IS487
Emerging Topics in IS

Dr. Abbas Malik
amaalik@psu.edu.sa

Dart

- Object-Oriented
- Strongly typed
- Single-threaded event loop
- Developer Experience

Dart

Main Method

```dart
void main() {
  print('Hello, World!');
}
```

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

Dart

Data Types and Variables

- int, double, num, String, bool, List<type>, Set<type>, Map<key_type, dynamic>, … more on types
- Variables: var keyword to declare a variable and tyle determined based on the value assigned to it.

Dart

Flow Control

- IF - ELSE
- IF - ELSE - IF
- For
- While
- Do-While

```dart
if (year >= 2001) {
  print('21st century');
} else if (year >= 1901) {
  print('20th century');
}

for (final object in flybyObjects) {
  print(object);
}

for (int month = 1; month <= 12; month++) {
  print(month);
}

while (year < 2016) {
  year += 1;
}
```

Dart

Function

```dart
int fibonacci(int n) {
  if (n == 0 || n == 1) return n;
  return fibonacci(n - 1) + fibonacci(n - 2);
}

var result = fibonacci(20);
```

- Shorthand => arrow function

```dart
(name) => print(name);
```

Dart

Import

```dart
// Importing core libraries
import 'dart:math';

// Importing libraries from external packages
import 'package:test/test.dart';

// Importing files
import 'path/to/my_other_file.dart';
```

Dart

Classes

Enum

Inheritance

Interfaces and Abstract Classes

Async

Exceptions

...

```
enum PlanetType { terrestrial, gas, ice }
```

Learn More on Dart

Dart Virtual Machine (VM)

- Run code quickly and efficiently
- **Phases**: Compile time and Runtime
- Two Types of Compilation
  - Just in Time (JIT)
  - Ahead of Time (AOT)
- Garbage Collection
- Concurrency – isolate & async

Dart

Core Libraries

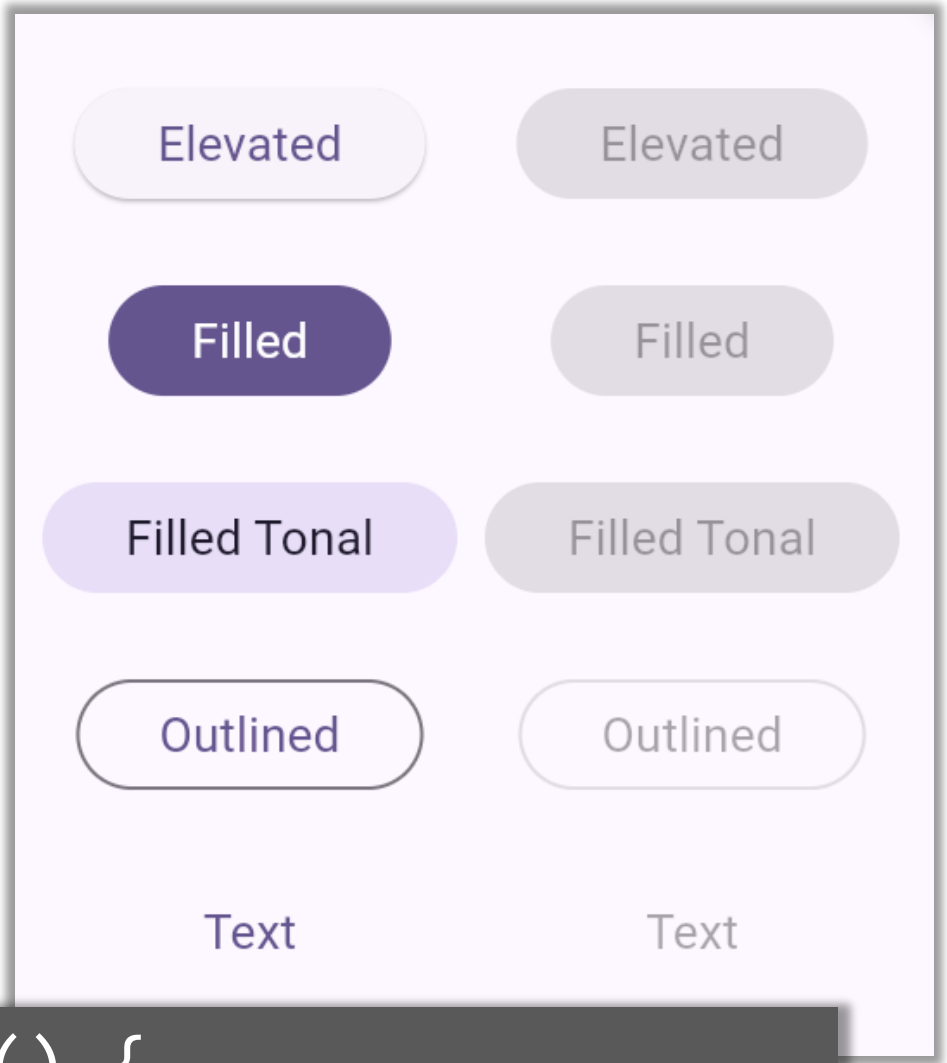
- Dart:core
- Dart:async
- Dart:io
- Dart:convert
- Dart:math

Flutter Widgets

Action

Buttons

- ElevatedButton
- FilledButton
- FilledButton.tonal
- OutlinedButton
- TextButton




```
onPressed: () {
          // onPressed code here!
        },
```

Flutter Widgets

ElevatedButtons

Inheritance
Object › DiagnosticableTree › Widget › StatefulWidget › ButtonStyleButton › ElevatedButton

Constructors

ElevatedButton ({Key? key, required VoidCallback? onPressed, VoidCallback? onLongPress, ValueChanged<bool>? onHover, ValueChanged<bool>? onFocusChange, ButtonStyle? style, FocusNode? focusNode, bool autofocus = false, Clip? clipBehavior, ~~MaterialStatesController~~? statesController, required Widget? child})

Create an ElevatedButton.

const

```
ElevatedButton(
  onPressed: () {
    print('Button is pressed');
  },
  child: Text('Elevated Button'),
),
```

Flutter Widgets

FilledButtons

Inheritance
Object › DiagnosticableTree › Widget › StatefulWidget › ButtonStyleButton › FilledButton

Constructors

FilledButton ({Key? key, required VoidCallback? onPressed, VoidCallback? onLongPress, ValueChanged<bool>? onHover, ValueChanged<bool>? onFocusChange, ButtonStyle? style, FocusNode? focusNode, bool autofocus = false, Clip? clipBehavior = Clip.none, ~~MaterialStatesController~~? statesController, required Widget? child})

Create a FilledButton.

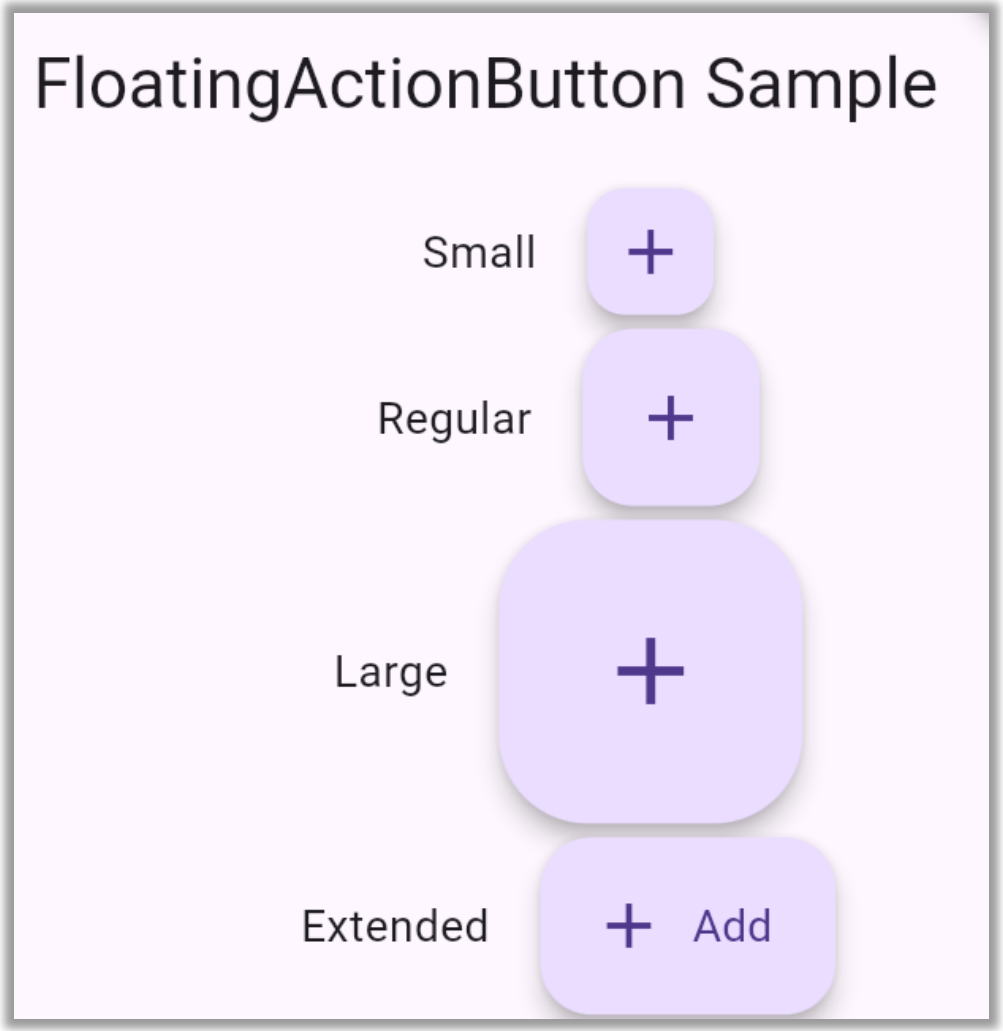
const

```
FilledButton(
  onPressed: () {
    print('Button is pressed');
  },
  child: Text('Elevated Button'),
),
```

Flutter Widgets

FloatingActionButton

- FAB
- FloatingActionButton.small
- FloatingActionButton
- FloatingActionButton.large
- FloatingActionButton.extended




```
onPressed: () {
          // code
},
```

Flutter Widgets

FloatingActionButton

```
FloatingActionButton(
  onPressed: () {
    print('Button is pressed');
  },
),
```

Constructors

FloatingActionButton ({Key? key, Widget? child, String? tooltip, Color? foregroundColor, Color? backgroundColor, Color? focusColor, Color? hoverColor, Color? splashColor, Object? heroTag = const _DefaultHeroTag(), double? elevation, double? focusElevation, double? hoverElevation, double? highlightElevation, double? disabledElevation, required VoidCallback? onPressed, MouseCursor? mouseCursor, bool mini = false, ShapeBorder? shape, Clip clipBehavior = Clip.none, FocusNode? focusNode, bool autofocus = false, MaterialTapTargetSize? materialTapTargetSize, bool isExtended = false, bool? enableFeedback})

Creates a circular floating action button.

const

Inheritance

Object › DiagnosticableTree › Widget › StatelessWidget › FloatingActionButton

Constructors

FloatingActionButton ({Key? key, Widget? child, String? tooltip, Color? foregroundColor, Color? backgroundColor, Color? focusColor, Color? hoverColor, Color? splashColor, Object? heroTag = const _DefaultHeroTag(), double? elevation, double? focusElevation, double? hoverElevation, double? highlightElevation, double? disabledElevation, required VoidCallback? onPressed, MouseCursor? mouseCursor, bool mini = false, ShapeBorder? shape, Clip clipBehavior = Clip.none, FocusNode? focusNode, bool autofocus = false, MaterialTapTargetSize? materialTapTargetSize, bool isExtended = false, bool? enableFeedback})

Creates a circular floating action button.

const

FloatingActionButton.extended ({Key? key, String? tooltip, Color? foregroundColor, Color? backgroundColor, Color? focusColor, Color? hoverColor, Object? heroTag = const _DefaultHeroTag(), double? elevation, double? focusElevation, double? hoverElevation, Color? splashColor, double? highlightElevation, double? disabledElevation, required VoidCallback? onPressed, MouseCursor? mouseCursor = SystemMouseCursors.click, ShapeBorder? shape, bool isExtended = true, MaterialTapTargetSize? materialTapTargetSize, Clip clipBehavior = Clip.none, FocusNode? focusNode, bool autofocus = false, double? extendedIconLabelSpacing, EdgeInsetsGeometry? extendedPadding, TextStyle? extendedTextStyle, Widget? icon, required Widget label, bool? enableFeedback})

Creates a wider StadiumBorder-shaped floating action button with an optional `icon` and a `label`.

const

FloatingActionButton.large ({Key? key, Widget? child, String? tooltip, Color? foregroundColor, Color? backgroundColor, Color? focusColor, Color? hoverColor, Color? splashColor, Object? heroTag = const _DefaultHeroTag(), double? elevation, double? focusElevation, double? hoverElevation, double? highlightElevation, double? disabledElevation, required VoidCallback? onPressed, MouseCursor? mouseCursor, ShapeBorder? shape, Clip clipBehavior = Clip.none, FocusNode? focusNode, bool autofocus = false, MaterialTapTargetSize? materialTapTargetSize, bool? enableFeedback})

Creates a large circular floating action button.

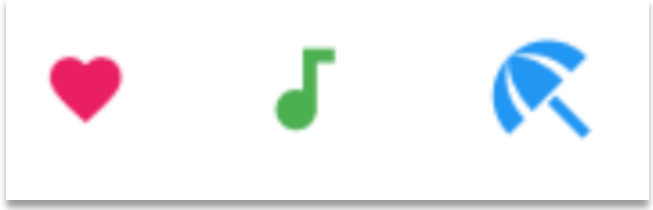
const

FloatingActionButton.small ({Key? key, Widget? child, String? tooltip, Color? foregroundColor, Color? backgroundColor, Color? focusColor, Color? hoverColor, Color? splashColor, Object? heroTag = const _DefaultHeroTag(), double? elevation, double? focusElevation, double? hoverElevation, double? highlightElevation, double? disabledElevation, required VoidCallback? onPressed, MouseCursor? mouseCursor, ShapeBorder? shape, Clip clipBehavior = Clip.none, FocusNode? focusNode, bool autofocus = false, MaterialTapTargetSize? materialTapTargetSize, bool? enableFeedback})

Creates a small circular floating action button.

const

Flutter Widgets



Material Design Icons Collection

IconButton

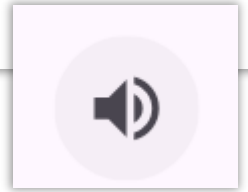
Constructors

IconButton ({Key? key, double? iconSize, VisualDensity? visualDensity, EdgeInsetsGeometry? padding, AlignmentGeometry? alignment, double? splashRadius, Color? color, Color? focusColor, Color? hoverColor, Color? highlightColor, Color? splashColor, Color? disabledColor, required VoidCallback? onPressed, ValueChanged<bool>? onHover, VoidCallback? onLongPress, MouseCursor? mouseCursor, FocusNode? focusNode, bool autofocus = false, String? tooltip, bool? enableFeedback, BoxConstraints? constraints, ButtonStyle? style, bool? isSelected, Widget? selectedIcon, ~~MaterialStatesController~~? statesController, required Widget icon})

Creates an icon button.

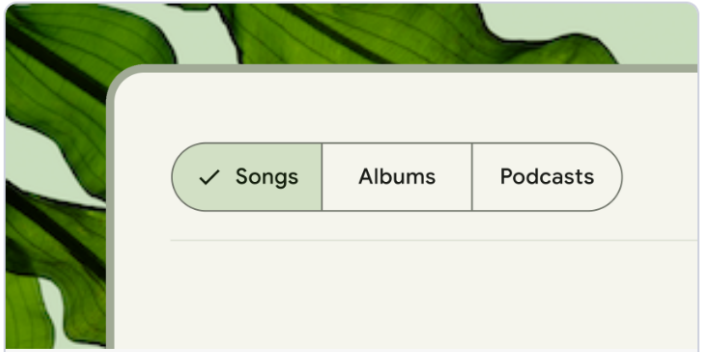
const

```
IconButton(
  onPressed: () {
    print('Button is pressed');
  },
  icon: const Icon(Icons.volume_up),
),
```



Flutter Widgets

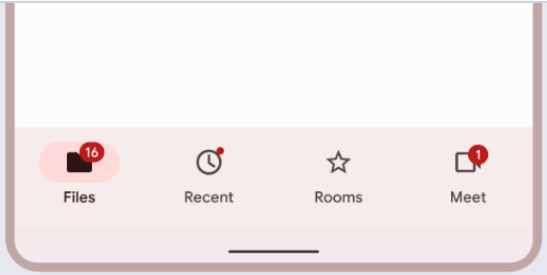
SegmentedButton



SegmentedButton

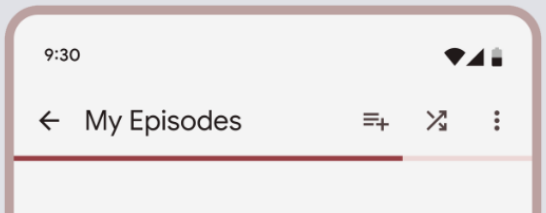
Single or multiple selected clickable blocks to help people select options, switch views, or sort elements.

Flutter Widgets
Communication



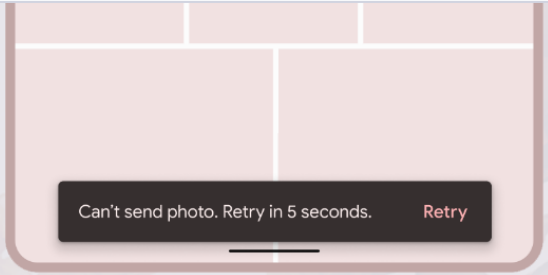
Badge

Icon-like block that conveys dynamic content such as counts or status. It can include labels or numbers.



LinearProgressIndicator

Vertical line that changes color as an ongoing process, such as loading an app or submitting a form, completes.



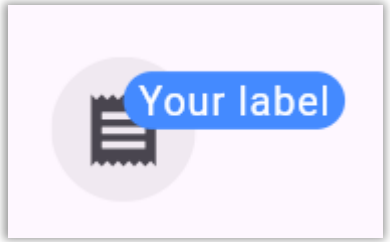
SnackBar

Brief messages about app processes that display at the bottom of the screen.

Flutter Widgets

Badge

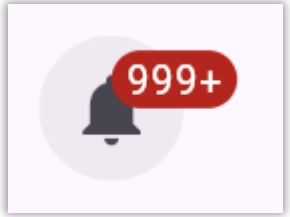
```
IconButton(
  icon: const Badge(
    label: Text('Your label'),
    backgroundColor: Colors.blueAccent,
    child: Icon(Icons.receipt),
  ),
  onPressed: () {},
),
```



Flutter Widgets

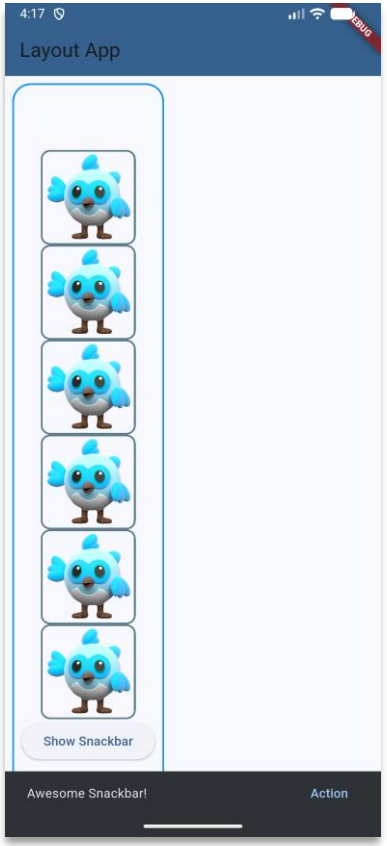
Badge

```
IconButton(
  icon: Badge.count(
    count: 9999,
    child: const Icon(Icons.notifications),
  ),
  onPressed: () {},
),
```



Flutter Widgets

SnackBar



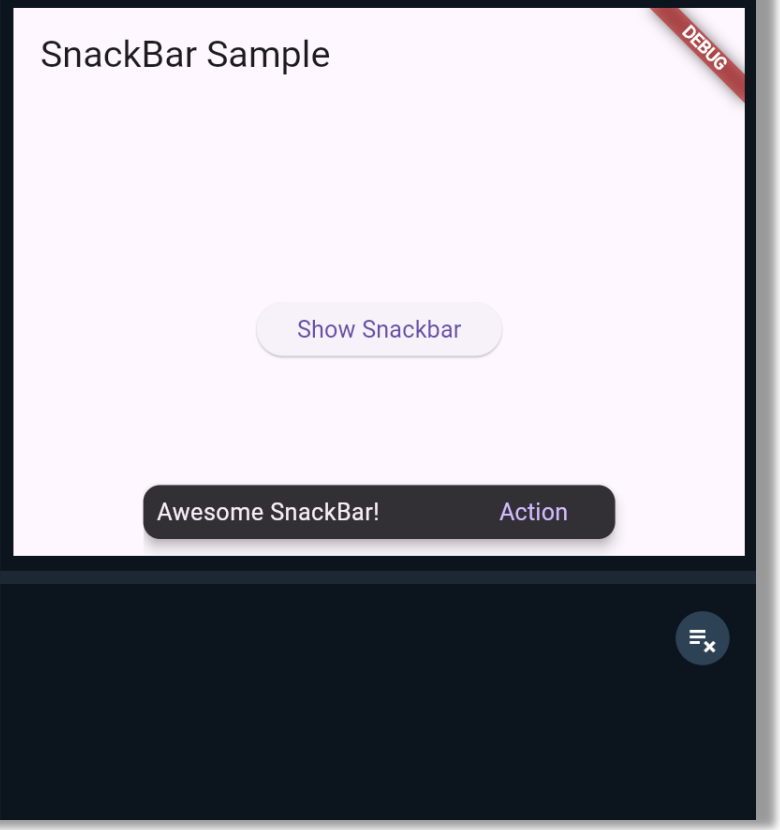

```
ElevatedButton(
  child: const Text('Show Snackbar'),
  onPressed: () {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: const Text('Awesome Snackbar!'),
        action: SnackBarAction(
        label: 'Action',
        onPressed: () {
          // Code to execute.
        },
      ),
    ),
  );
  },
),
```

Flutter Widgets

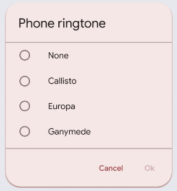
SnackBar

```dart
    return 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),
            width: 280.0, // Width of the SnackBar.
            padding: const EdgeInsets.symmetric(
              horizontal: 8.0, // Inner padding for SnackBar content.
            ),
            behavior: SnackBarBehavior.floating,
            shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10.0)),
          ),
        );
      },
    );
  }
}
```

Reload Run

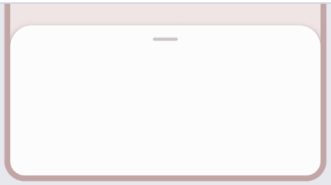


Flutter Widgets
Containment



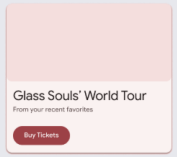
AlertDialog

Hovering containers that prompt app users to provide more data or make a decision.



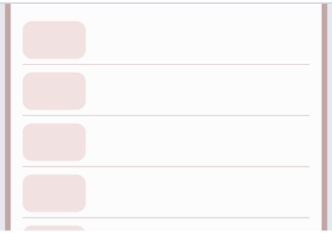
Bottom sheet

Containers that anchor supplementary content to the bottom of the screen.



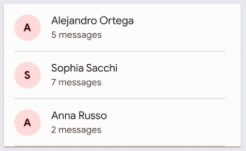
Card

Container for short, related pieces of content displayed in a box with rounded corners and a drop shadow.



Divider

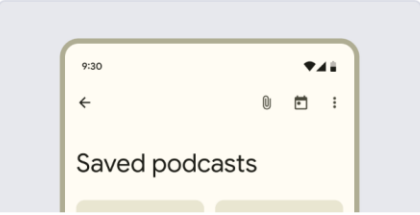
Thin line that groups content in lists and containers.



ListTile

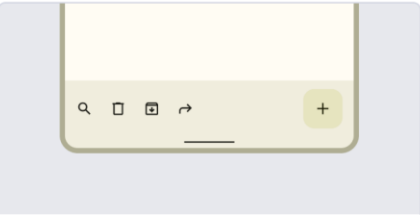
A single fixed-height row that typically contains some text as well as a leading or trailing icon.

Flutter Widgets
Navigation



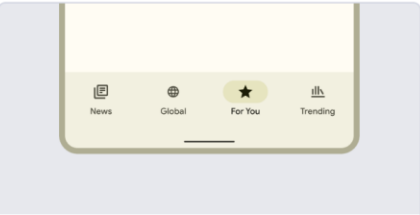
AppBar

Container that displays content and actions at the top of a screen.



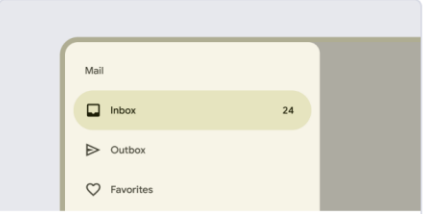
Bottom app bar

Container that displays navigation and key actions at the bottom of a screen.



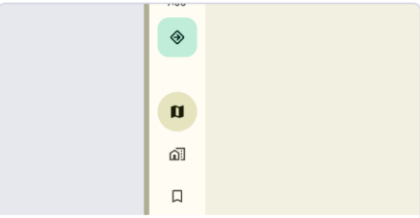
NavigationBar

Persistent container that enables switching between primary destinations in an app.



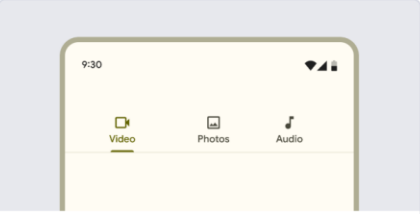
NavigationDrawer

Container that slides from the leading edge of the app to navigate to other sections in an app.



Navigation rail

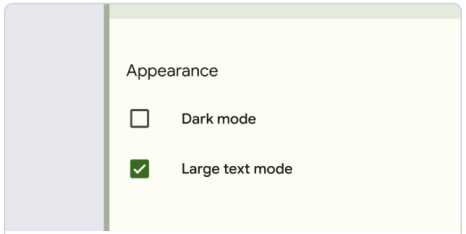
Persistent container on the leading edge of tablet and desktop screens to navigate to parts of an app.



TabBar

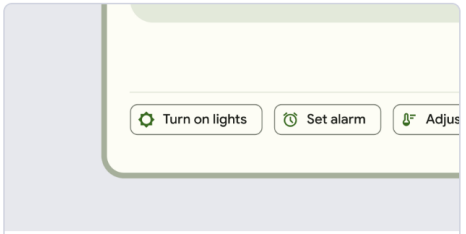
Layered containers that organize content across different screens, data sets, and other interactions.

Flutter Widgets
Selection



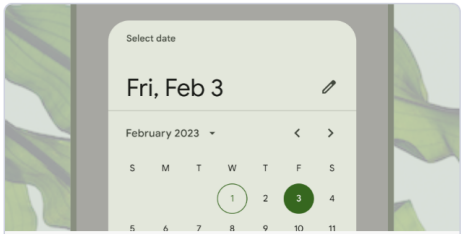
Checkbox

Form control that app users can set or clear to select one or more options from a set.



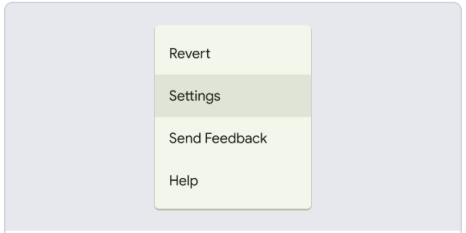
Chip

Small blocks that simplify entering information, making selections, filtering content, or triggering actions.



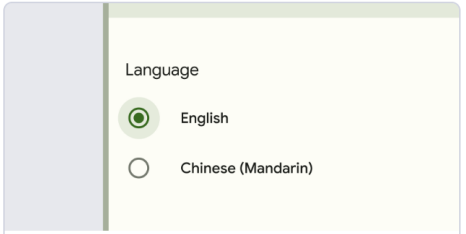
DatePicker

Calendar interface used to select a date or a range of dates.



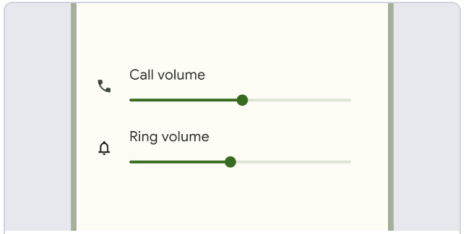
Menu

Container that displays a list of choices on a temporary surface.



Radio

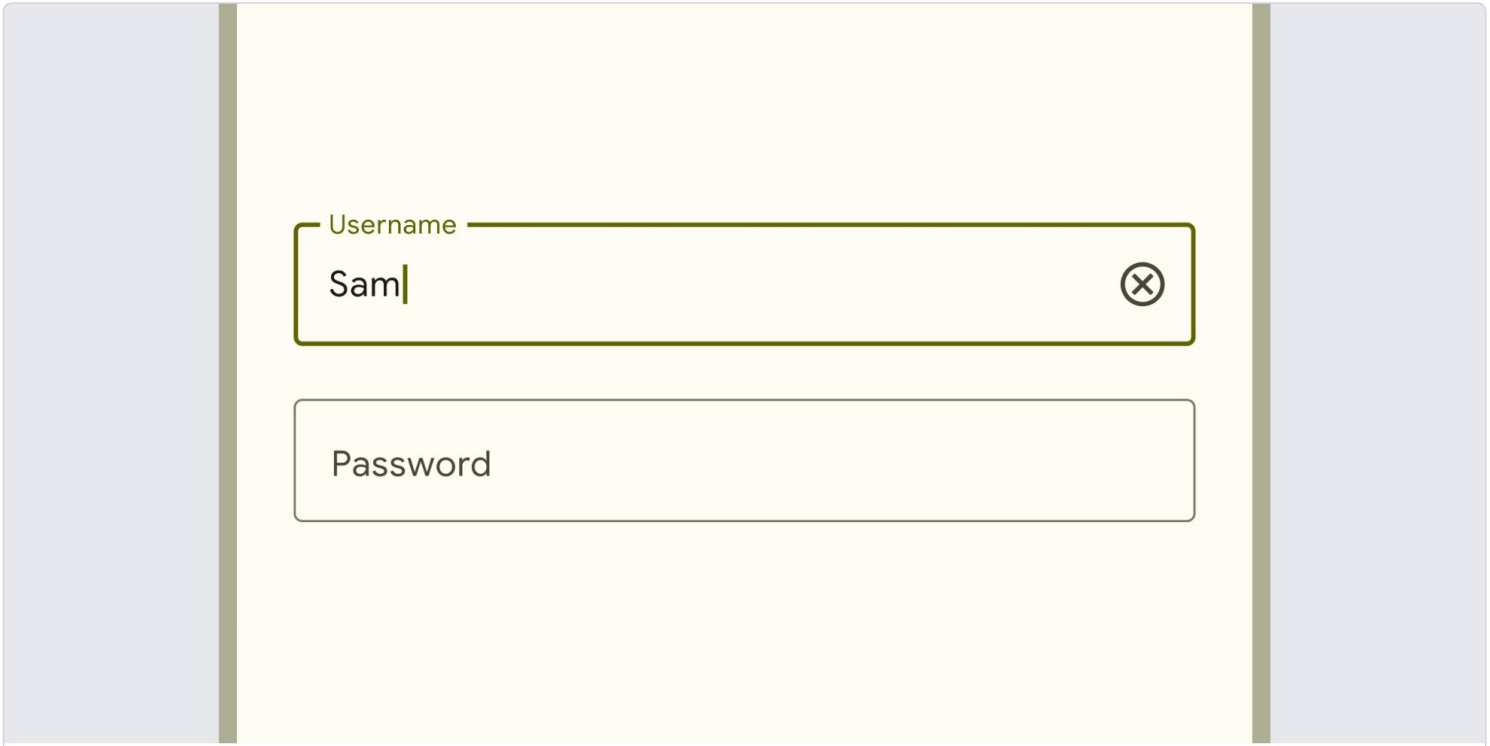
Form control that app users can set or clear to select only one option from a set.



Slider

Form control that enables selecting a range of values.

Flutter Widgets
Text Input

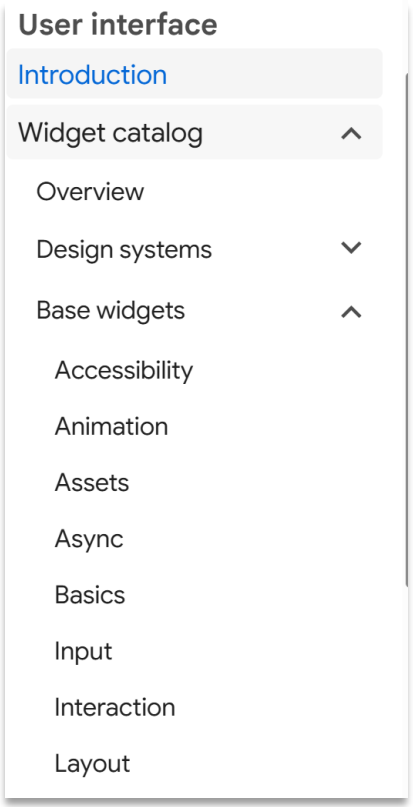


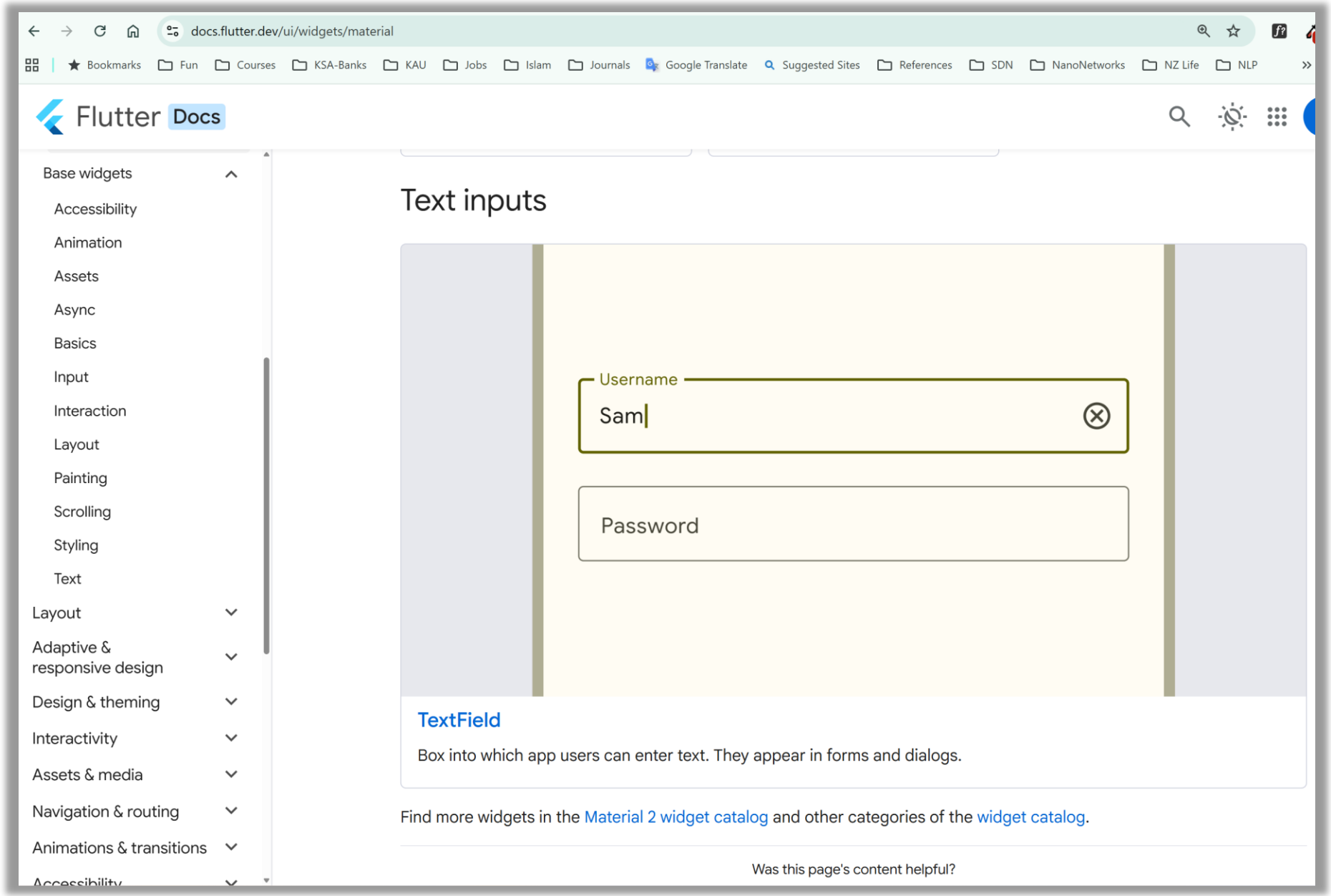
TextField

Box into which app users can enter text. They appear in forms and dialogs.

Widgets

https://docs.flutter.dev/ui





StatefulWidget

```dart
class MyWidget extends StatefulWidget {
  final String name;
  const MyWidget({super.key, required this.name});
  @override
  State<MyWidget> createState() => _MyWidgetState();
}

class _MyWidgetState extends State<MyWidget> {
  int count = 0;
  @override
  Widget build(BuildContext context) {
    return Text('${widget.name}: $count');
  }
}
```

Immutable Widget

StatefulWidget

```
class _MyWidgetState extends State<MyWidget> {
  int count = 0;
  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () {
        setState(() {
          count++;
        });
      },
      child: Text('${widget.name}: $count'),
    );
  }
}
```

Number Game App

- Time for practical demo
- Show two numbers in App
- User click on larger number to get 5 points
- Or loose 5 points if clicked on the smaller number
- Need StatfulWidget

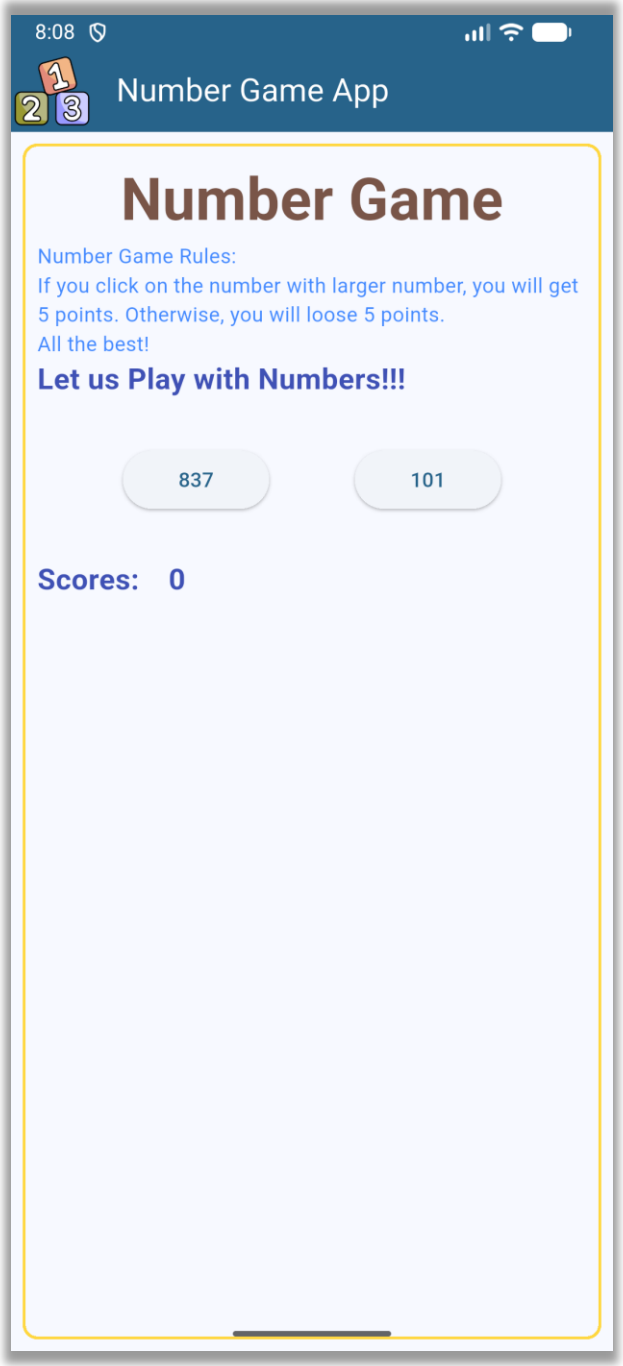


Image Link

References

- https://dart.dev/language
- https://docs.flutter.dev/ui/widgets