

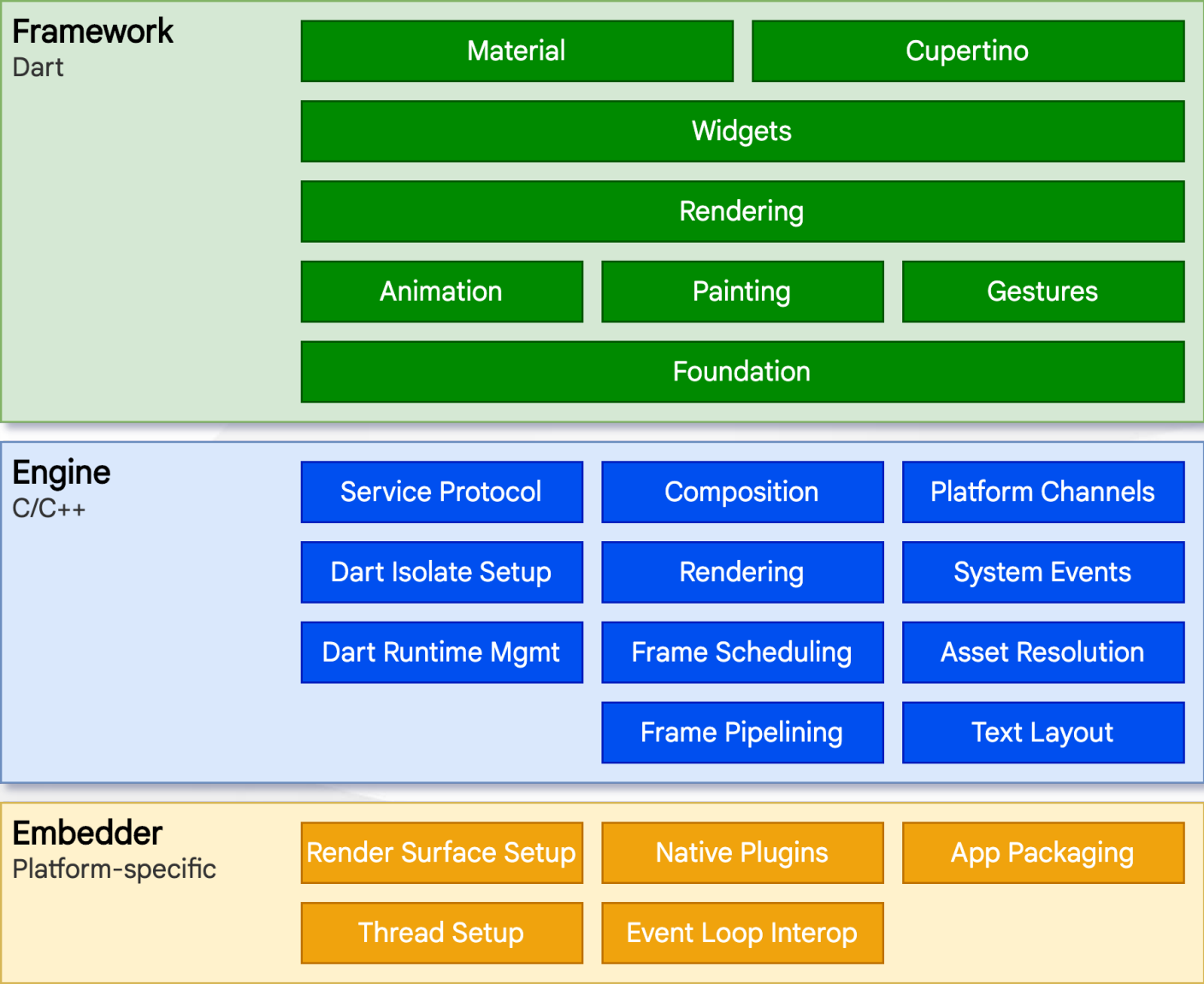


Made by **Google**

# 1. What is Flutter?

**Flutter** is Google's UI toolkit for building beautiful, natively compiled applications for mobile, web, desktop, and embedded devices from a single codebase.

# 1.1. Architectural layers



## 1. What is Flutter?



### 1.2. Get started

Get started now?

<https://flutter.dev/docs/get-started/install>

Coming from another platform?

Docs:

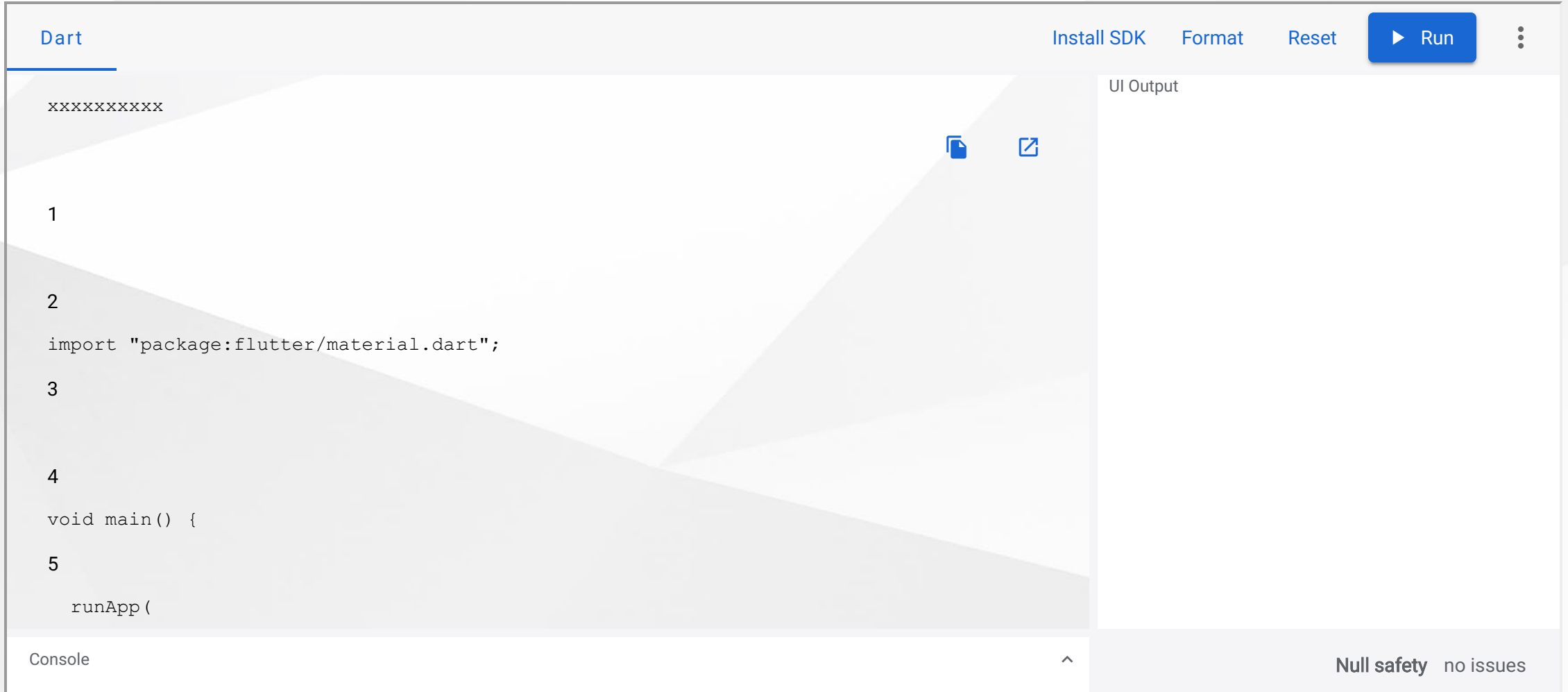
[iOS](#), [Android](#), [Web](#), [React Native](#) and [Xamarin](#).

## 1.3. Try Flutter in your browser

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

void main() {
  runApp(
    const Center(
      child: Text(
        "Hello World!!!",
        textDirection: TextDirection.ltr,
      ),
    ),
  );
}
```

## 1.3. Try Flutter in your browser



The screenshot shows the DartPad web interface. At the top, there's a header with the word "Dart" on the left and a row of buttons on the right: "Install SDK", "Format", "Reset", and a blue "Run" button with a play icon. Below the header is a large code editor area. The code in the editor is as follows:







```
xxxxxxxxxx

1
2
import "package:flutter/material.dart";
3
4
void main() {
5
  runApp(
```

To the right of the code editor is a "UI Output" panel, which is currently empty. Below the code editor is a "Console" panel, also empty. At the bottom right of the interface, there is a status bar that says "Null safety no issues".

## 1.4. Who's using Flutter?

Organizations around the world are building apps with Flutter.

*	*	*
		
		

## 1.4. Who's using Flutter?

See what's being created:



## 2. User Interface

- 2.1. Introduction to widgets
- 2.2. Building layouts
- 2.3. Adding interactivity
- 2.4. Assets & images
- 2.5. Navigation & routing
- 2.6. Animations
- 2.7. Advanced UI
- 2.8. Widget catalog

## 2.1. Introduction to widgets

- Flutter `Widgets` are inspired by React `Components`
- Rendered by their current configuration (or `BuildContext`) and state
- When state changes, it rebuilds
- the framework diffs against the previous description in order to determine the minimal changes needed

## 2.1. Introduction to widgets

- Everything is a Widget
  - But don't put everything in one Widget!
- References:  
<https://romain-rastel.medium.com/everything-is-a-widget-but-dont-put-everything-in-a-widget-32f89b5c8bdb>

Everything Should Be Made as Simple as Possible, But Not Simpler

## 2.1. Introduction to widgets

### Basic widgets:

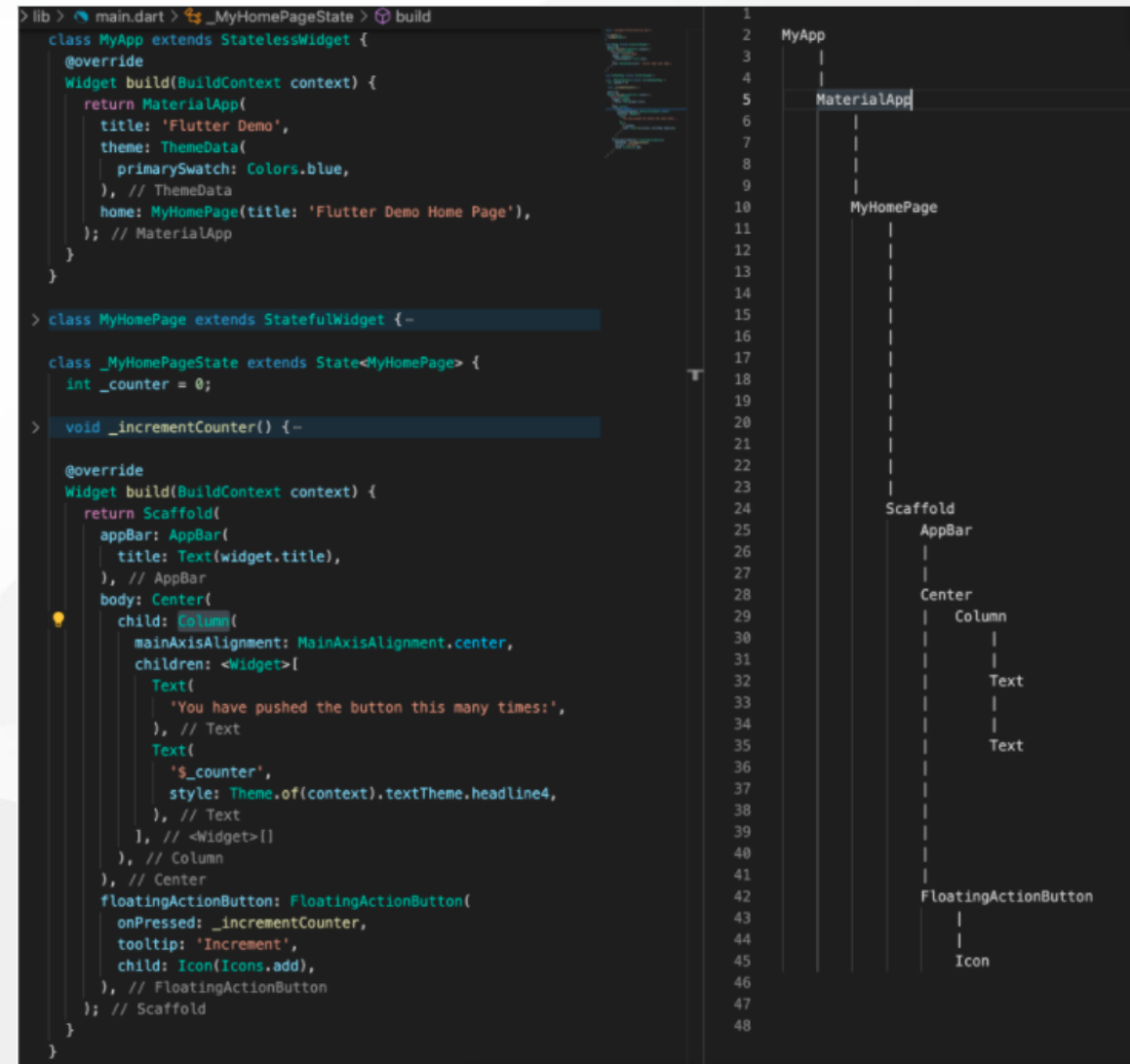
- **Text** - create a run of styled text within your application.
- **Row**, **Column** are flex widgets
- **Stack** - place widgets on top of each other in paint order.
- **Container** - create a rectangular visual element, decorated with a background, a border, or a shadow; also have margins, padding, and constraints applied to its size, ...

... more widgets from there: <https://api.flutter.dev/flutter/widgets/widgets-library.html>

### 2.1. Introduction to widgets

#### Notion of Widgets tree

Widgets are organized in tree structure(s).

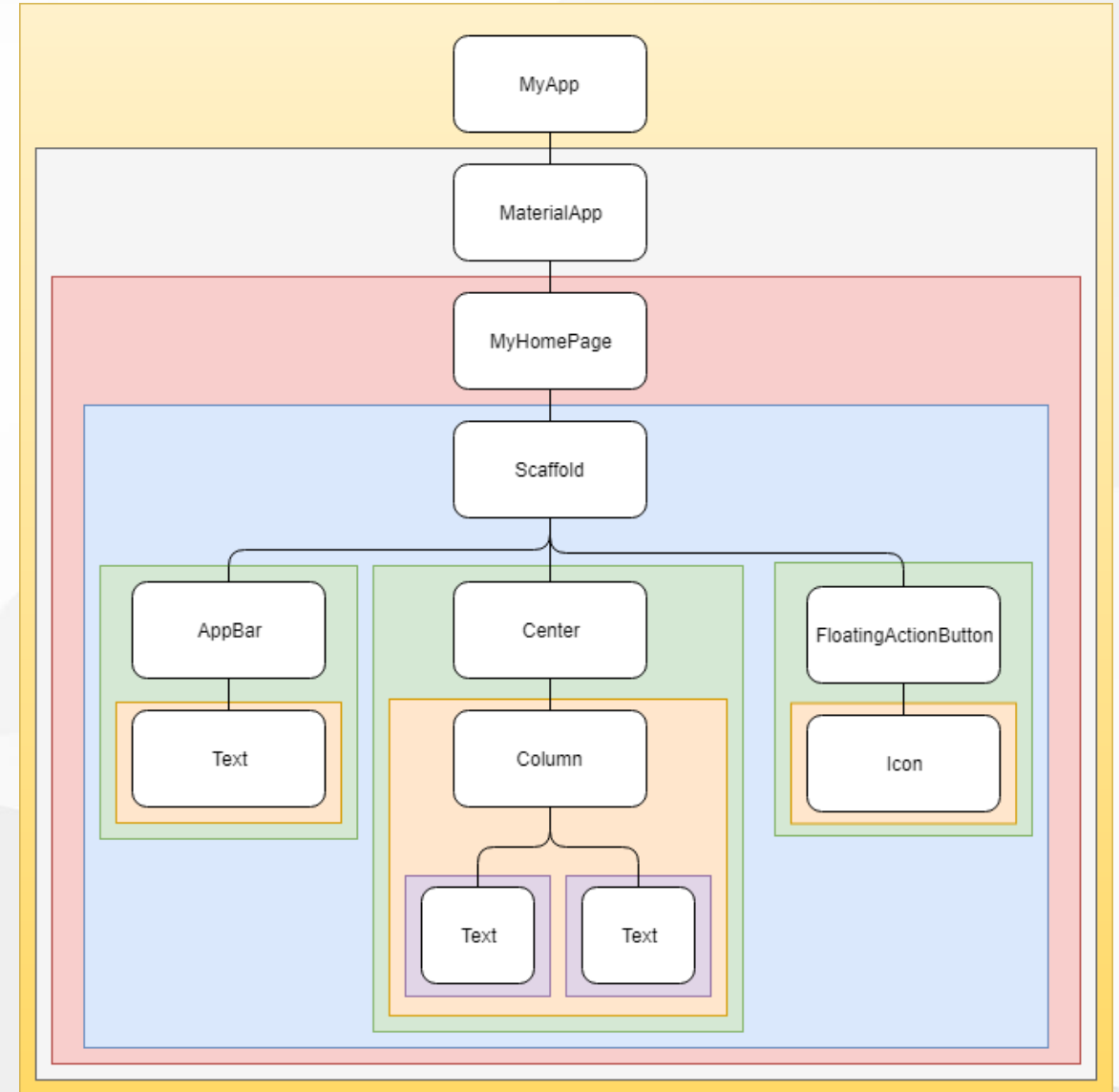


## 2.1. Introduction to widgets

### Notion of Context or BuildContext

Location of a Widget within the tree structure

A context only belongs to one widget.



## 2.1. Introduction to widgets

### Stateful and stateless widgets (1)

StatelessWidget	StatefulWidget
<p>Examples:</p> <ul style="list-style-type: none"><li>- <a href="#">Icon</a></li><li>- <a href="#">IconButton</a></li><li>- <a href="#">Text</a></li></ul>	<p>Examples:</p> <ul style="list-style-type: none"><li>- <a href="#">Checkbox</a></li><li>- <a href="#">Radio</a></li><li>- <a href="#">Slider</a></li><li>- <a href="#">InkWell</a></li><li>- <a href="#">Form</a></li><li>- <a href="#">TextField</a></li></ul>
Super-class: <a href="#">StatelessWidget</a>	Super-class: <a href="#">StatefulWidget</a>

## 2.1. Introduction to widgets

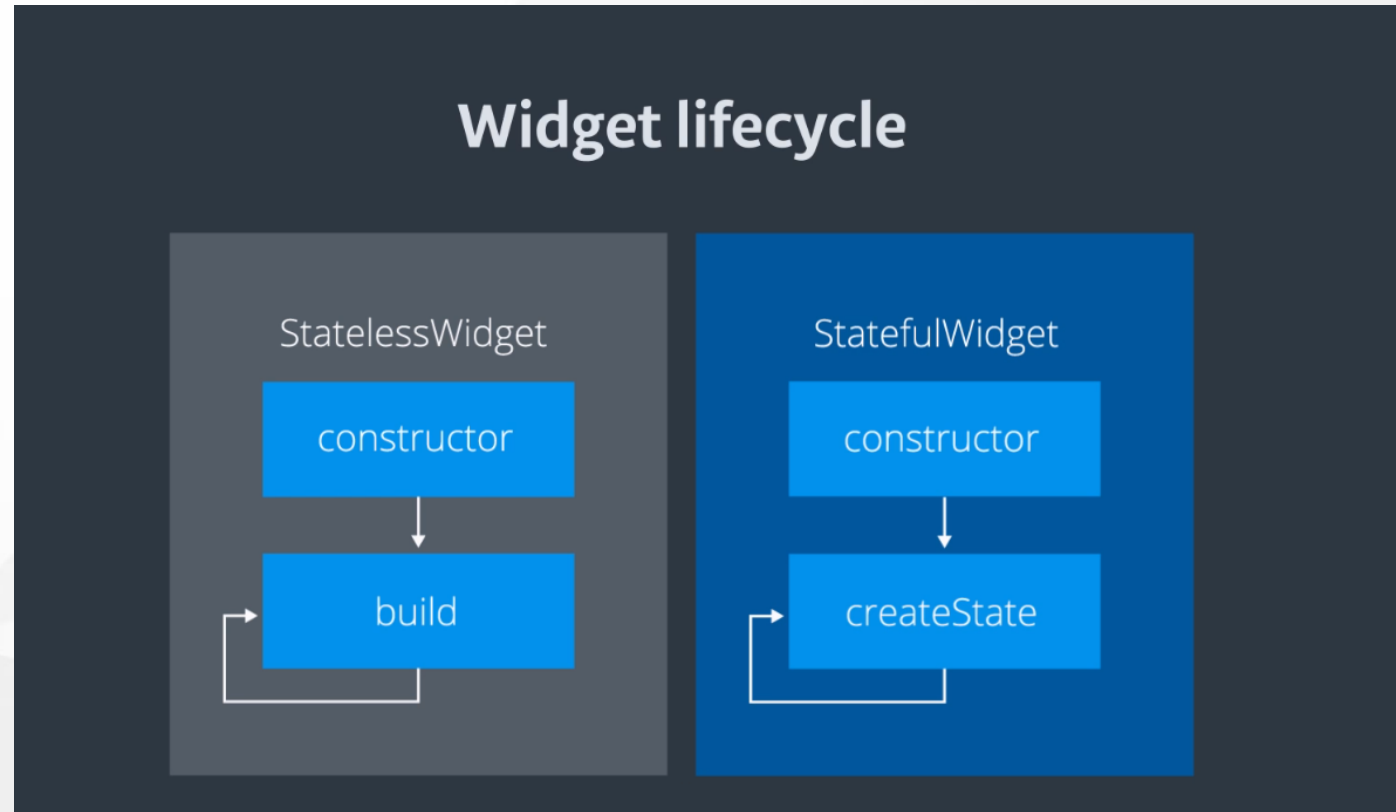
### Stateful and stateless widgets (2)

StatelessWidget	StatefulWidget
Not have to care the state	There are some inner data held and may vary during the lifetime of this widget - called a State



## 2.1. Introduction to widgets

### Widget's Lifecycle (1)

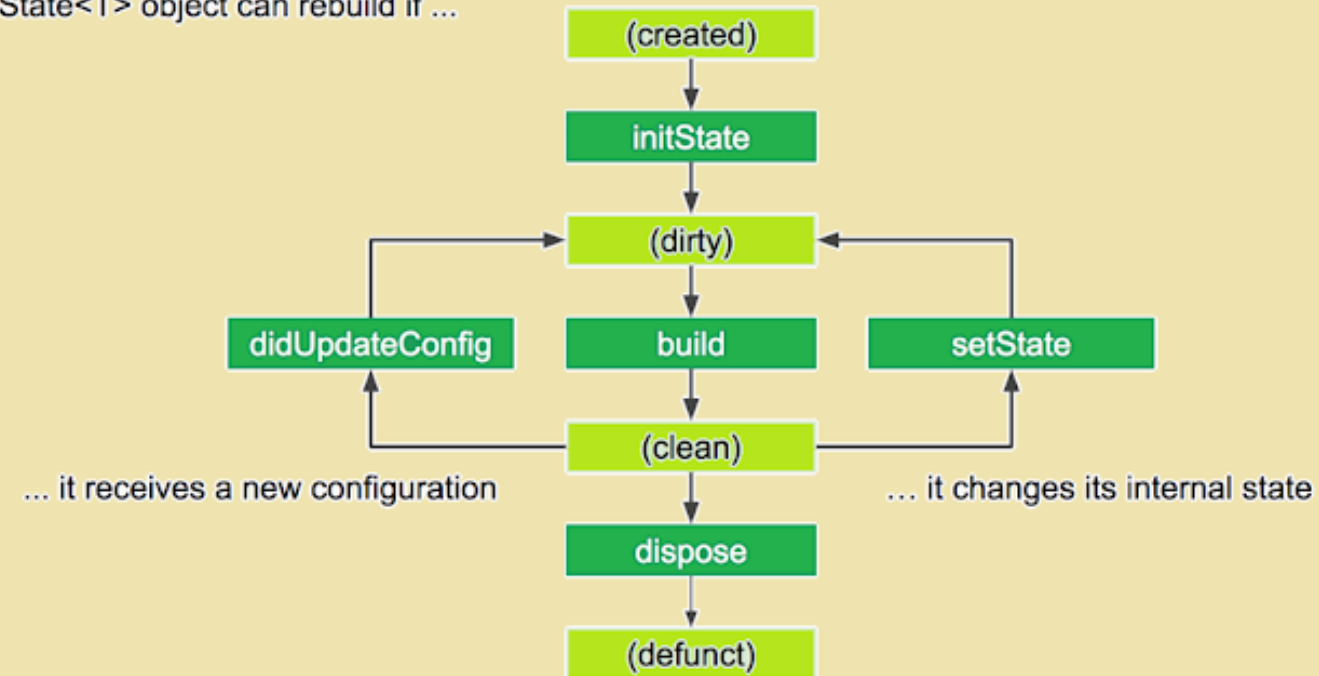


## 2.1. Introduction to widgets

### Widget's Lifecycle (2)

The life cycle of the  
StatefulWidget

A State<T> object can rebuild if ...



## 2.1. Introduction to widgets

### Notion of State

A State defines the `behavioural` part of a `StatefulWidget` instance.

It holds information aimed at interacting / interferring with the Widget in terms of:

- behaviour
- layout

Any changes which is applied to a State forces the Widget to rebuild.

## 2.1. Introduction to widgets

### Relation between a State and a Context

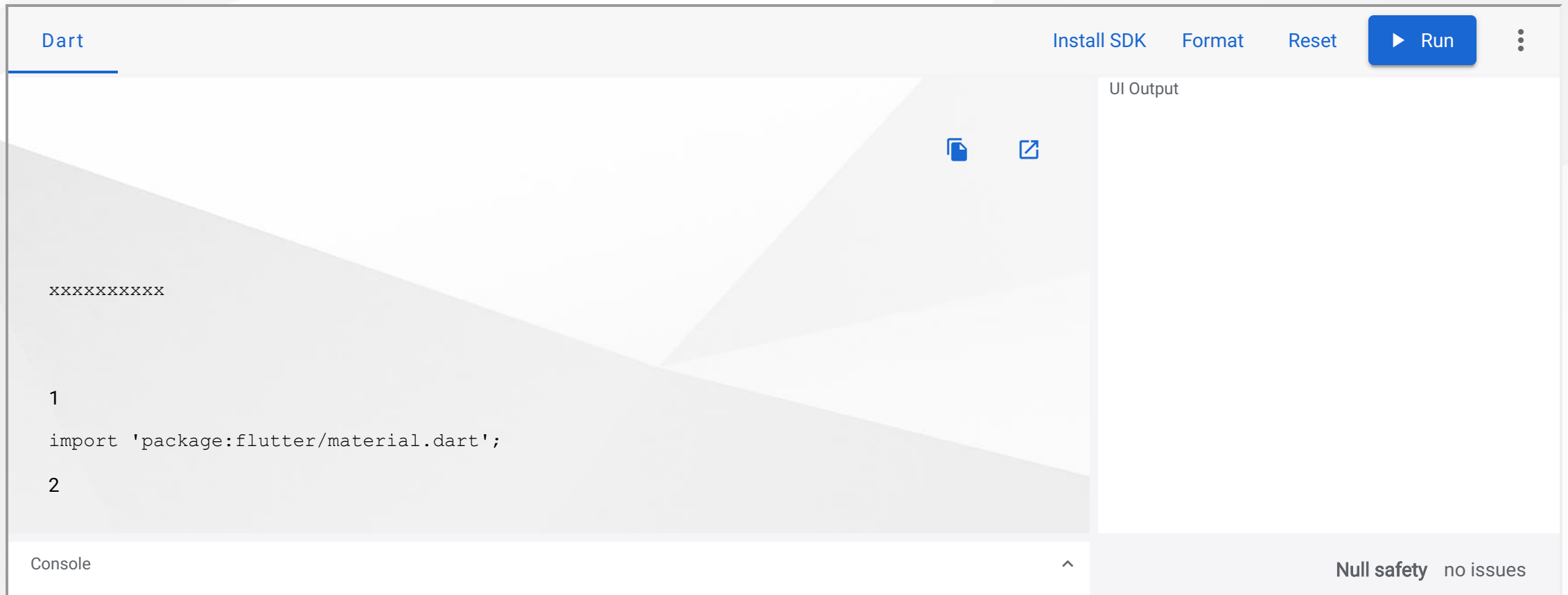
For Stateful widgets, a `State` is associated with a `Context`. This association is permanent and the `State` object will never change its context.

Even if the Widget Context can be moved around the tree structure, the State will remain associated with that context.

When a State is associated with a Context, the State is considered as mounted.

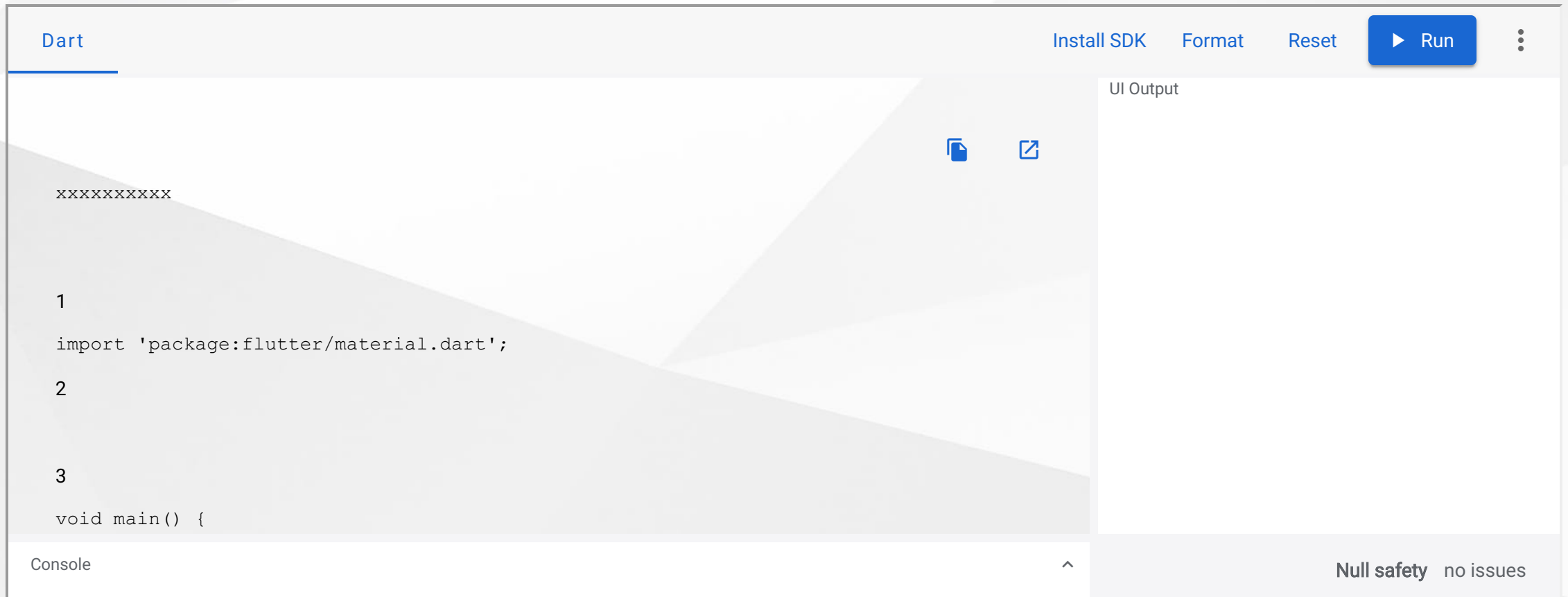
## 2.1. Introduction to widgets

### Standard Code of A StatelessWidget



## 2.1. Introduction to widgets

### Standard Code of A StatefulWidget



## 2.1. Introduction to widgets

### Keys

- Use `keys` to control which widgets are `rebuilt`
- For example in builds a list items in `ListView`:
  - Without `keys`, the item is rebuilt even if it is no longer visible in viewport.
  - By assigning each entry in the list a “semantic” key, only the items visible in the view will be rebuilt.

For more information, see the [Key](#) API.

## 2.1. Introduction to widgets

### Global keys

- To uniquely identify child widgets.
- Must be globally unique across the entire widget hierarchy.
- Can be used to retrieve the state associated with a widget.

For more information, see the [GlobalKey](#) API.



## 2.2. Building layouts

- Layouts in Flutter
- Tutorial
- Creating adaptive and responsive apps
- Understanding constraints
- Box constraints

# 2.2.1 Layouts in Flutter

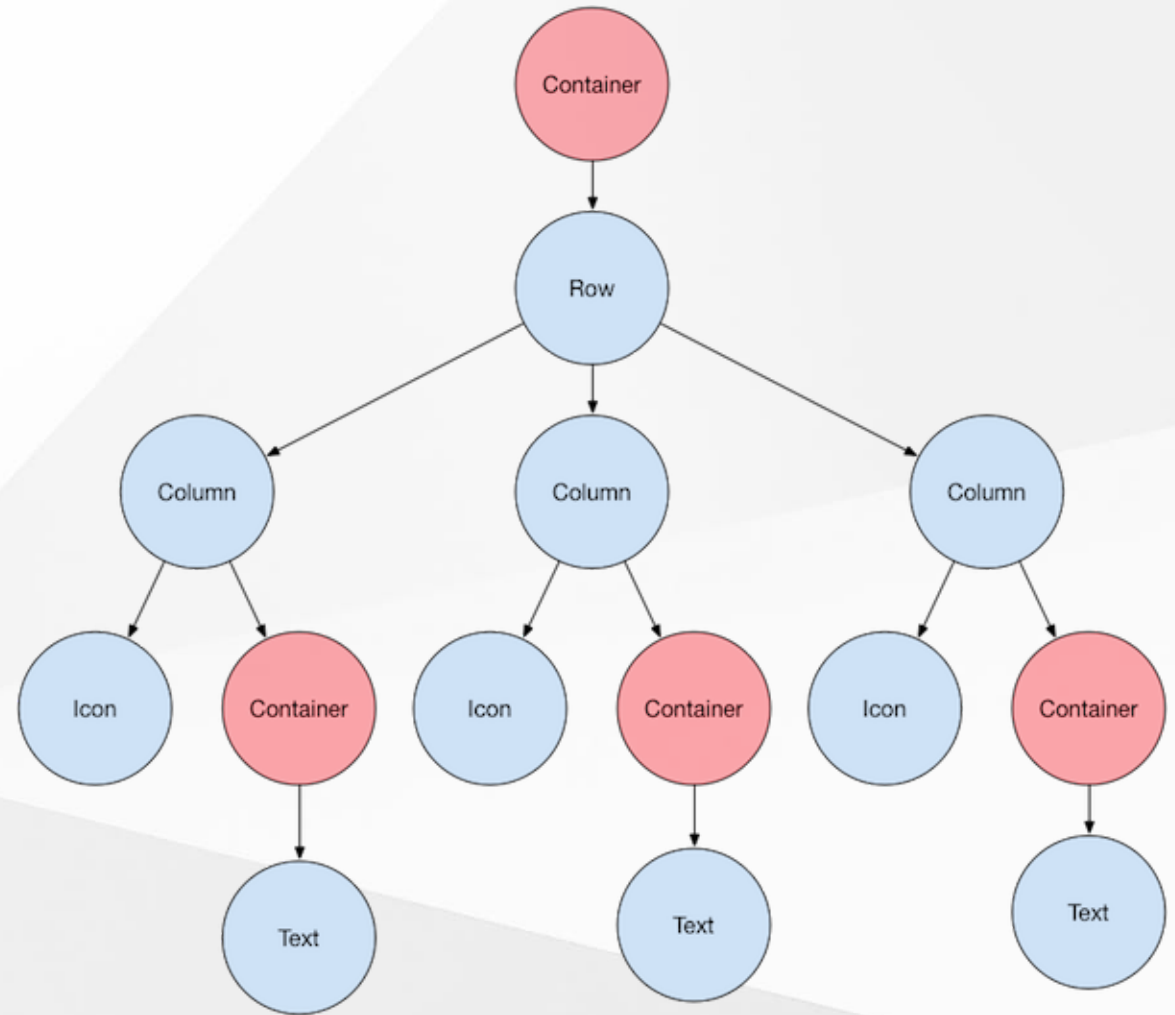
## Example (1)

Design	Visual Layout
<div><div>CALL</div><div>ROUTE</div><div>SHARE</div></div>	<div><div>CALL</div><div>ROUTE</div><div>SHARE</div></div>

## 2.2.1 Layouts in Flutter

### Example (2)

### Widgets Tree



## 2.2.1 Layouts in Flutter

**Design Languages libraries built-in:**

- [Material](#) - Google Material Design
- [Cupertino](#) - iOS Design Language

## 2.2.1 Layouts in Flutter

### Common layout widgets:

#### Standard widgets

- **Container**: Adds padding, margins, borders, background color, or other decorations to a widget.
- **GridView**: Lays widgets out as a scrollable grid.
- **ListView**: Lays widgets out as a scrollable list.
- **Stack**: Overlaps a widget on top of another.

## 2.2.1 Layouts in Flutter

### Common layout widgets:

#### Material widgets

- **Card**: Organizes related info into a box with rounded corners and a drop shadow.
- **ListTile**: Organizes up to 3 lines of text, and optional leading and trailing icons, into a row.

## 2.2.2 Tutorial

Flutter 2.5 is released to stable! For details, see [What's new in Flutter 2.5](#).

# Building layouts

[Docs](#) > [Development](#) > [UI](#) > [Layout](#) > [Tutorial](#)

## Contents

[Step 0: Create the app base code](#)

[Step 1: Diagram the layout](#)

[Step 2: Implement the title row](#)

[Step 3: Implement the button row](#)

[Step 4: Implement the text section](#)

[Step 5: Implement the image section](#)

[Step 6: Final touch](#)

[Get started](#)

[Samples & tutorials](#)

[Development](#)

▼ [User interface](#)

[Introduction to widgets](#)

▼ [Building layouts](#)

[Layouts in Flutter](#)

[Tutorial](#)

[Creating adaptive and responsive apps](#)

## 2.2.3 Creating adaptive and responsive apps

### Difference between Adaptive and Responsive app

- Adaptive and responsive can be viewed as separate dimensions of an app
- Responsive
  - Typically, a responsive app has had its layout tuned for the available screen size...
  - [Create a responsive app](#)
- Adaptive
  - Adapting an app to run on different device types, such as mobile and desktop, requires dealing with mouse and keyboard input, ...
  - [Building adaptive apps](#)



## 2.2.3 Understanding constraints

Flutter 2.5 is released to stable! For details, see [What's new in Flutter 2.5](#).

# Understanding constraints

[Docs](#) > [Development](#) > [UI](#) > [Layout](#) > [Understanding constraints](#)

[Get started](#)

[Samples & tutorials](#)

[Development](#)

▼ [User interface](#)

[Introduction to widgets](#)

▼ [Building layouts](#)

[Layouts in Flutter](#)

[Tutorial](#)

[Creating adaptive and responsive apps](#)



## 2.3. Adding interactivity

Flutter 2.5 is released to stable! For details, see [What's new in Flutter 2.5](#).

# Adding interactivity to your Flutter ap

[Get started](#)



[Samples & tutorials](#)



[Development](#)



▼ [User interface](#)

[Introduction to widgets](#)

▶ [Building layouts](#)

[Adding interactivity](#)

[Assets and images](#)

▶ [Navigation & routing](#)

▶ [Animations](#)

[Docs](#) > [Development](#) > [UI](#) > [Adding interactivity](#)

### Contents

[Stateful and stateless widgets](#)

[Creating a stateful widget](#)

[Step 0: Get ready](#)

[Step 1: Decide which object manages the widget's state](#)

[Step 2: Subclass StatefulWidget](#)

[Step 3: Subclass State](#)

[Step 4: Plug the stateful widget into the widget tree](#)

[Problems?](#)

## 2.4. Adding assets and images

Flutter 2.5 is released to stable! For details, see [What's new in Flutter 2.5](#).

# Adding assets and images

[Docs](#) > [Development](#) > [UI](#) > [Assets and images](#)

## Contents

[Specifying assets](#)

[Asset bundling](#)

[Asset variants](#)

[Loading assets](#)

[Loading text assets](#)

[Loading images](#)

[Declaring resolution-aware image assets](#)

[Get started](#)

[Samples & tutorials](#)

[Development](#)

▼ [User interface](#)

[Introduction to widgets](#)

▶ [Building layouts](#)

[Adding interactivity](#)

[Assets and images](#)

▶ [Navigation & routing](#)

▶ [Animations](#)

## 2.5. Navigation and routing (1)

Two approaches:

- Imperative approach, Navigation v1.0
  - see the [Navigation recipes](#)
  - Or using [Fluro](#) package
- Declarative approach, Navigation v2.0
  - [Learning Flutter's new navigation and routing system](#)
  - Alternate packages:
    - [vrout](#)
    - [beamer](#) (not stable)

## 2.5. Navigation and routing (2)

### Deep linking:

- Examples:
  - `http://flutterbooksample.com/book/1`
  - `customscheme://flutterbooksample.com/book/1`

### URL strategy on the web

- Hash (default)  
For example, `flutterexample.dev/#/path/to/screen`.
- Path  
For example, `flutterexample.dev/path/to/screen`.

## 2.5. Navigation and routing (3)

### Fluro

- Simple route navigation
- Function handlers (map to a function instead of a route)
- Wildcard parameter matching
- Querystring parameter parsing
- Common transitions built-in
- Simple custom transition creation
- Follows stable Flutter channel
- Null-safety

## 2.5. Navigation and routing (4)

### **VRouter** (for reference only)

- Automated web url handling
- Nesting routes
- Transition
- Advanced url naming
- Reacting to route changing
- Customizable pop events
- And much more...

## 2.6. Animations

### Approaches:

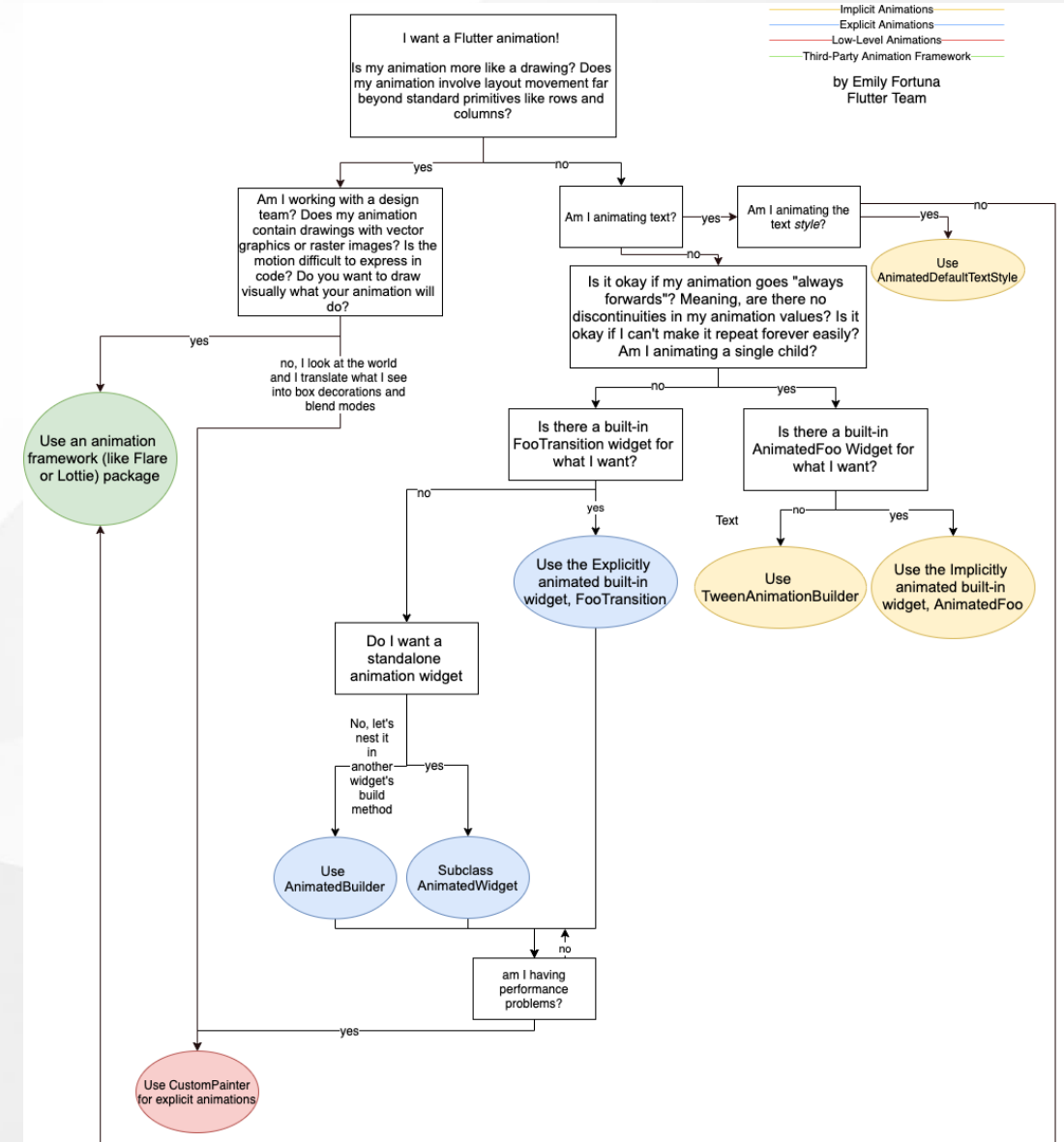
- Implicit Animations
- Explicit Animations
- Low-Level Animation
  - draw it with canvas via `CustomPainter`
- Third-party animation framework
  - `flare_flutter`
  - `lottie`



## 2.6. Animations

Full picture: [Click here](#)

Video: [How to choose which Flutter Animation Widget is right for you?](#)



## 2.6. Animations

### Common animation patterns

- Animated list or grid
- Shared element transition
  - Shared element transitions between routes (pages)
  - **Hero animations**
- **Staggered animation**
  - Animations that are broken into smaller motions, where some of the motion is delayed.
  - The smaller animations might be sequential, or might partially or completely overlap.

## 2.7. Advanced UI

- Using Actions and Shortcuts
- Gestures
- Slivers
- Splash screens

## 2.8. Widget catalog

Flutter 2.5 is released to stable! For details, see [What's new in Flutter 2.5.](#)

Get started

Samples & tutorials

Development

▼ User interface

Introduction to widgets

▶ Building layouts

Adding interactivity

Assets and images

▶ Navigation & routing

Animations

# Widget catalog

[Docs](#) > [Development](#) > [UI](#) > [Widgets](#)

Create beautiful apps faster with Flutter's collection of visual, structural, platform, and interactive widgets. In addition to widgets by category, you can also see all the widgets in the [widget index](#).

[Accessibility](#)

Make your app accessible.

[Visit](#)

[Animation and Motion](#)

Bring animations to your app.

[Visit](#)

[Assets, Images, and Icons](#)

Manage assets, display, and show icons.

[Visit](#)

<https://flutter.dev/docs/development/ui/widgets>

44

## 3. State management

- 3.1. [Introduction](#)
- 3.2. [Think declaratively](#)
- 3.3. [Ephemeral vs app state](#)
- 3.4. [Simple app state management](#)
- 3.5. [Options](#)
- 3.6. [Riverpod](#)

## 4. Data & Networking

- 3.1. Cross-platform http networking
- 3.2. [Networking cookbook](#)
- 3.3. [JSON and serialization](#)
- 3.4. OpenAPI and generate Data Provider
- 3.5. [Firebase](#)

## 5. Internationalization

Flutter 2.5 is released to stable! For details, see [What's new in Flutter 2.5](#).

# Internationalizing Flutter apps

[Docs](#) > [Development](#) > [a11y & i18n](#) > [i18n](#)

## Contents

[Introduction to localizations in Flutter](#)

[Setting up an internationalized app: the Flutter\\_localizations package](#)

[Adding your own localized messages](#)

[Localizing for iOS: Updating the iOS app bundle](#)

[Advanced topics for further customization](#)

[Advanced locale definition](#)

[Tracking the locale: The Locale class and the Localizations widget](#)

[Get started](#)

[Samples & tutorials](#)

[Development](#)

▶ [User interface](#)

▶ [Data & backend](#)

▼ [Accessibility & internationalization](#)

[Accessibility](#)

[Internationalization](#)

▶ [Platform integration](#)

## Other References and ebooks (1)

- Flutter Complete Reference
  - Official website: <https://fluttercompletereference.com/>
  - [Full version](#)
  - [Preview version](#)
- Performance & optimization
  - [App Size](#)
  - [Deferred components](#)
- [Platform-specific behaviors and adaptations](#)



## Other References and ebooks (2)

- [Widget index](#)
- [API reference](#)
- [flutter CLI reference](#)
- [Package site](#)
- [FAQ](#)

Thank you