deepOrange.withOpacity(0.5))),



USING THE LISTVIEW AND LISTTILE

- The constructor **ListView.builder** is used to create an on-demand **linear scrollable list** of widgets .
- Within the builder, you use the **itemBuilder** callback to **create the list of children** widgets.
- The **scrollDirection** argument defaults to **Axis.vertical** but can be **changed** to **Axis.horizontal**.
.....
- The **ListTile** widget is commonly used with the **ListView** widget to easily format and organize icons, **titles**, and **descriptions** in a **linear** layout.
- You can also use the **onTap** and **onLongPress** **callbacks** to execute an action when the user taps the **ListTile**.



Flutter GridView

GridView is a widget in Flutter that arranges the list of its children in a two-dimensional grid pattern. It allows us to store and display our items in a matrix form.

This widget is highly useful when you want to display a set of widgets in a grid format. These widgets can be images, cards, icons, or any others.

Moreover, it automatically handles scrolling when there are more items that can practically fit on the device's screen. This saves the developer from the hassle of taking care of the scrolling methods.

Types of GridView

There are two main types of **GridView** that are used often.

1. **GridView.count**

2. **GridView.builder**

GridView.count()

It is the most frequently used grid layout in [Flutter](#) because here, we already know the grid's size. It allows developers to **specify the fixed number of rows and columns**. The GridView.count() contains the following properties:

crossAxisCount: It is used to specify the number of columns in a grid view.

crossAxisSpacing: It is used to specify the number of pixels between each child widget listed in the cross axis.

mainAxisSpacing: It is used to specify the number of pixels between each child widget listed in the main axis.

padding(EdgeInsetsGeometry): It is used to specify the space around the whole list of widgets.

scrollDirection: It is used to specify the direction in which the items on GridView scrolls. By default, it scrolls in a vertical direction.

reverse: If it is true, it will reverse the list in the opposite direction along the main axis.

physics: It is used to determine how the list behaves when the user reaches the end or the start of the widget while scrolling.

```
GridView.count( crossAxisCount: 3, // Adjust the number of columns here
children: _generateGridItems(), // A function that returns a list of widgets ),
```

This constructor is useful when you have a small number of items that are not dynamic and are not planned to change.

Flutter – Dismissible Widget

The **Dismissible** widget in Flutter is used to create items that can be dismissed by swiping them off the screen. It's commonly used in lists or grids where you want to provide a way for users to remove items with a swipe gesture.

Basic Syntax of Dismissible Widget

```
Dismissible(  
  key: UniqueKey(), // or any unique key for tracking items  
  child: YourContentWidget(),  
  background: YourBackgroundWidget(),  
  secondaryBackground: YourSecondaryBackgroundWidget(),  
  confirmDismiss: (DismissDirection direction) async {  
    // Your confirmation logic goes here  
    // Return true to allow dismissal, false to prevent it  
    return true;  
  },  
  onDismissed: (DismissDirection direction) {  
    // Your action when item is dismissed goes here  
  },  
  onResize: () {  
    // Your resize animation logic goes here (optional)  
  },  
  direction: DismissDirection.endToStart, // or other DismissDirection values  
  dragStartBehavior: DragStartBehavior.start, // or DragStartBehavior.down  
)
```

1: Create MyApp Class

In this class we are going to implement the `MaterialApp`, here we are also set the Theme of our App.

```
class MyApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      debugShowCheckedModeBanner: false,  
      theme: ThemeData(  
        primarySwatch: Colors.green, // Set the app's primary theme color  
      ),  
      title: 'Dismissible Example',  
      home: DismissibleExample(),  
    );  
  }  
}
```

2: Create DismissibleExample Class

In this class we are going to Implement the Dismissible widget whenever the user Swipe the List items then the swiped items are deleted. Comments are added for better understanding.

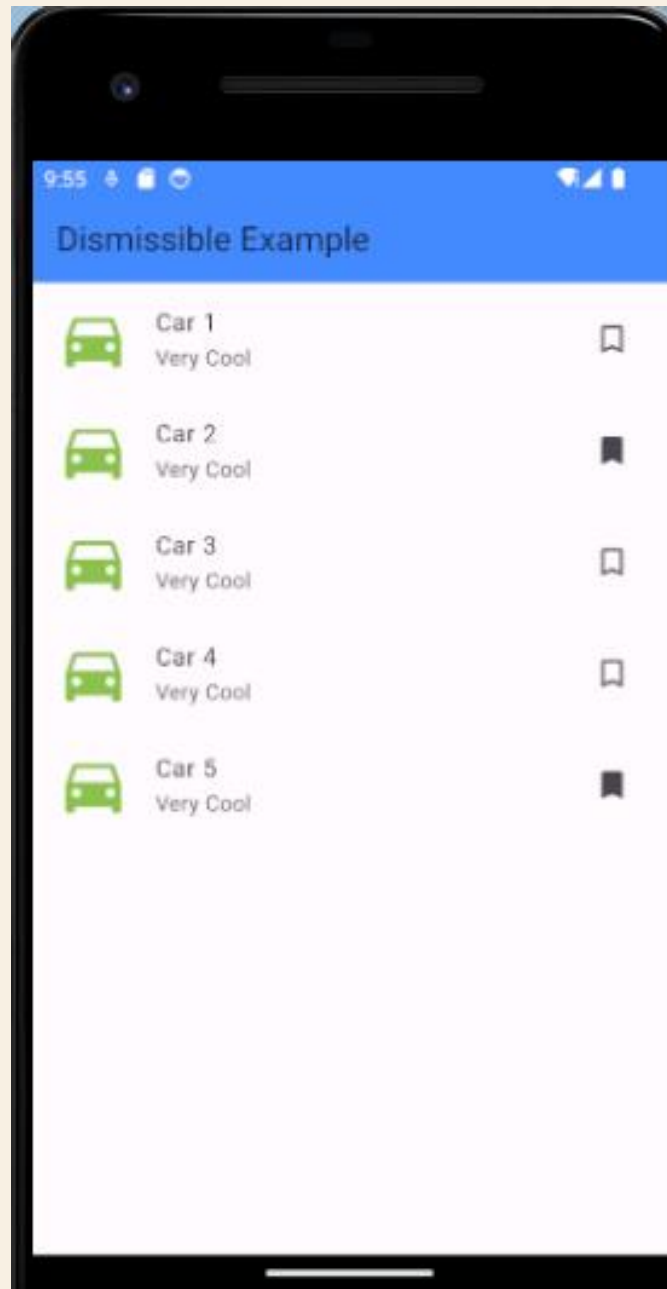
```
class DismissibleExample extends StatefulWidget {
  @override
  _DismissibleExampleState createState() => _DismissibleExampleState();
}

class _DismissibleExampleState extends State<DismissibleExample> {
  // Sample list of items
  List<String> items = List.generate(5, (index) => 'Car ${index + 1}');

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.blueAccent,
        title: Text('Dismissible Example'),
      ),
    ),
  )
}
```



```
body: ListView.builder(
  itemCount: items.length,
  itemBuilder: (context, index) {
    final item = items[index];
    return Dismissible(
      key: Key(item), // Unique key for each item
      onDismissed: (direction) {
        // Remove the item from the list when dismissed
        setState(() {
          items.removeAt(index);
        });
        // Show a snackbar to indicate item removal
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text('$item dismissed'),
          ),
        );
      },
      background: Container(
        color: Colors.red, // Background color when swiping
        child: Icon(Icons.delete, color: Colors.white, size: 36, ),
        alignment: Alignment.centerRight,
        padding: EdgeInsets.only(right: 20),
      ),
      child: ListTile(
        leading: Icon(Icons.directions_car, size: 48.0, color: Colors.lightGreen, ),
        title: Text(item),
        subtitle: Text('Very Cool'),
      ),
      trailing: (index % 3).isEven ? Icon(Icons.bookmark_border) : Icon(Icons.bookmark),
    );
  },
);
```



GridView.builder()

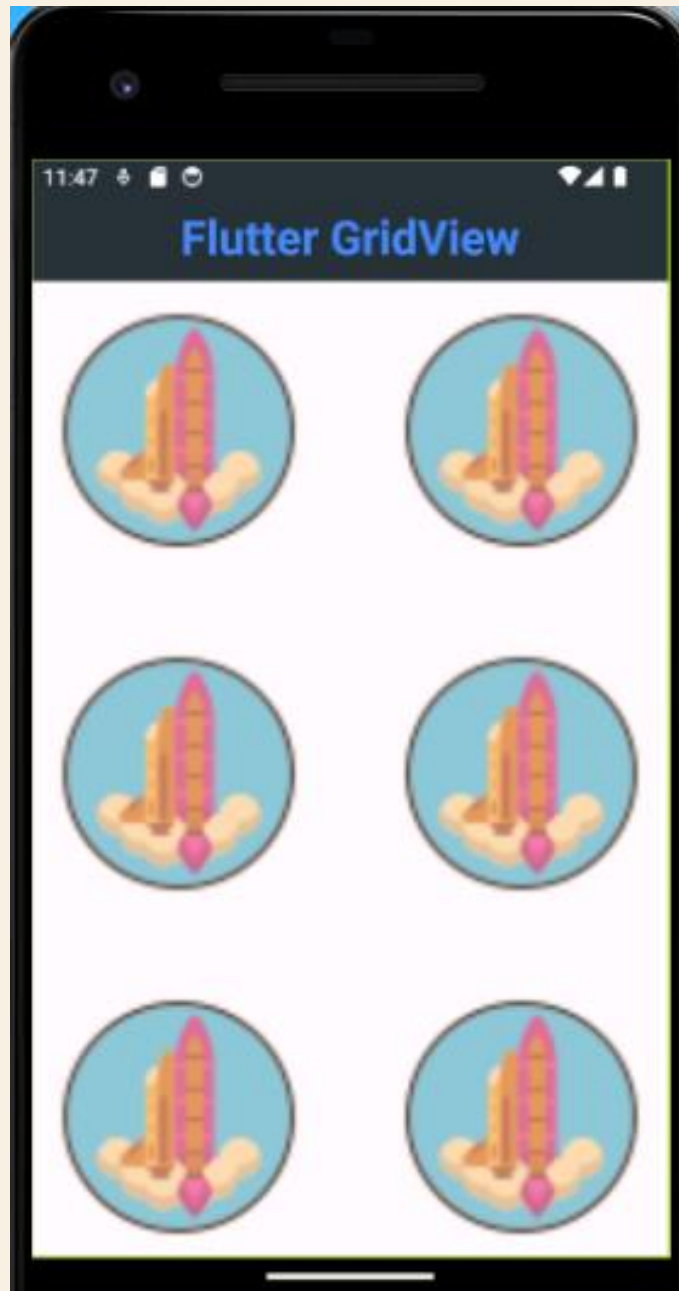
This constructor is more dynamic and provides more suitability when you have a large number of items in the grid or when the children widgets are built dynamically during runtime.

```
GridView.builder(  
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(  
    crossAxisCount: 3, // Number of columns in the grid  
    crossAxisSpacing: 8.0, // Spacing between columns  
    mainAxisSpacing: 8.0, // Spacing between rows  
  ),  
  itemCount: itemCount,  
  itemBuilder: (BuildContext context, int index) {  
    return Container(  
      color: Colors.white,  
      child: Center(  
        child: Text(  
          'Item $index',  
          style: TextStyle(color: Colors.white),  
        ),  
      ),  
    );  
  },  
);
```

```

home: Scaffold(
  appBar: AppBar(
    backgroundColor: Colors.blueGrey[900],
    title: Center(
      child: Text(
        'Flutter GridView',
        style: TextStyle(
          color: Colors.blueAccent,
          fontWeight: FontWeight.bold,
          fontSize: 30.0, ), ), ), ),
  body: GridView.count(
    crossAxisCount: 2,
    crossAxisSpacing: 30.0,
    mainAxisSpacing: 30.0,
    children: List.generate(6, (index) {
      return Padding(
        padding: const EdgeInsets.all(10.0),
        child: Container(
          decoration: BoxDecoration(
            image: DecorationImage(
              image: AssetImage('images/rockets.png'),
              fit: BoxFit.cover, ),
            borderRadius: BorderRadius.all(
              Radius.circular(20.0), ), ), ), ), );}, ), ), ), );} }

```



```

return MaterialApp(
  debugShowCheckedModeBanner: false,
  title: 'Grid build',
  theme: ThemeData(
    colorScheme: ColorScheme.fromSeed(seedColor: Colors.blue),
    useMaterial3: true, ),
  home: HomeScreen(), );}}
class HomeScreen extends StatelessWidget {
  HomeScreen({Key? key}) : super(key: key);
  final List<Map> myProducts =
    List.generate(8, (index) => {"id": index, "name": "Product $index"}).toList();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('GridView Builder'), ),
      body: Padding(padding: const EdgeInsets.all(8.0),
        child: GridView.builder(
          gridDelegate: const SliverGridDelegateWithMaxCrossAxisExtent(
            maxCrossAxisExtent: 200,
            childAspectRatio: 3 / 2,
            crossAxisSpacing: 20,
            mainAxisSpacing: 20),
          itemCount: myProducts.length,
          itemBuilder: (BuildContext ctx, index) {
            return Container( alignment: Alignment.center,
              decoration: BoxDecoration(color: Colors.blueAccent,
                borderRadius: BorderRadius.circular(15)),
              child: Text(myProducts[index]["name"]),); })),), );}}

```

