ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо видно && слышно?

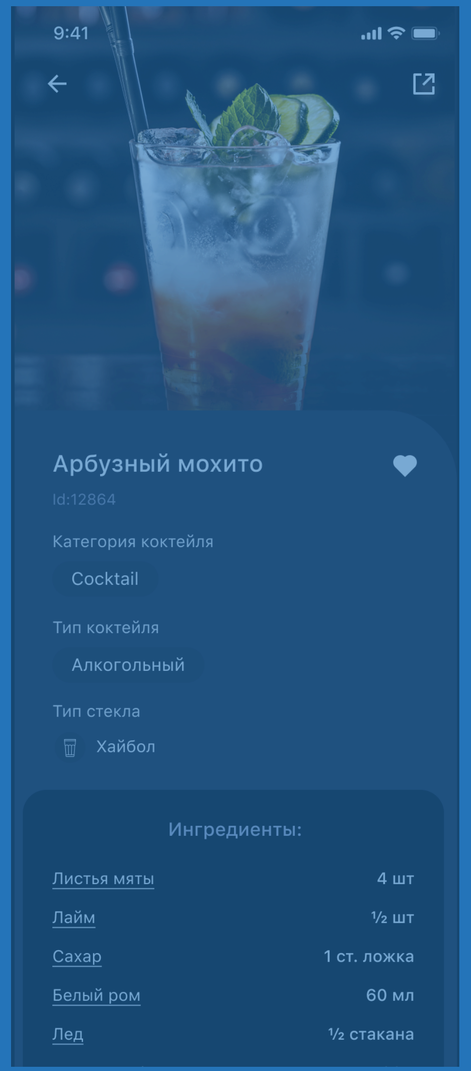
Ставьте + , если все хорошо
Напишите в чат, если есть проблемы

Предыдущая тема: Dart. Основы

<to discuss> min to code and share

1.Создаем flutter (dart) package cocktail_app_models
2.Создаем lib/src
3.Создаем класс коктейля в src
4.Формируем файл экспорта в lib
5.Создаем flutter application cocktail_app
6.Импортируем package с моделью в pubspec.yaml
7.Создаем в main() инстанс модели коктейля

Это **не** домашнее задание



https://rapidapi.com/thecocktaildb/api/the-cocktail-db

Dart

Flutter

Announcing Dart 2.10

Announcing Flutter 1.22

```
❯ dartanalyzer .
Analyzing console-simple...
  lint • The function main should have a return type but doesn't. • bin/flow.dart:4:1 • always_declare_return_types
  lint • Only use double quotes for strings containing single quotes. • bin/metrics.dart:4:29 • prefer_single_quotes
  lint • Only use double quotes for strings containing single quotes. • bin/metrics.dart:4:39 • prefer_single_quotes
  hint • The value of the local variable 'isNotUsedVar' isn't used. • bin/main.dart:2:8 • unused_local_variable
3 lints and 1 hint found.
❯ dart analyze
Analyzing console-simple...                1.1s

   info • The function main should have a return type but doesn't at bin/flow.dart:4:1 • (always_declare_return_types)
   info • The value of the local variable 'isNotUsedVar' isn't used at bin/main.dart:2:8 • (unused_local_variable)
   info • Only use double quotes for strings containing single quotes at bin/metrics.dart:4:29 • (prefer_single_quotes)
   info • Only use double quotes for strings containing single quotes at bin/metrics.dart:4:39 • (prefer_single_quotes)

4 issues found.
❯ ~/otus/dart-examples/cli/console-simple
```

Flutter Mobile Developer

Тема 2. Flutter. Как оно устроено. Пишем и запускаем первое приложение

Цель урока

- Рассмотреть устройство Flutter Engine;
- Правильно выбрать необходимый тип виджета для выполнения задачи;
- Запустить Flutter-приложение на реальном устройстве или симуляторе.

О чем будем говорить

- Что такое виджет Flutter?
- Stateless & Stateful widgets
- Inherited Widgets
- Keys & Global Keys
- Flutter Engine

Что такое виджет Flutter?

The central idea is that **you build your UI out of widgets.**

Widgets describe what their view should look like given their current configuration and state.

When a widget's state changes, the widget rebuilds its description, which the framework diffs against the previous description in order to determine the minimal changes needed in the underlying render tree to transition from one state to the next.

Widget catalog

Create beautiful apps faster with Flutter's collection of visual, structural, platform, and interactive widgets. In addition to browsing 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 images, and show icons.

Visit

Async

Async patterns to your Flutter application.

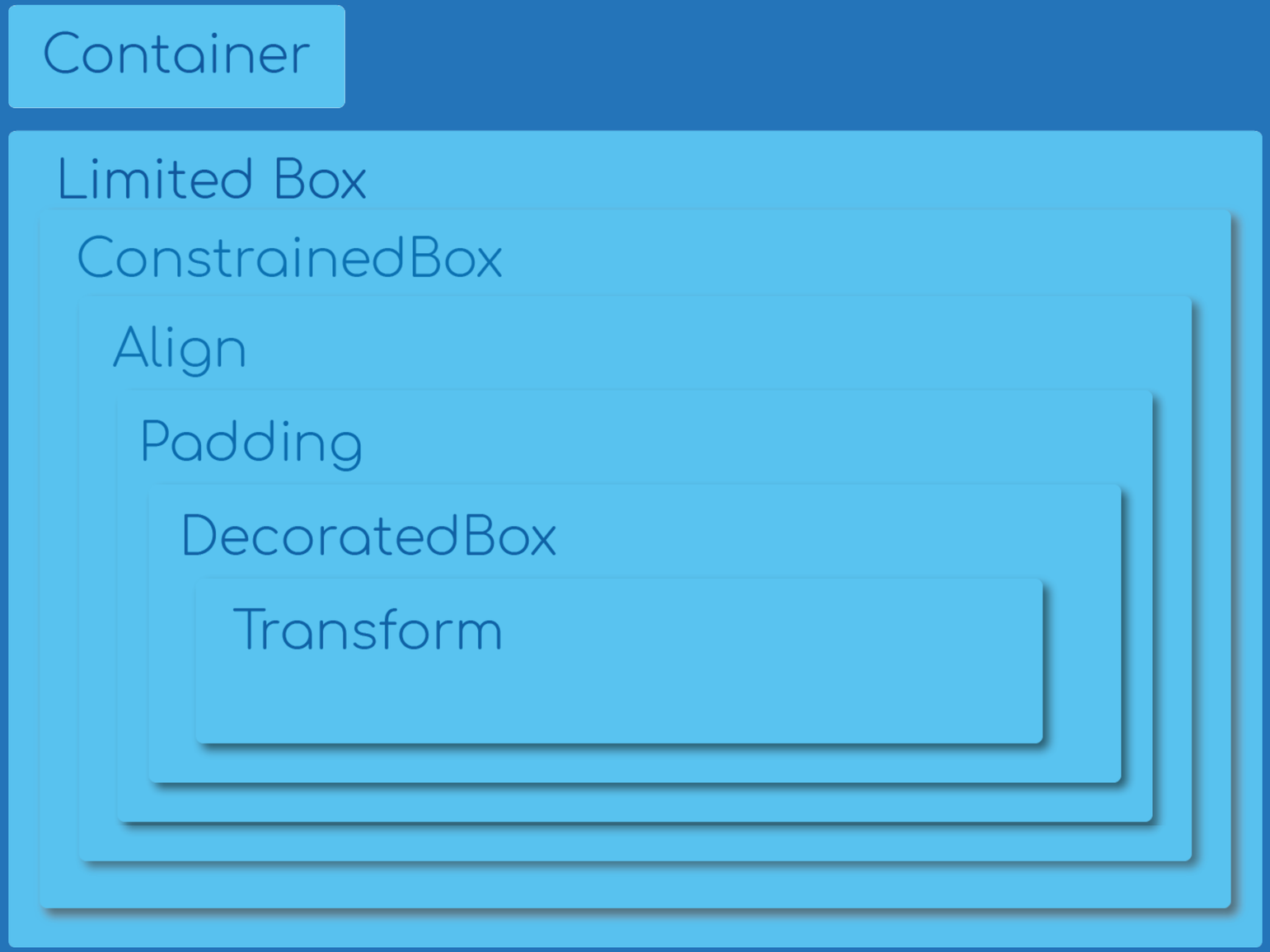
Basics

Widgets you absolutely need to know before building your first Flutter app.

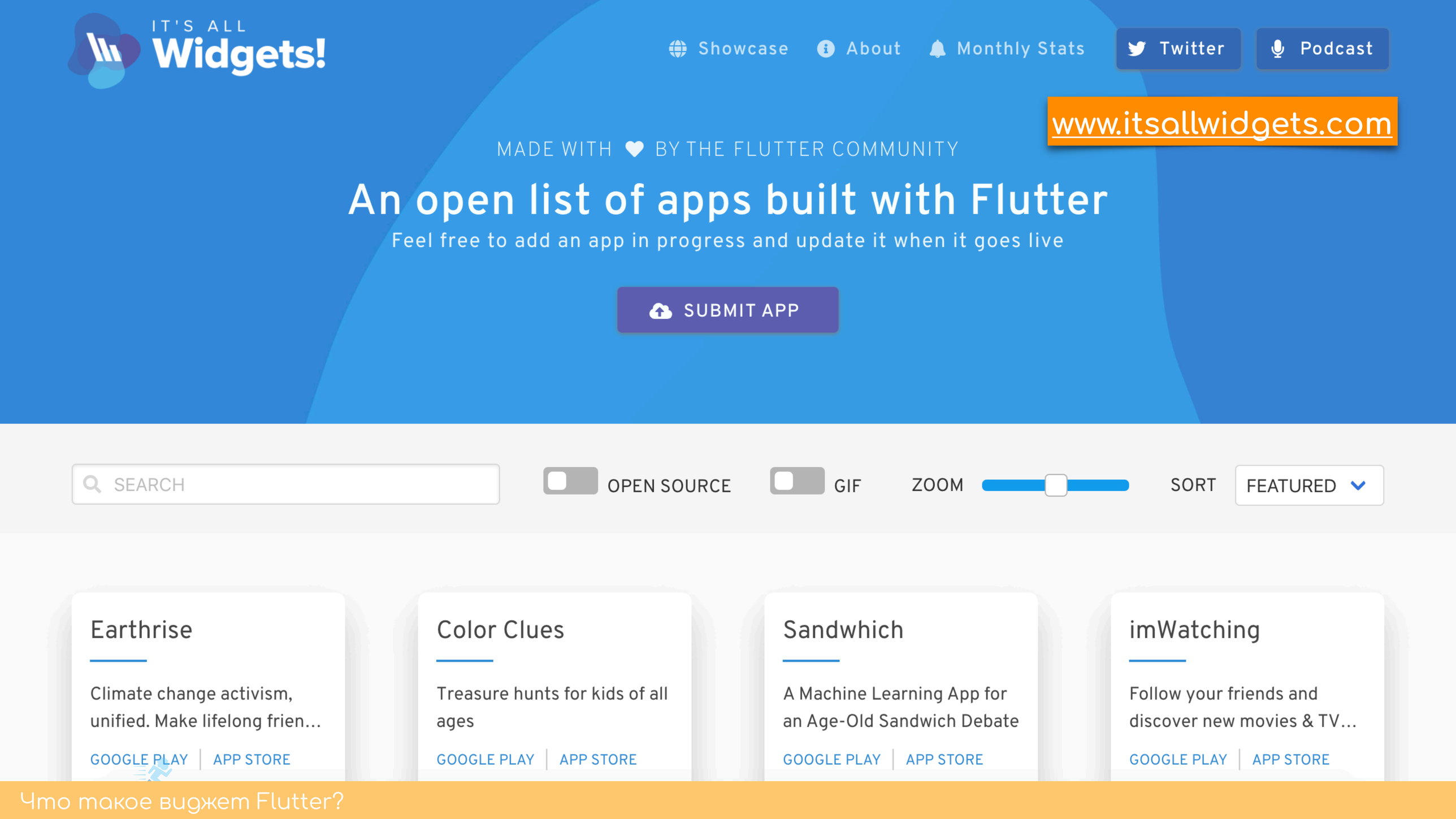
Cupertino (iOS-style widgets)

Beautiful and high-fidelity widgets for current iOS design language.

Widget is not UI Component



```
  if (foregroundDecoration != null) {
    current = DecoratedBox(
      decoration: foregroundDecoration,
      position: DecorationPosition.foreground,
      child: current,
    );
  }

  if (constraints != null)
    current = ConstrainedBox(constraints: constraints, child: current);

  if (margin != null)
    current = Padding(padding: margin, child: current);

  if (transform != null)
    current = Transform(transform: transform, child: current);

  return current;
}
```


IT'S ALL
Widgets!
Showcase
About
Monthly Stats
Twitter
Podcast
www.itsallwidgets.com
MADE WITH ♥ BY THE FLUTTER COMMUNITY
An open list of apps built with Flutter
Feel free to add an app in progress and update it when it goes live
SUBMIT APP
SEARCH
OPEN SOURCE
GIF
ZOOM
SORT
FEATURED
Earthrise
Climate change activism, unified. Make lifelong frien...
GOOGLE PLAY | APP STORE
Color Clues
Treasure hunts for kids of all ages
GOOGLE PLAY | APP STORE
Sandwhich
A Machine Learning App for an Age-Old Sandwich Debate
GOOGLE PLAY | APP STORE
imWatching
Follow your friends and discover new movies & TV...
GOOGLE PLAY | APP STORE

О чем будем говорить

Что такое виджет Flutter?

Stateless & Stateful widgets

Inherited Widgets

Keys & Global Keys

Flutter Engine

Stateless & Stateful widgets

- Demo - Hello World and own widgets!

Stateless & Stateful widgets

- Demo - ex_01_hello_world

```
4   void main() {
5     runApp(
6       Container(
7         decoration: BoxDecoration(shape: BoxShape.circle, color: Colors.white),
8         alignment: Alignment.center,
9         child: Text(
10          'Hello, world!',
11          textDirection: TextDirection.ltr,
12          style: TextStyle(color: Colors.black, fontSize: 40),
13        ), // Text
14      ), // Container
15    );
16  }
17
```

Stateless & Stateful widgets

• Demo - ex_02_hello_world_refactor

```
4   void main() {
5     runApp(
6       MyOwnWidget(),
7     );
8   }
9
10  class MyOwnWidget extends StatelessWidget {
11    const MyOwnWidget({
12      Key key,
13    }) : super(key: key);
14
15    @override
16    Widget build(BuildContext context) {
17      return Container(
18        decoration: BoxDecoration(shape: BoxShape.circle, color: Colors.white),
19        alignment: Alignment.center,
20        child: Text(
21          'Hello, world!',
22          textDirection: TextDirection.ltr,
23          style: TextStyle(color: Colors.black, fontSize: 40),
24        ), // Text
```

Widget build(BuildContext context)

Все построение вашего UI происходит в методе build(context)

```
@override
Widget build(BuildContext context) {
  return Container(
    decoration: BoxDecoration(shape: BoxShape.circle, color: Colors.white),
    alignment: Alignment.center,
    child: Text(
      'Hello, world!',
      textDirection: TextDirection.ltr,
      style: TextStyle(color: Colors.black, fontSize: 40),
    ), // Text
  ); // Container
}
}
```

Gestures Detector

• Demo - ex_03_hello_world_gestures

```
15    @override
16    Widget build(BuildContext context) {
17      return GestureDetector(
18        onTap: (){
19          print('tapped!');
20        },
21        child: Container(
22          decoration: BoxDecoration(shape: BoxShape.circle, color: Colors.white),
23          alignment: Alignment.center,
24          child: Text(
25            'Hello, world!',
26            textDirection: TextDirection.ltr,
27            style: TextStyle(color: Colors.black, fontSize: 40),
28          ), // Text
29        ), // Container
30      ); // GestureDetector
31    }
32  }
```

Stateful Widgets & State

- Demo - ex_04_hello_world_stateful

```dart
class MyOwnWidget extends StatefulWidget {
  @override
  _MyOwnWidgetState createState() => _MyOwnWidgetState();
}

class _MyOwnWidgetState extends State<MyOwnWidget> {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

setState(() {})

```
/// Notify the framework that the internal state of this object has changed.
/// ...
/// You can determine whether it is legal to call this method by checking
/// whether the [mounted] property is true.
@protected
void setState(VoidCallback fn)
```

State lifecycle

- Demo - ex_05_hello_world_state_lifecycle

```
14  class _MyOwnWidgetState extends State<MyOwnWidget> {
15    bool _isShapeCircle;
16
17    @override
18    void initState() {
19      super.initState();
20      _isShapeCircle = true;
21    }
22
23    @override
24    void dispose() => super.dispose();
25
26    @override
27    void didUpdateWidget(MyOwnWidget oldWidget) => super.didUpdateWidget(oldWidget);
28
29    @override
30    Widget build(BuildContext context) => GestureDetector(
31          onTap: () => setState(() => _isShapeCircle = !_isShapeCircle),
```

initState()

```
/// Called when this object is inserted into the tree.
///
/// The framework will call this method exactly once for each [State] object
/// it creates.
@protected
@mustCallSuper
void initState()
```

didUpdateWidget(T oldWidget)

```
/// Called whenever the widget configuration changes.
///
/// If the parent widget rebuilds and request that this location in the tree
/// update to display a new widget with the same [runtimeType] and
/// [Widget.key], the framework will update the [widget] property of this
/// [State] object to refer to the new widget and then call this method
/// with the previous widget as an argument.
@mustCallSuper
@protected
void didUpdateWidget(T oldWidget)
```

State lifecycle - state.widget, didUpdateWidget

- Demo - ex_06_hello_world_did_update_widget

```dart
class _HelloWorldTitleOnlyWidgetState extends State<HelloWorldTitleOnlyWidget> {
  @override
  Widget build(BuildContext context) {
    return Text(
      widget.title,
      textDirection: TextDirection.ltr,
      style: TextStyle(color: Colors.black, fontSize: 40),
    ); // Text
  }

  @override
  void didUpdateWidget(HelloWorldTitleOnlyWidget oldWidget) {
    super.didUpdateWidget(oldWidget);
```

Stateless & Stateful widgets

A widget is either stateful or stateless.

If a widget can change -
when a user interacts with it, for example—it's stateful.

Who manages the stateful widget's state?

https://flutter.dev/docs/development/ui/interactive#managing-state

Who manages the stateful widget's state?
The widget itself? The parent widget? Both? Another object? The answer is... it depends.

There are several valid ways to make your widget interactive. Here are the most common ways to manage state:

- The widget manages its own state
- The parent manages the widget's state
- A mix-and-match approach

How do you decide which approach to use? The following principles should help you decide:
If the state in question is user data, for example the checked or unchecked mode of a checkbox, or the position of a slider, then the state is best managed by the parent widget.

If the state in question is aesthetic, for example an animation, then the state is best managed by the widget itself.

If in doubt, start by managing state in the parent widget.

О чем будем говорить

Что такое виджет Flutter?

Stateless & Stateful widgets

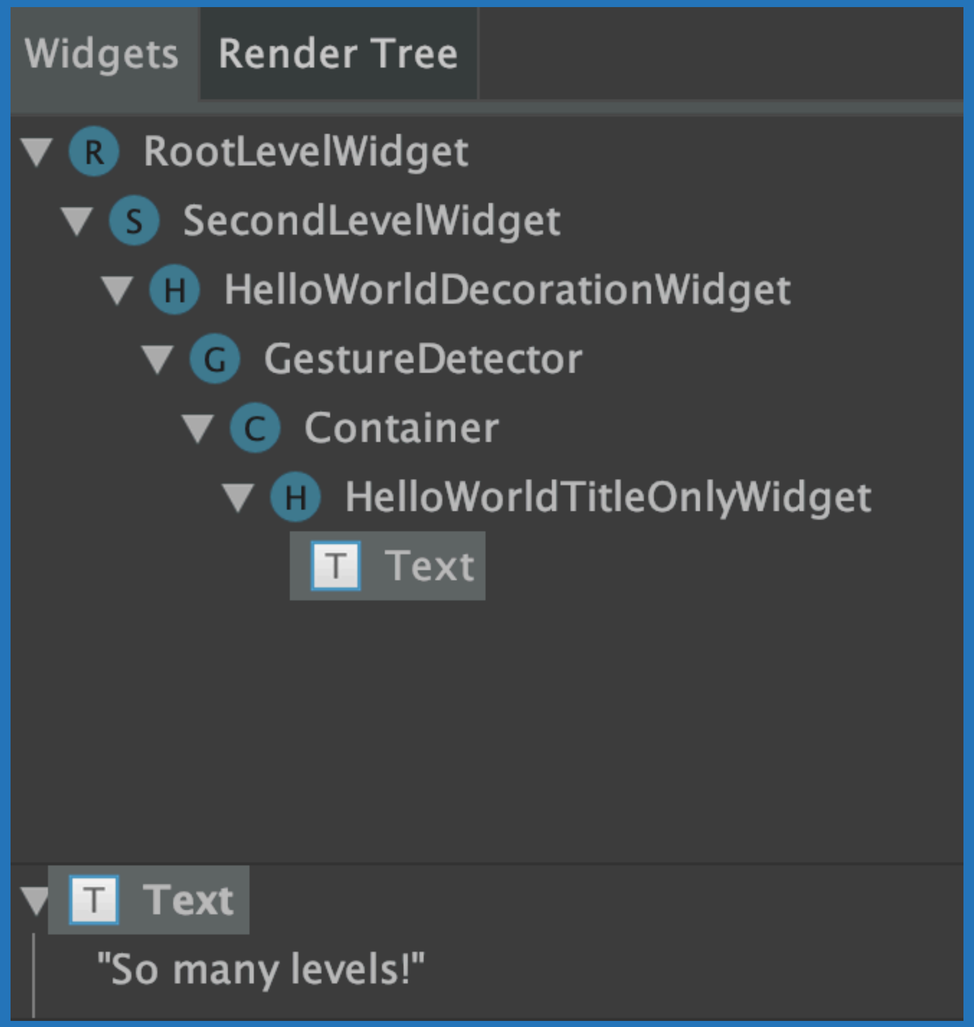
Inherited Widgets

Keys & Global Keys

Flutter Engine

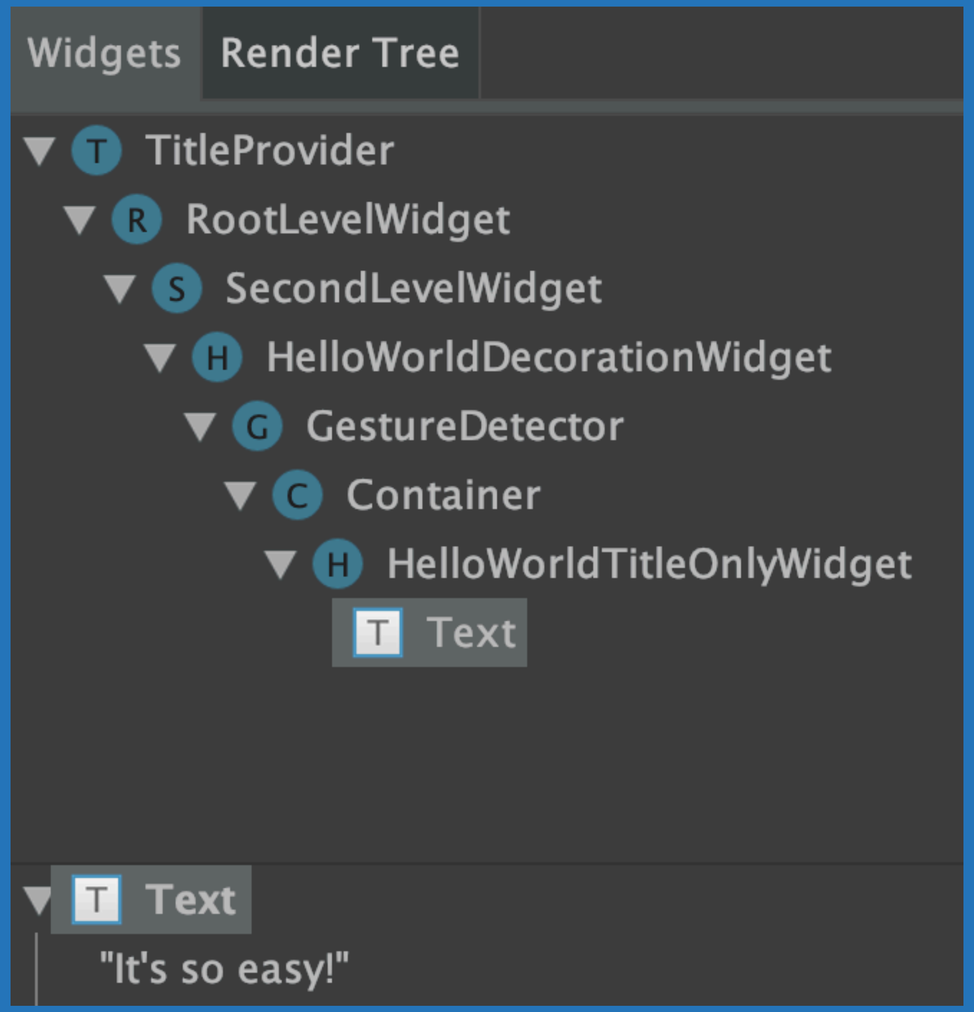
Up to down configuration propagation

- Demo - ex_07_hello_world_so_many_levels_problem



Inherited Widget

- Demo - ex_08_hello_world_inherited_widget



Inherited Widget

```dart
void main() {
  const String rootLevelTitle = 'It\'s so easy!';
  runApp(
    TitleProvider(
      title: rootLevelTitle,
      child: RootLevelWidget(),
    ), // TitleProvider
  );
}
```

Inherited Widget common pattern to use

- Demo - ex_09_hello_world_title_provider_of

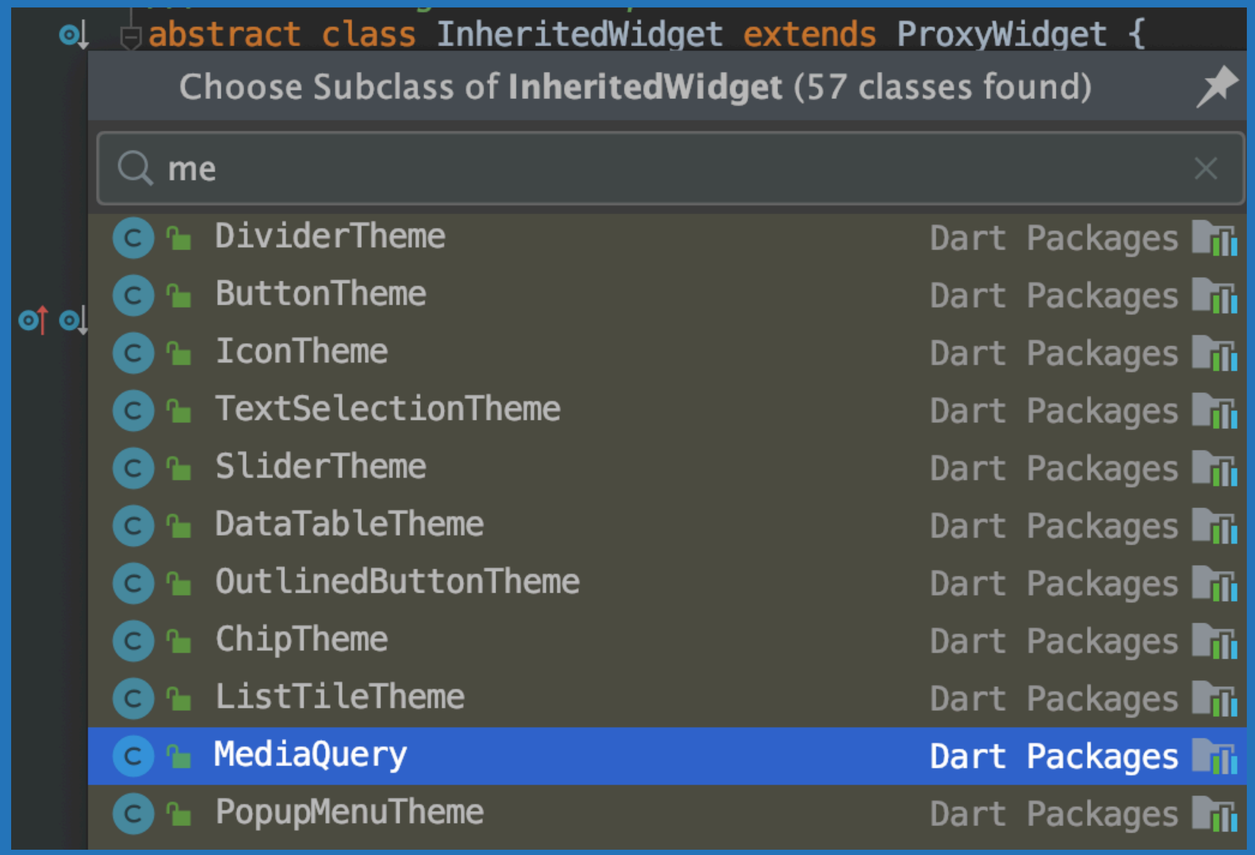
```dart
class TitleProvider extends InheritedWidget {
  final String title;
  final Widget child;

  TitleProvider({@required this.title, @required this.child}) : super(child: child);

  @override
  bool updateShouldNotify(InheritedWidget oldWidget) => true;

  static TitleProvider of(BuildContext context) =>
      context.dependOnInheritedWidgetOfExactType<TitleProvider>();
}
```

```dart
class HelloWorldTitleOnlyWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final title = TitleProvider.of(context).title;
    final size = MediaQuery.of(context).size;
```

Inherited Widget

/// Base class for widgets that efficiently propagate information down the tree.

abstract class InheritedWidget extends ProxyWidget {



Inherited Notifier - events propagation

https://api.flutter.dev/flutter/widgets/InheritedNotifier-class.html

/// An inherited widget for a [Listenable] [notifier], which updates its
/// dependencies when the [notifier] is triggered.
///
/// This is a variant of [InheritedWidget], specialized for subclasses of
/// [Listenable], such as [ChangeNotifier] or [ValueNotifier].

abstract class InheritedNotifier<T extends Listenable> extends InheritedWidget

О чем будем говорить

Что такое виджет Flutter?

Stateless & Stateful widgets

Inherited Widgets

Keys & Global Keys

Flutter Engine

Keys & Global Keys

https://flutter.dev/docs/development/ui/widgets-intro#keys

Keys

Use keys to control which widgets the framework matches up with other widgets when a widget rebuilds.
By default, the framework matches widgets in the current and previous build according to their runtimeType and the order in which they appear.
With keys, the framework requires that the two widgets have the same key as well as the same runtimeType.

https://flutter.dev/docs/development/ui/widgets-intro#global-keys

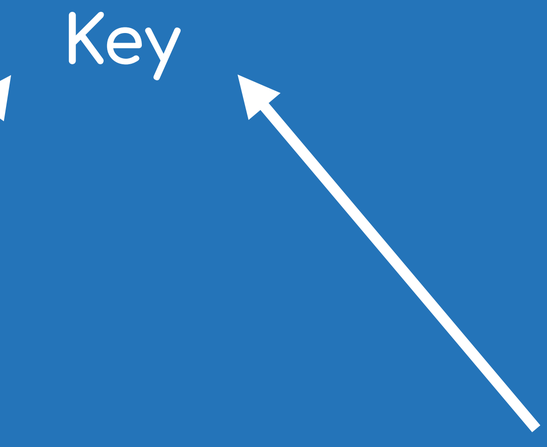
Global keys

Use global keys to uniquely identify child widgets. Global keys must be globally unique across the entire widget hierarchy, unlike local keys which need only be unique among siblings. Because they are globally unique, a global key can be used to retrieve the state associated with a widget.

Widgets & Keys

```
abstract class Widget extends DiagnosticableTree {
  /// Initializes [key] for subclasses.
  const Widget({ this.key });

/// Controls how one widget replaces another widget in the tree.
///
/// Using a [GlobalKey] as the widget's [key] allows the element
/// to be moved around the tree (changing parent) without losing state. When a
/// new widget is found (its key and type do not match a previous widget in
/// the same location), but there was a widget with that same global key
/// elsewhere in the tree in the previous frame, then that widget's element is
/// moved to the new location.
final Key key;
```

Keys



Local Key

Object Key

Value Key

Unique Key

Keys must be unique
amongst the [Element]s
with the same parent

Global Key

A key that is unique
across the entire app.

Global keys

```
/// A key that is unique across the entire app.
///
/// Global keys uniquely identify elements. Global keys provide access to other
/// objects that are associated with those elements, such as [BuildContext].
/// For [StatefulWidget]s, global keys also provide access to [State].
///
abstract class GlobalKey<T extends State<StatefulWidget>> extends Key
```

Global Key

- Demo - ex_10_hello_world_title_global_keys

```dart
final globalKey = GlobalKey<_HelloWorldTitleOnlyWidgetState>(debugLabel: 'title key');

void main() {
  runApp(
    GestureDetector(
      onTap: () {
        globalKey.currentState.updateNewState();
      },
      child: RootLevelWidget(),
    ), // GestureDetector
  );
}
```

О чем будем говорить

Что такое виджет Flutter?

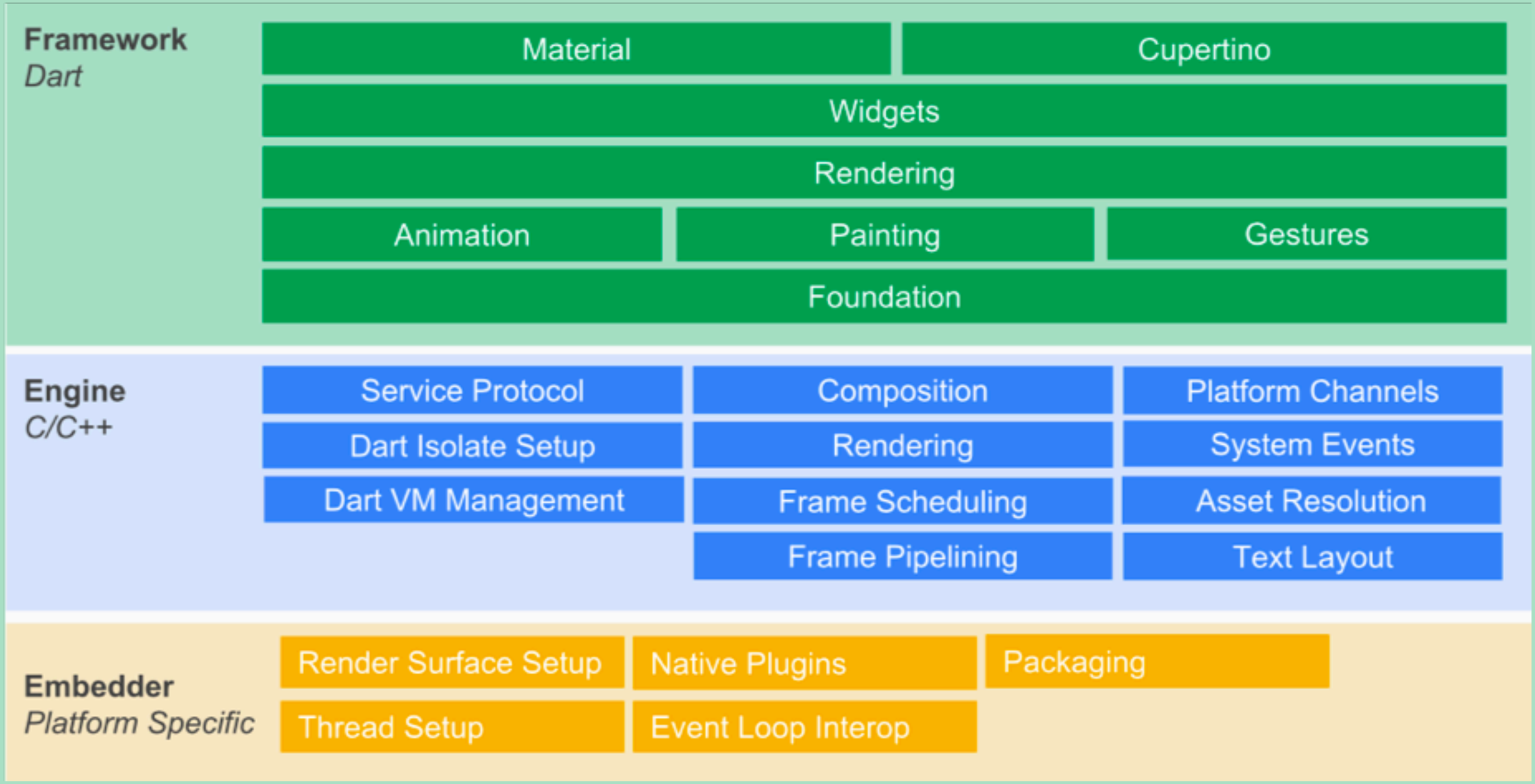
Stateless & Stateful widgets

Inherited Widgets

Keys & Global Keys

Flutter Engine

Flutter Engine



Flutter Embedder

```
/// The platform that user interaction should adapt to target.

enum TargetPlatform {
  /// Android: <https://www.android.com/>
  android,

  /// Fuchsia: <https://fuchsia.dev/fuchsia-src/concepts>
  fuchsia,

  /// iOS: <https://www.apple.com/ios/>
  iOS,

  /// Linux: <https://www.linux.org>
  linux,

  /// macOS: <https://www.apple.com/macos>
  macOS,

  /// Windows: <https://www.windows.com>
  windows,
}
```

Flutter Engine

https://github.com/flutter/flutter/wiki/Custom-Flutter-Engine-Embedders

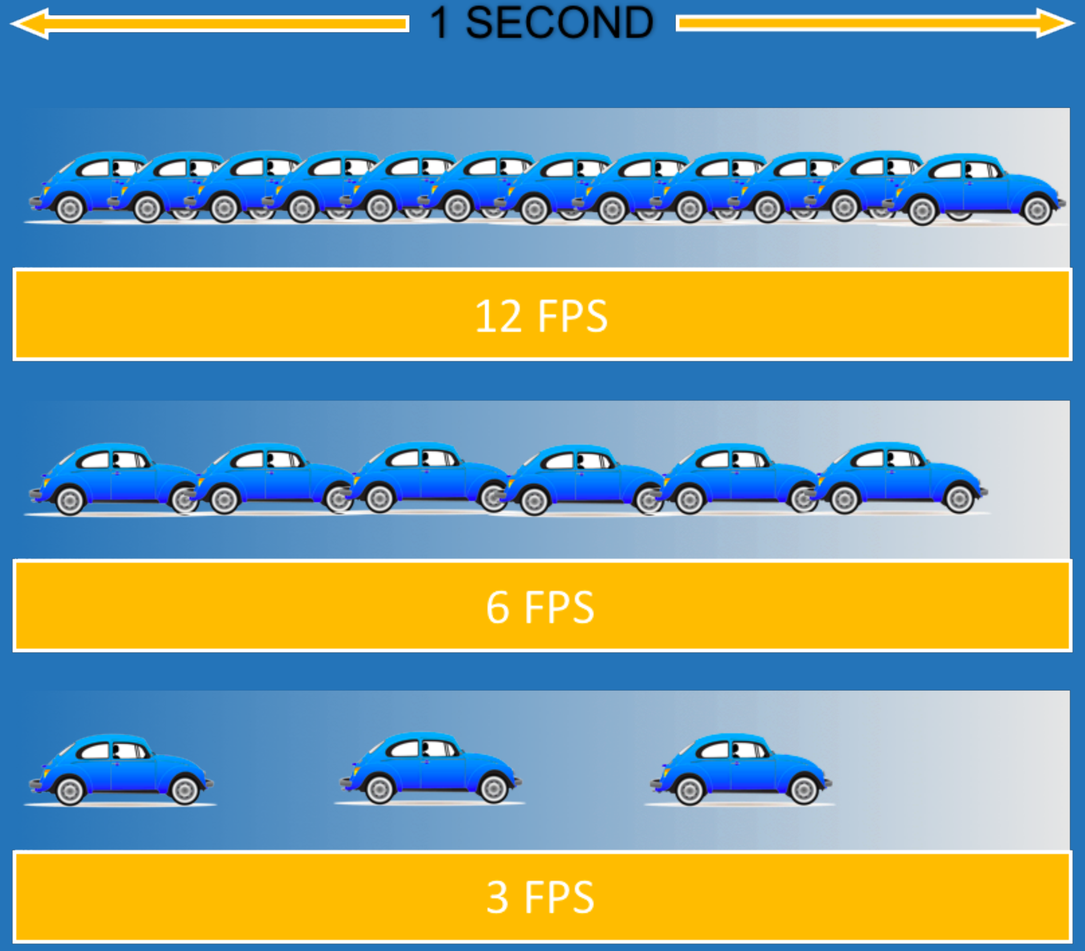
The Flutter Engine is window toolkit agnostic - Flutter engine не зависит от платформы,
он работает через Embedder.

Сам по себе Flutter Engine не знает, как работать с desktop - эту проблему решает embeder.

Но главные задачи Flutter Engine - это рендеринг и рантайм для вашего кода на языке Dart:

- **Skia** - a 2D graphics rendering library
- **Dart** - a VM for a garbage-collected object-oriented language

Flutter Engine - fps



The Skia Graphics Engine is a compact open source graphics library written in C++.

SKIA

Skia Graphics Library

Skia is an open source 2D graphics library which provides common APIs that work across a variety of hardware and software platforms. It serves as the graphics engine for Google Chrome and Chrome OS, Android, Mozilla Firefox and Firefox OS, and many other products.

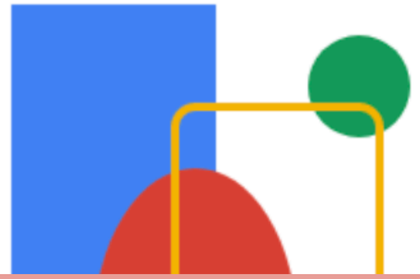
Skia is sponsored and managed by Google, but is available for use by anyone under the BSD Free Software License. While engineering of the core components is done by the Skia development team, we consider contributions from any source.

- Canonical source tree: skia.googlesource.com/skia.
- Issue tracker: bug.skia.org.
- Discussion forum: skia-discuss@googlegroups.com.
- API Reference and Overview: skia.org/user/api.
- Skia Fiddle: fiddle.skia.org.

Showcase

Click on any image below to see the source code that generated the image.

Shapes



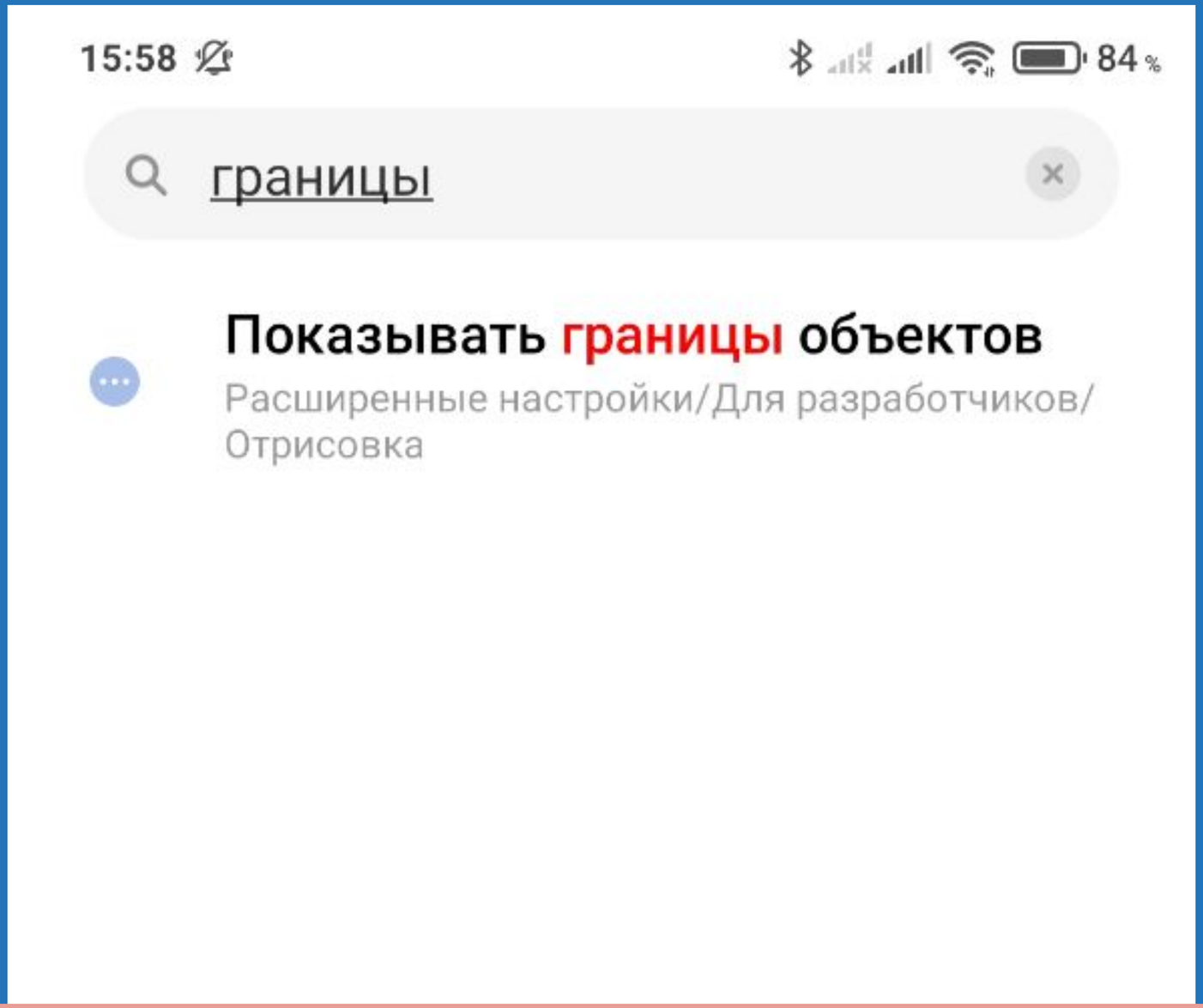
Bézier Curves

Translations and Rotations

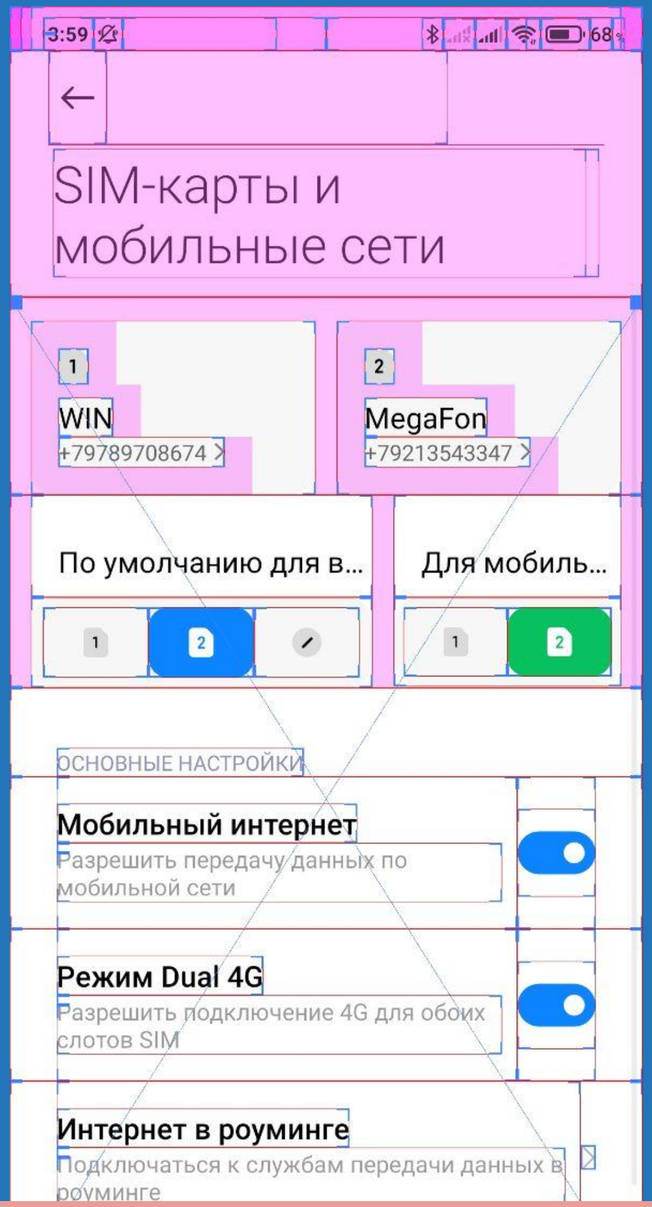
Text Rendering

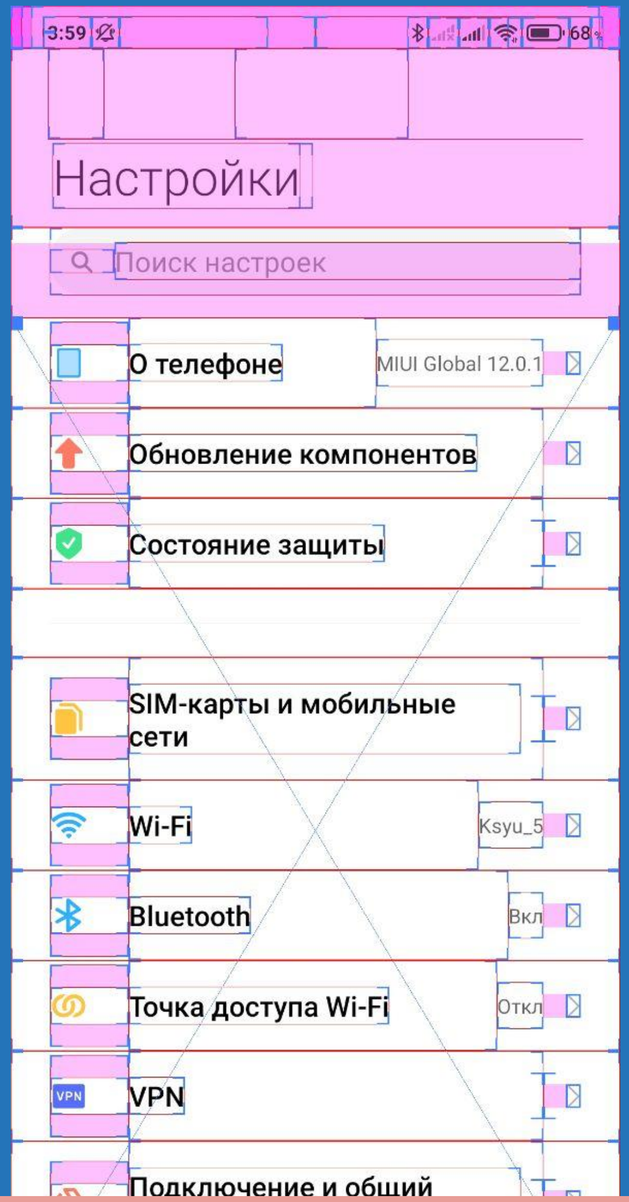
https://fiddle.skia.org/named/

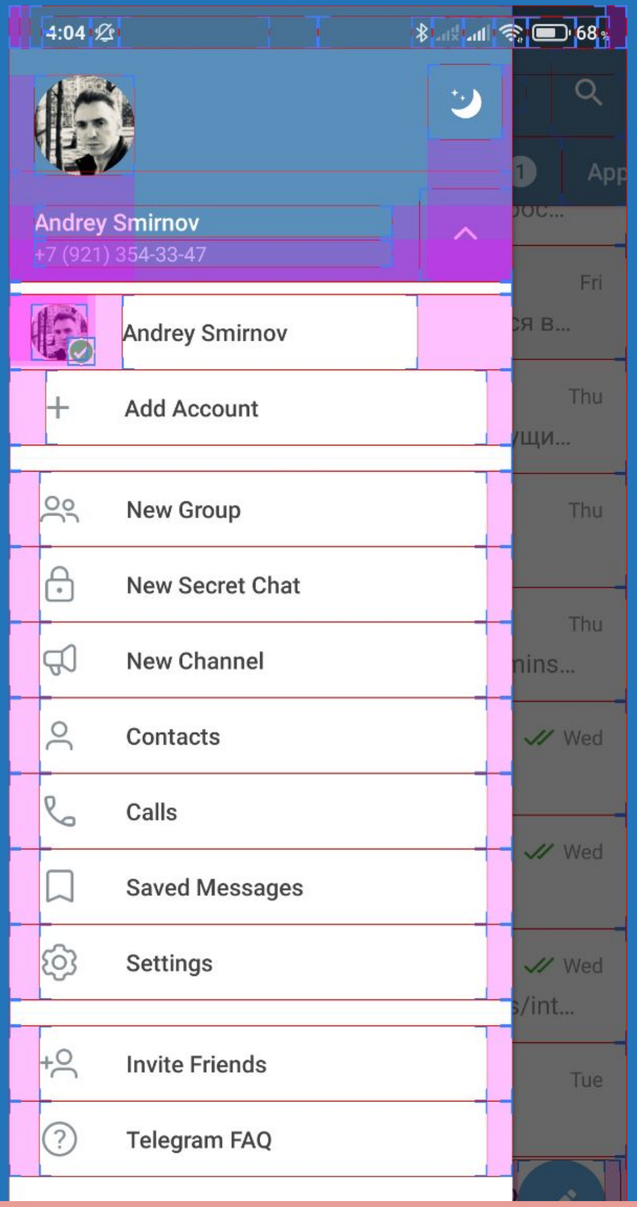
Flutter Engine



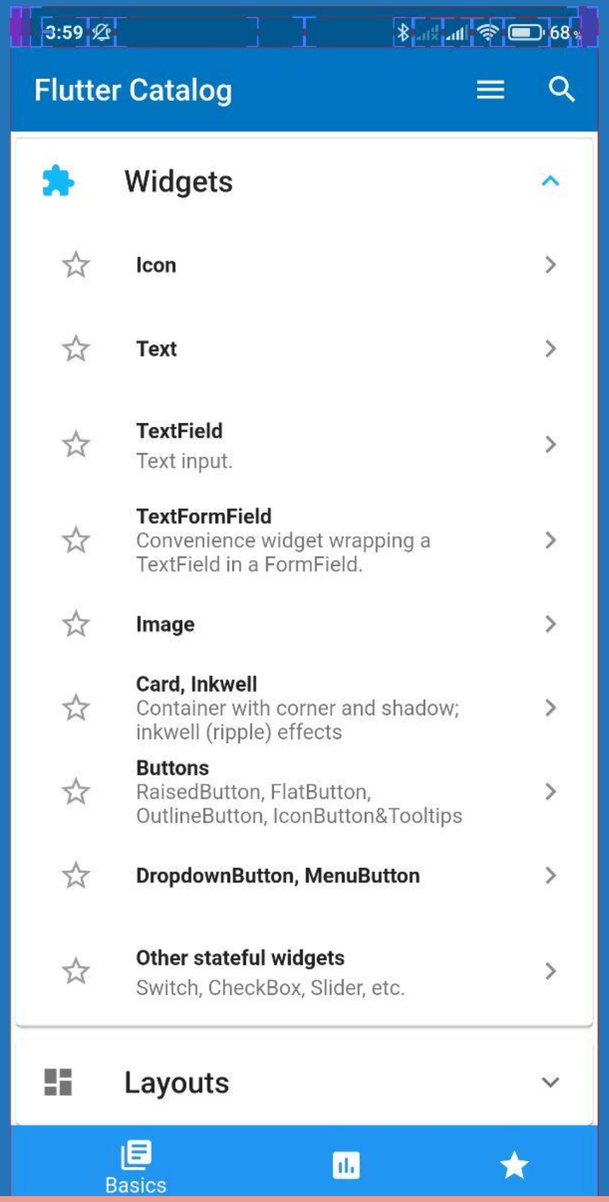
Flutter Engine

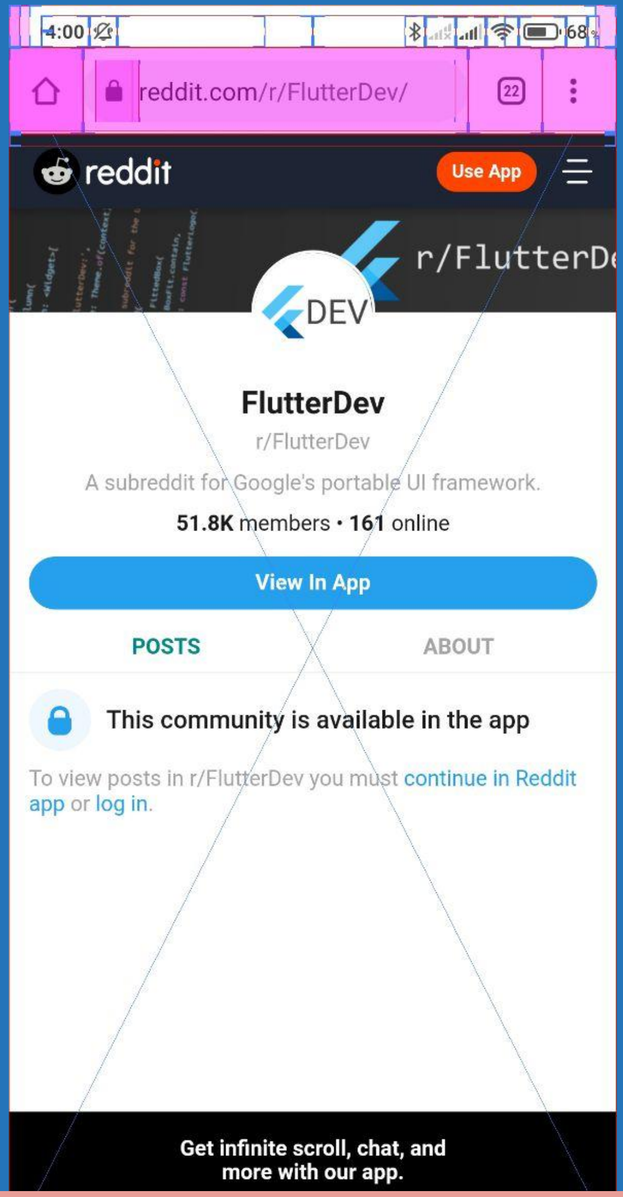






Flutter Engine



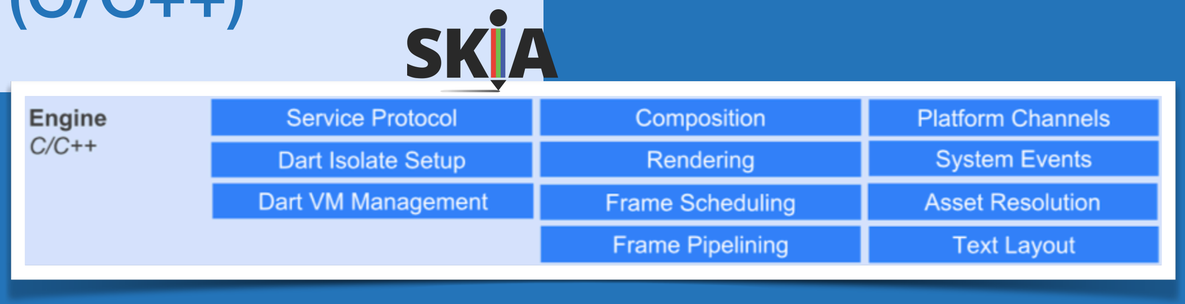


Flutter Engine - Window class & ui.window

Flutter Framework (Dart & Wingets)

ui.Window

Flutter Engine (C/C++)



Window class

https://api.flutter.dev/flutter/dart-ui/Window-class.html

Window class Null safety

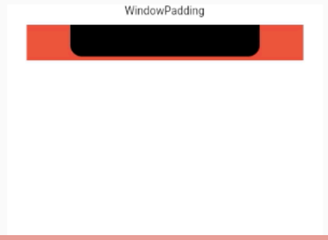
The most basic interface to the host operating system's user interface.

It exposes the size of the display, the core scheduler API, the input event callback, the graphics drawing API, and other such core services.

There is a single Window instance in the system, which you can obtain from `WidgetsBinding.instance.window`.

There is also a window singleton object in `dart:ui` if `WidgetsBinding` is unavailable. But we strongly advise to avoid statically referencing it. See the document of window for more details of why it should be avoided.

Insets and Padding



Window class

https://api.flutter.dev/flutter/dart-ui/Window-class.html

Flutter › dart:ui › Window class

CLASSES

AccessibilityFeatures
BackdropFilterEngineLayer
CallbackHandle
Canvas
ChannelBuffers
ClipPathEngineLayer
ClipRectEngineLayer
ClipRRectEngineLayer
Codec
Color
ColorFilter
ColorFilterEngineLayer
EngineLayer

inherited

render(Scene scene) → void
Updates the application's rendering on the GPU with the newly provided Scene. This function must be called within the scope of the onBeginFrame or onDrawFrame callbacks being invoked. If this function is called a second time during a single onBeginFrame/onDrawFrame callback sequence or called outside the scope of those callbacks, the call will be ignored. [...]

scheduleFrame() → void
Requests that, at the next appropriate opportunity, the onBeginFrame and onDrawFrame callbacks be invoked. [...]

sendPlatformMessage(String name, ByteData? data, PlatformMessageResponseCallback? callback) → void
Sends a message to a platform-specific plugin. [...]

setIsolateDebugName(String name) → void
Set the debug name associated with this window's root isolate. [...]

toString() → String
Returns a string representation of this object.
inherited

updateSemantics(SemanticsUpdate update) → void

Window class

https://api.flutter.dev/flutter/dart-ui/Window-class.html

Flutter > dart:ui > Window > onBeginFrame property

PROPERTIES

accessibilityFeatures
alwaysUse24HourFormat
defaultRouteName
devicePixelRatio
hashCode
initialLifecycleState
locale
locales
onAccessibilityFeaturesC...
onBeginFrame
onDrawFrame
onLocaleChanged
onMetricsChanged

onBeginFrame property

Null safety

FrameCallback? onBeginFrame

A callback that is invoked to notify the application that it is an appropriate time to provide a scene using the SceneBuilder API and the render method. When possible, this is driven by the hardware VSync signal. This is only called if scheduleFrame has been called since the last time this callback was invoked.

The onDrawFrame callback is invoked immediately after onBeginFrame, after draining any microtasks (e.g. completions of any Futures) queued by the onBeginFrame handler.

The framework invokes this callback in the same zone in which the callback was set.

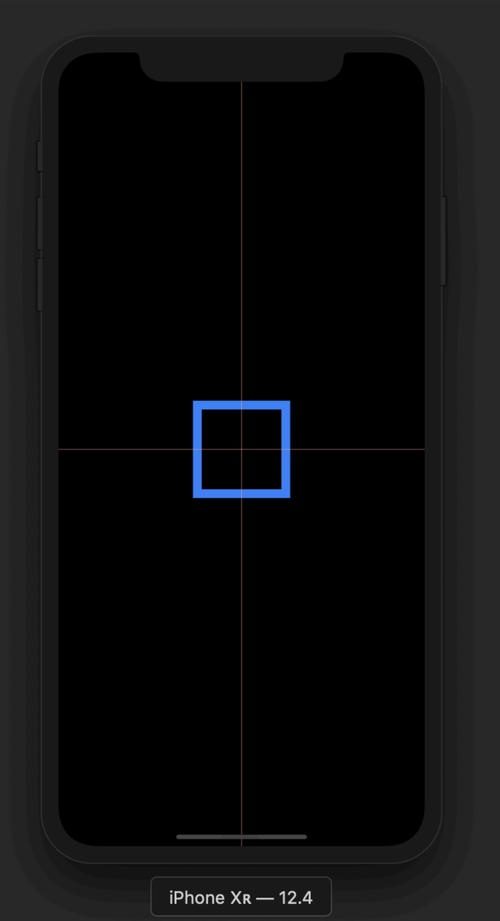
See also:

- SchedulerBinding, the Flutter framework class which manages the scheduling of frames.
- RendererBinding, the Flutter framework class which manages layout and painting.

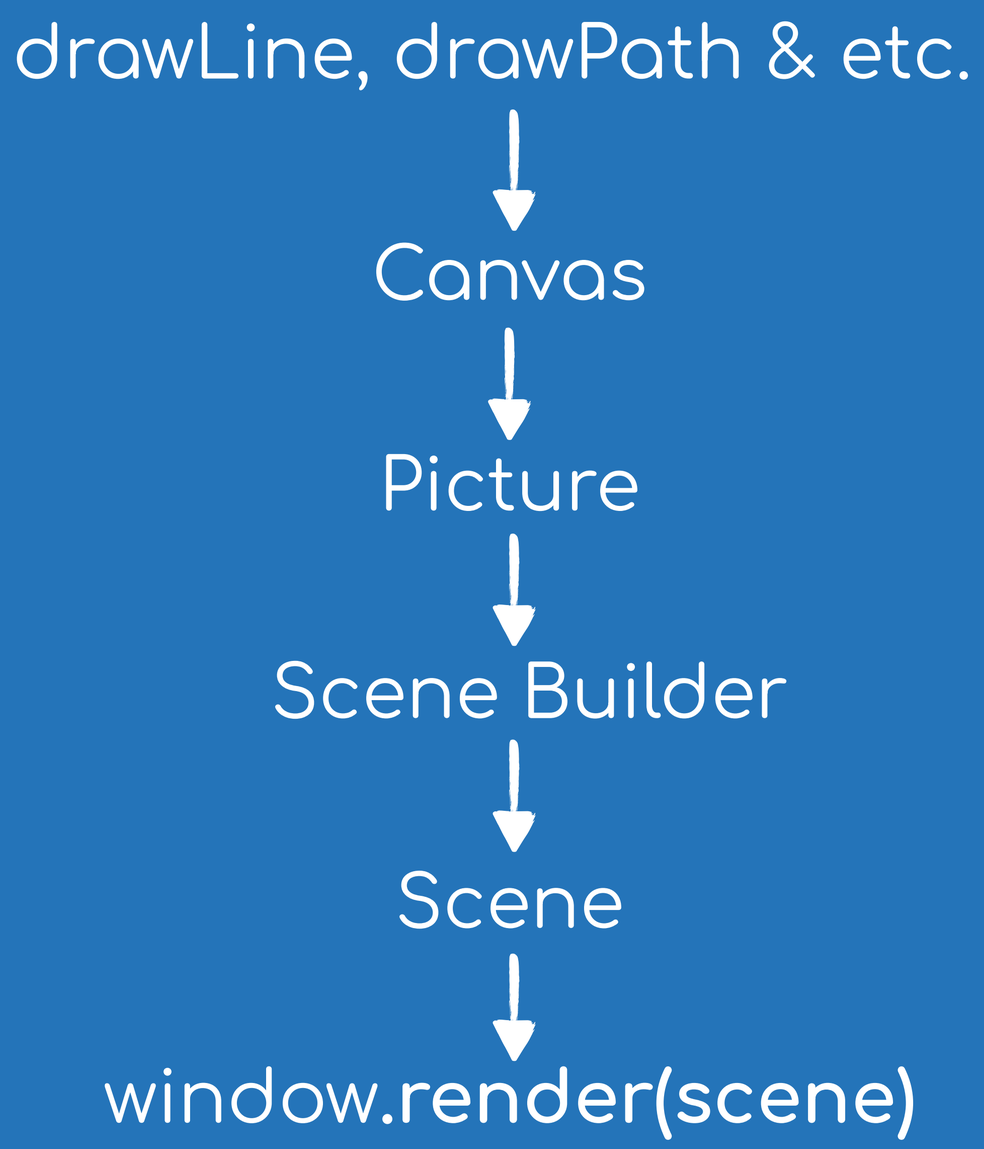
main() -> without runApp() ?

main() -> without runApp() ?

```
final PictureRecorder recorder = PictureRecorder();
final Canvas canvas = Canvas(recorder);

final paint = Paint()
  ..style = PaintingStyle.stroke
  ..color = const Color(0xff4285F4);

Rect rect = Rect.fromLTRB(..);

canvas
  ..translate(centerX, centerY)
  ..drawRect(rect, paint);

final Picture picture = recorder.endRecording();

final SceneBuilder sceneBuilder = SceneBuilder()
  ..pushOffset(0, 0)
  ..addPicture(Offset.zero, picture)
  ..pop();

window.render(sceneBuilder.build());
```



Rendering on Window



Window class

SceneBuilder().build()

Scene Builder

```
final ui.SceneBuilder sceneBuilder = ui.SceneBuilder()
        ..
```

window.onBeginFrame =
callback(timeStamp)

window.render(scene)

window.scheduleFrame()

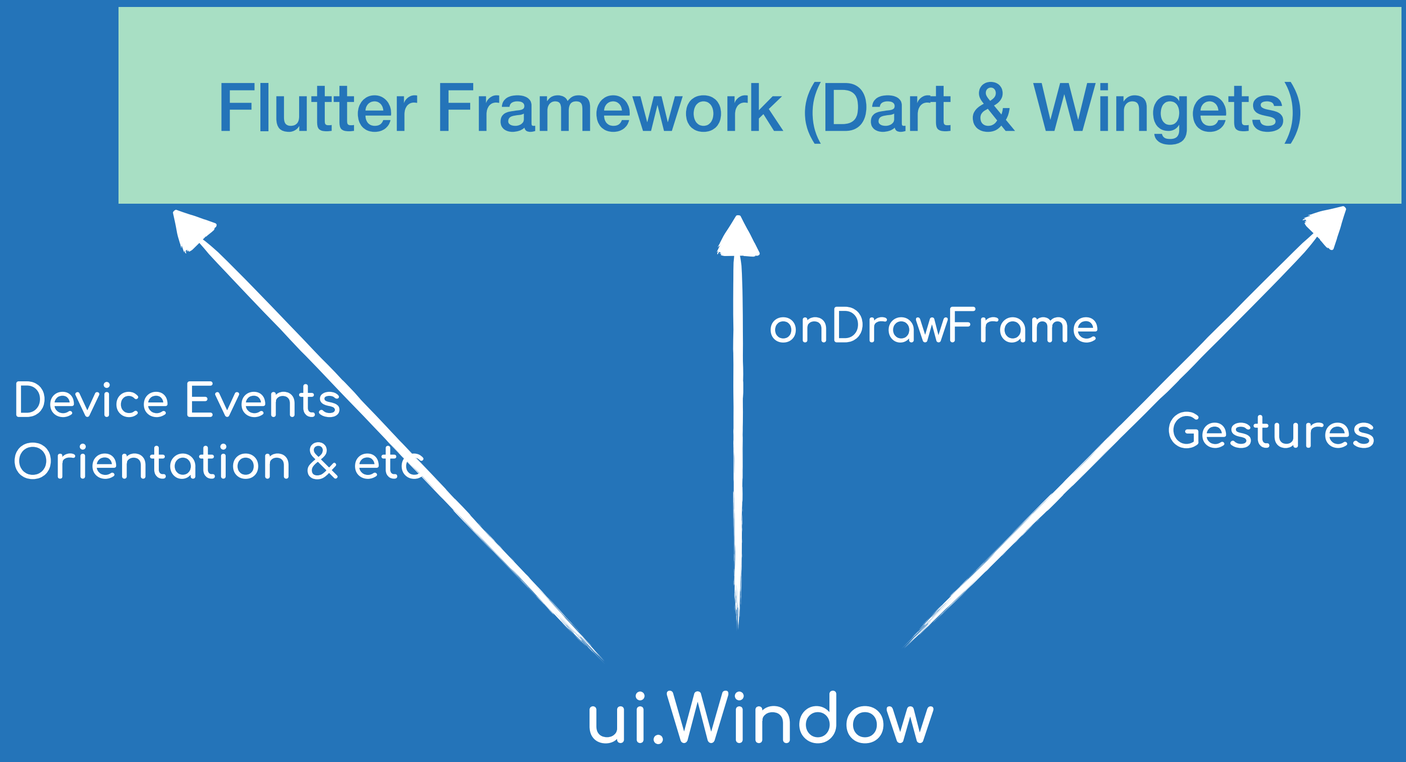
ns

Window class

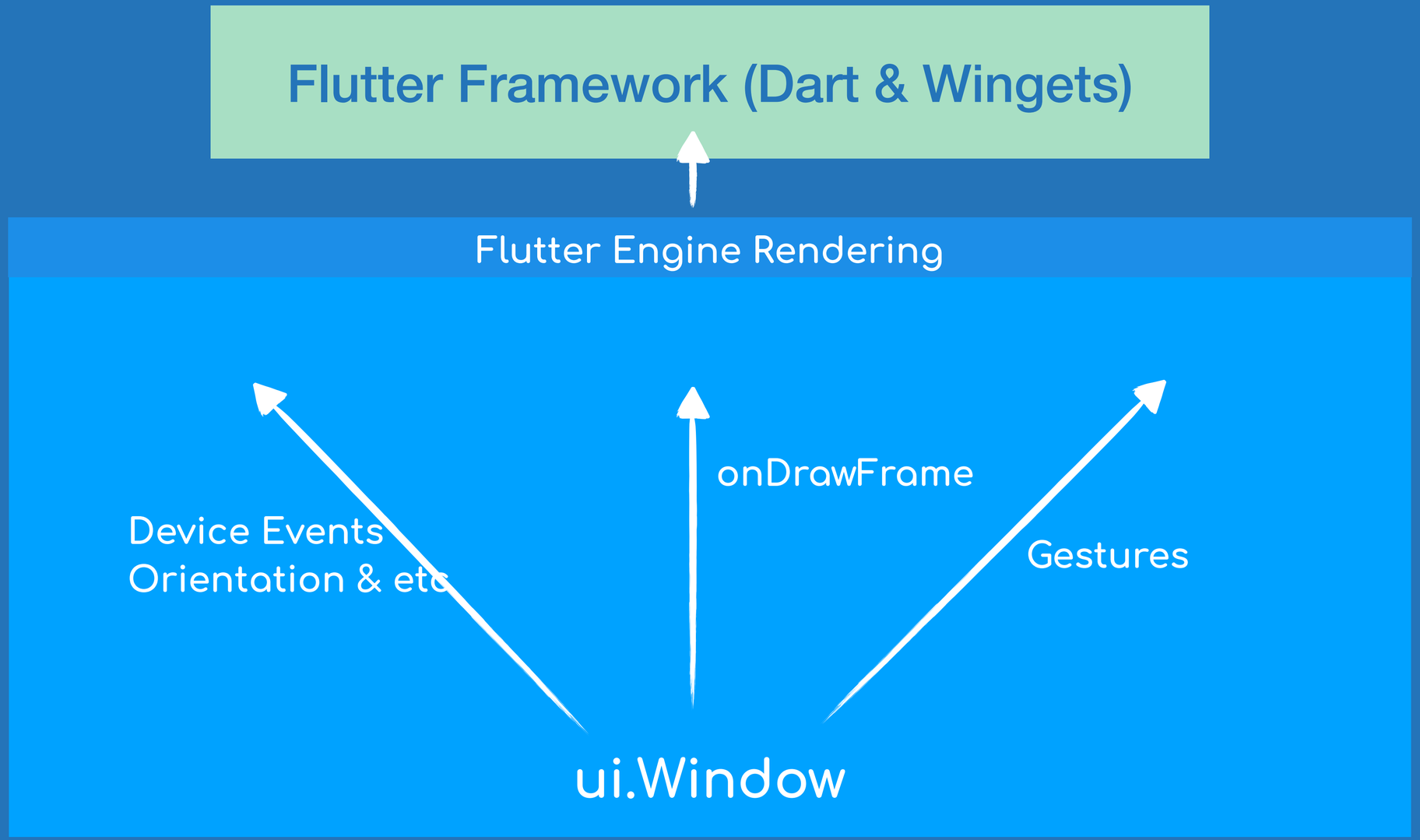
- Demo - ex_12_scene_builder

```dart
void main() {
  window.onBeginFrame = beginFrame;
  window.scheduleFrame();
}

///
/// See more details on https://fiddle.skia.org/c/@shapes
///
void beginFrame(Duration timeStamp) {
  final double devicePixelRatio = window.devicePixelRatio;
  final Size logicalSize = window.physicalSize / devicePixelRatio;

  final Rect physicalBounds = Offset.zero & (logicalSize * devicePixelRatio);
  final PictureRecorder recorder = PictureRecorder();
  final Canvas canvas = Canvas(recorder, physicalBounds)..scale(devicePixelRatio, devicePixelRatio);
```

Widgets & Window



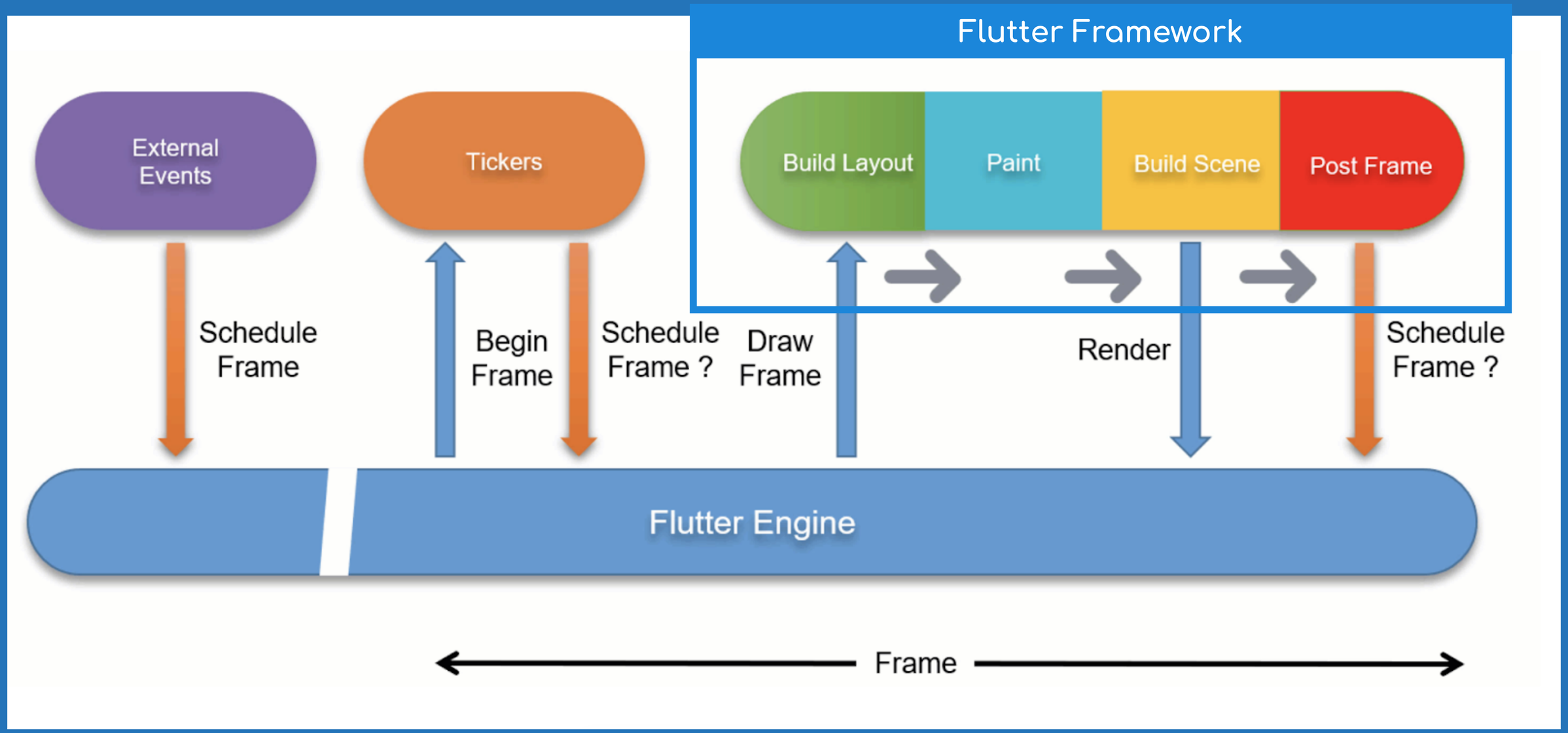
Flutter Engine Rendering



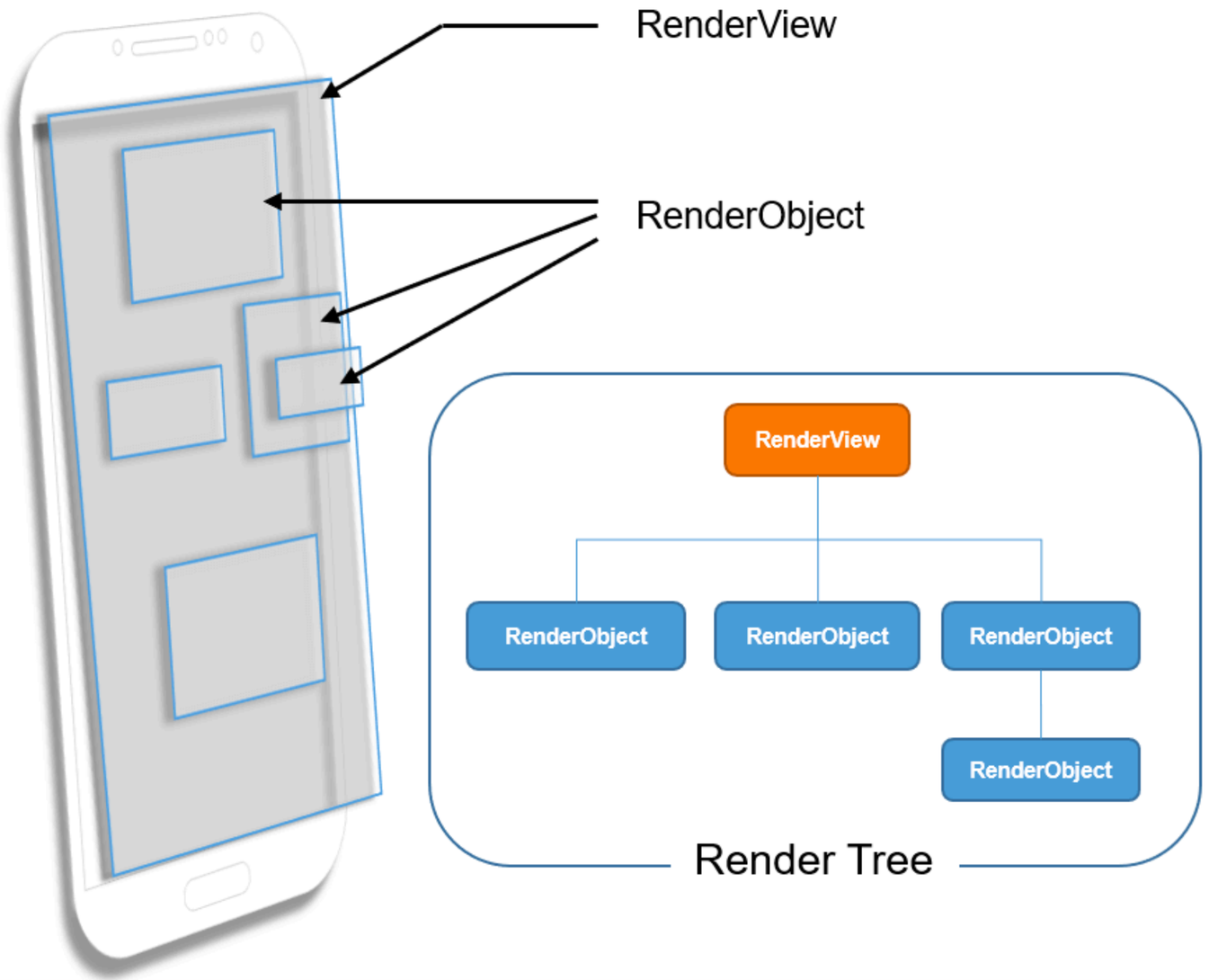
debugPrintScheduleFrameStacks

```
/// Log the call stacks that cause a frame to be scheduled.
///
/// This is called whenever [SchedulerBinding.scheduleFrame] schedules a frame. This
/// can happen for various reasons, e.g. when a [Ticker] or
/// [AnimationController] is started, or when [RenderObject.markNeedsLayout] is
/// called, or when [State.setState] is called.
///
/// To get a stack specifically when widgets are scheduled to be built, see
/// [debugPrintScheduleBuildForStacks].
bool debugPrintScheduleFrameStacks = false;
```

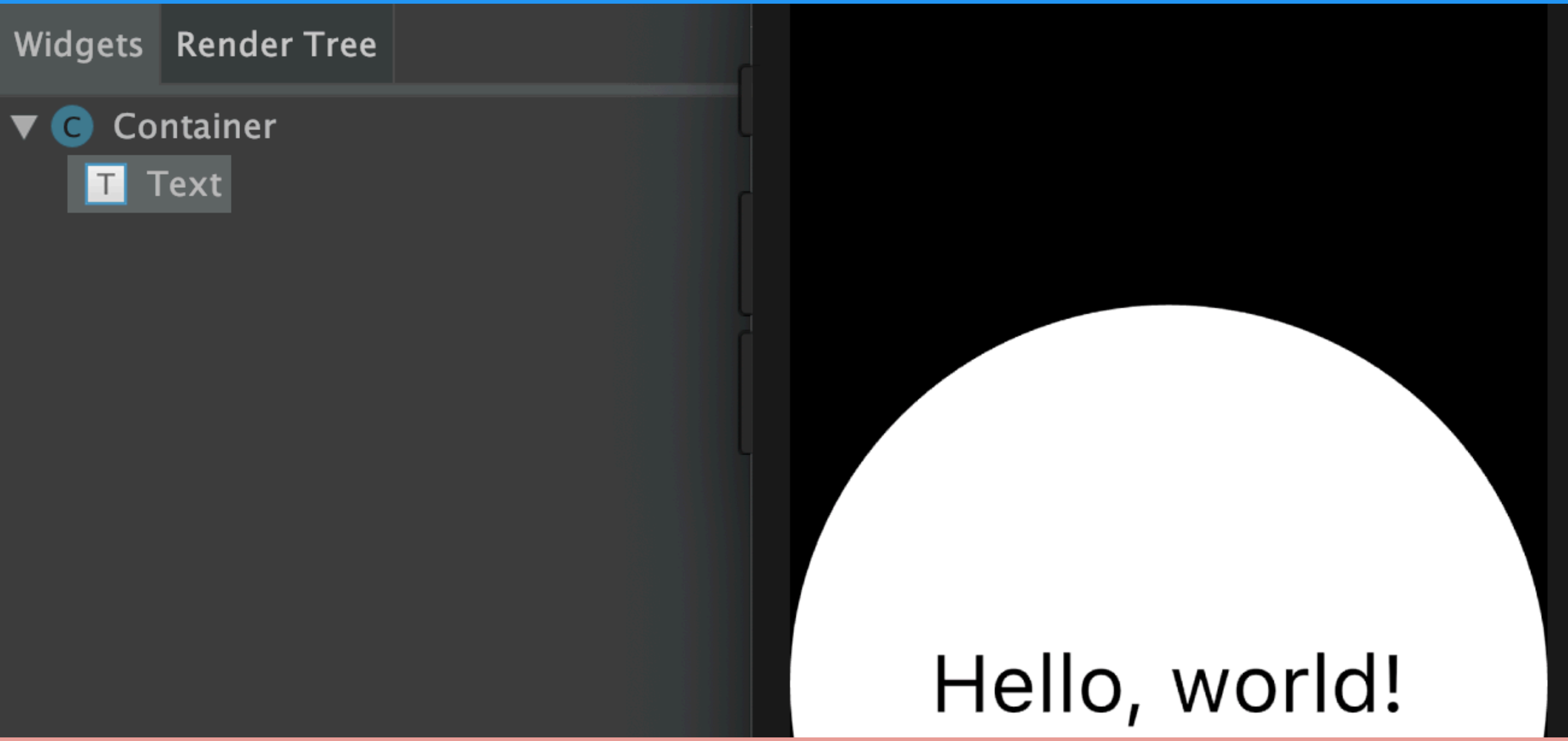
Flutter Engine Rendering



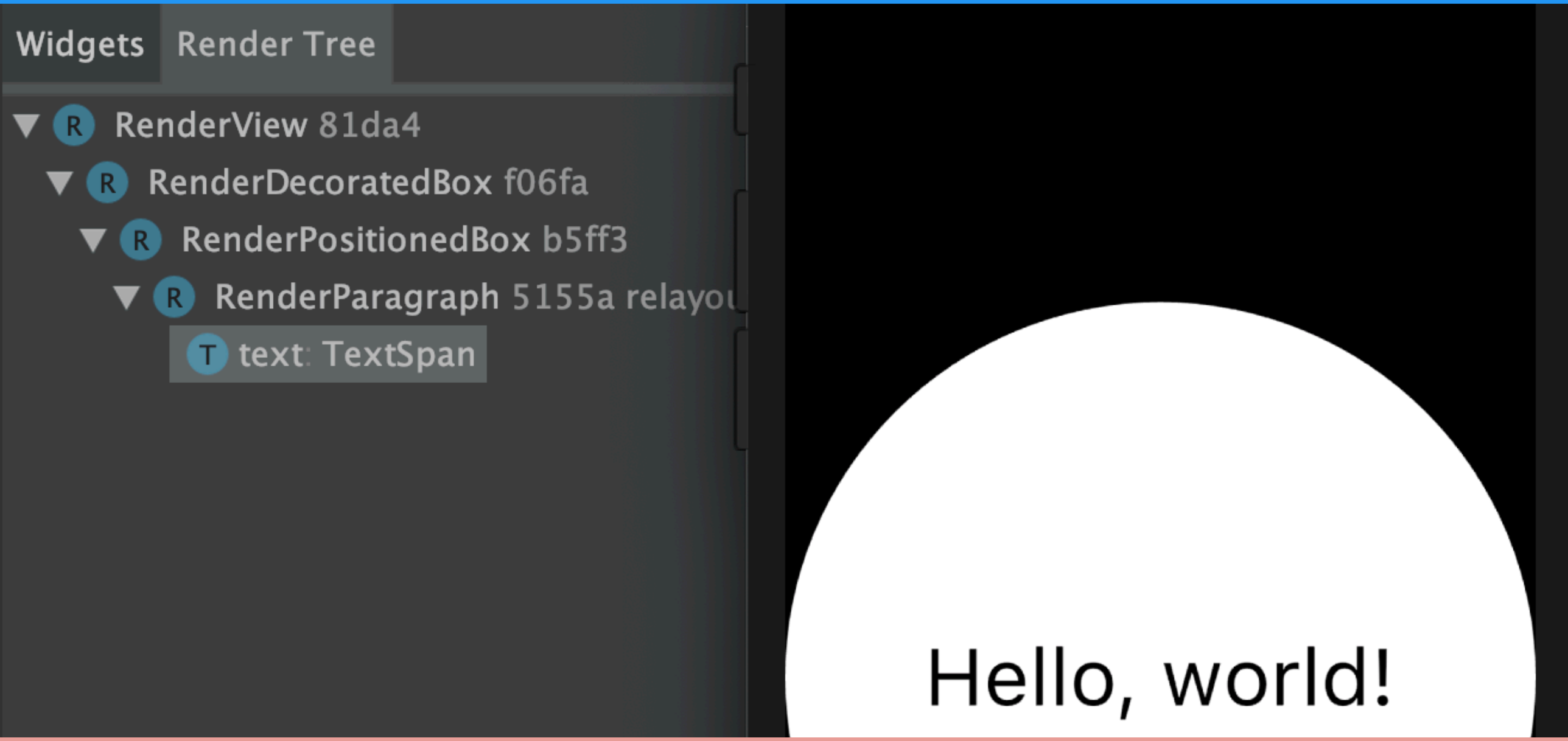
Render Tree


RenderView
RenderObject
RenderView
RenderObject
RenderObject
RenderObject
RenderObject
Render Tree

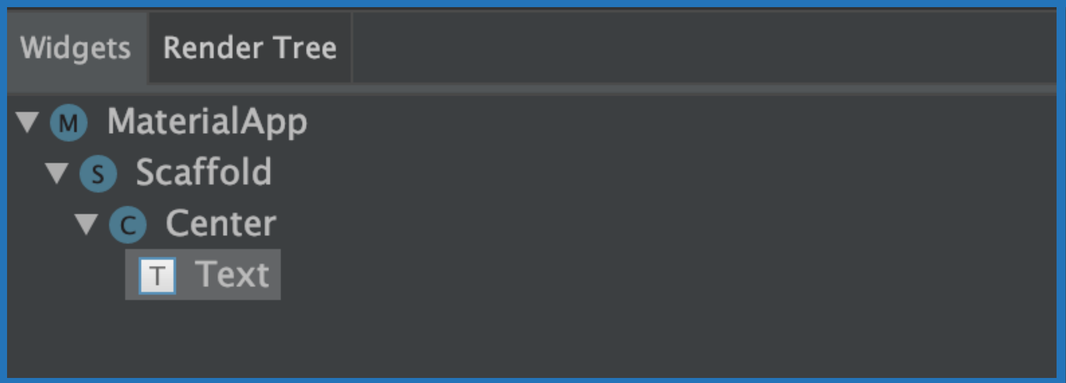
Widgets Tree



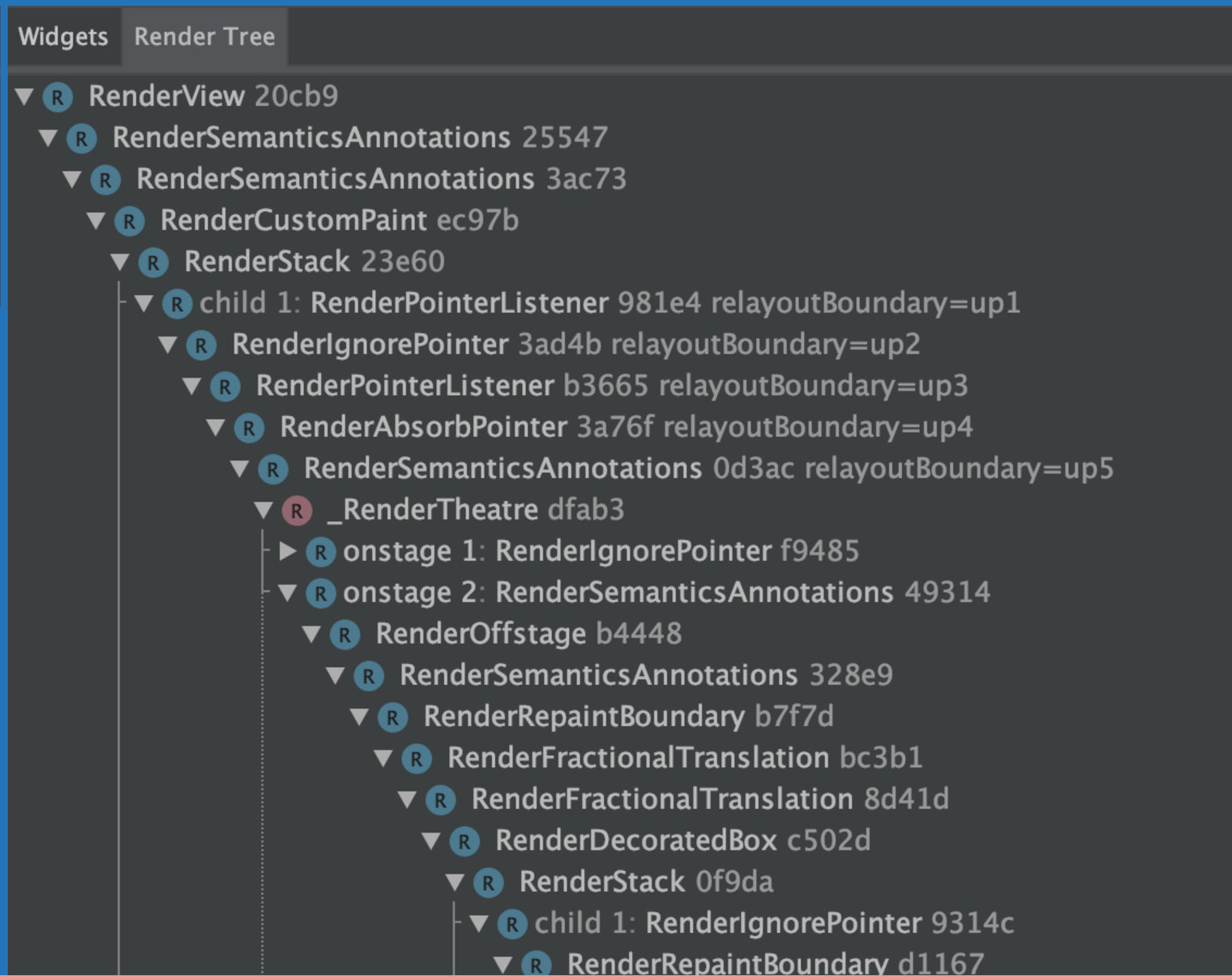
Render Tree


Widgets
Render Tree
RenderView 81da4
RenderDecoratedBox f06fa
RenderPositionedBox b5ff3
RenderParagraph 5155a relayou
text: TextSpan
Hello, world!

Render Tree & widgets tree are not equal




```
void main() {
  runApp(
    MaterialApp(
      home: Scaffold(
        body: Center(
          child: Text(
            'Hello, world!!',
            textDirection: TextDirection.ltr,
          ),
        ),
      ),
    ),
  );
}
```



Flutter Engine

RunApp()

• Demo

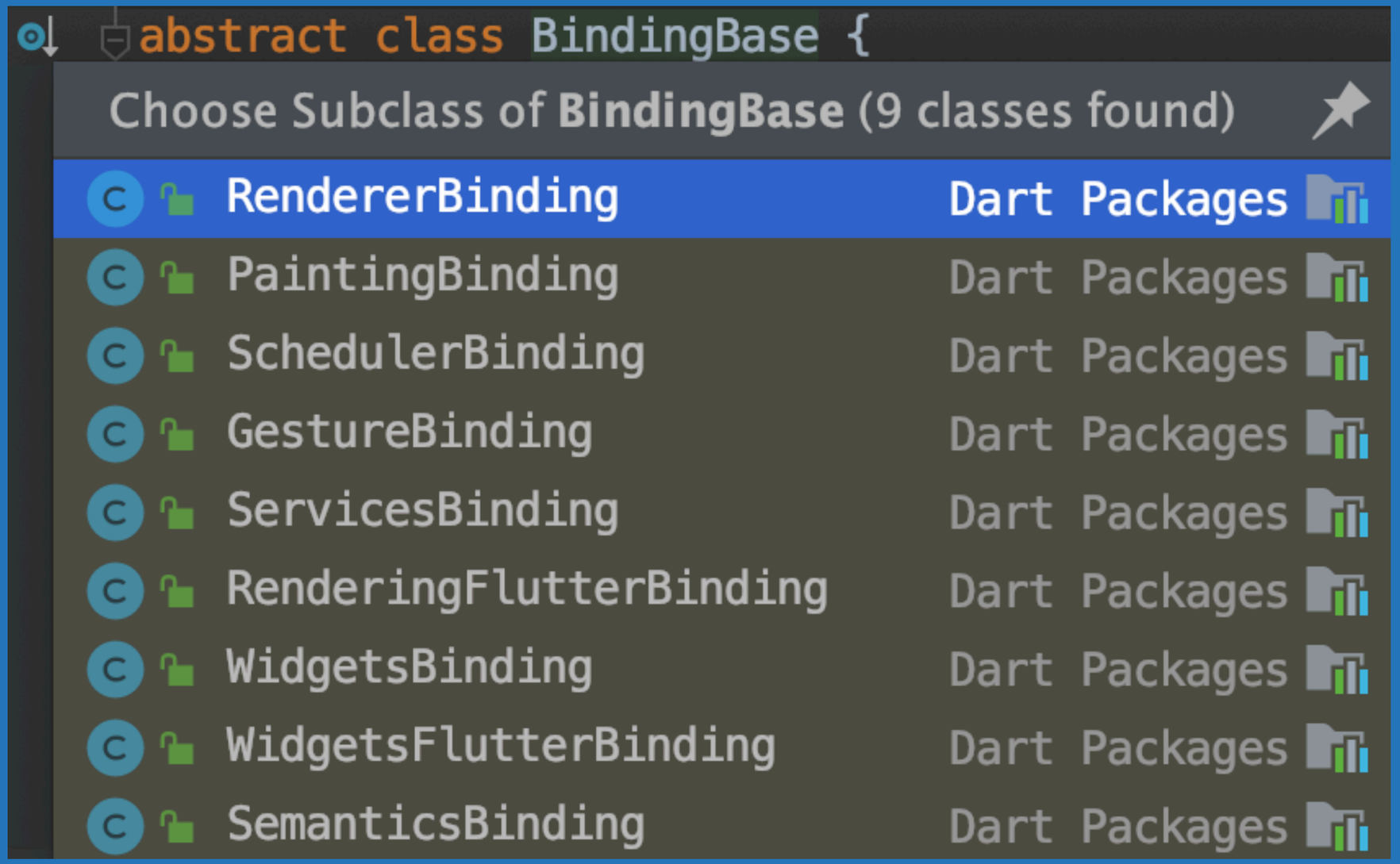
```
/// Initializes the binding using [WidgetsFlutterBinding] if necessary.
///
/// See also:
///
///  * [WidgetsBinding.attachRootWidget], which creates the root widget for the
///    widget hierarchy.
///  * [RenderObjectToWidgetAdapter.attachToRenderTree], which creates the root
///    element for the element hierarchy.
///  * [WidgetsBinding.handleBeginFrame], which pumps the widget pipeline to
///    ensure the widget, element, and render trees are all built.
void runApp(Widget app) {
  WidgetsFlutterBinding.ensureInitialized()
    ..scheduleAttachRootWidget(app)
    ..scheduleWarmUpFrame();
}
```

RunApp()

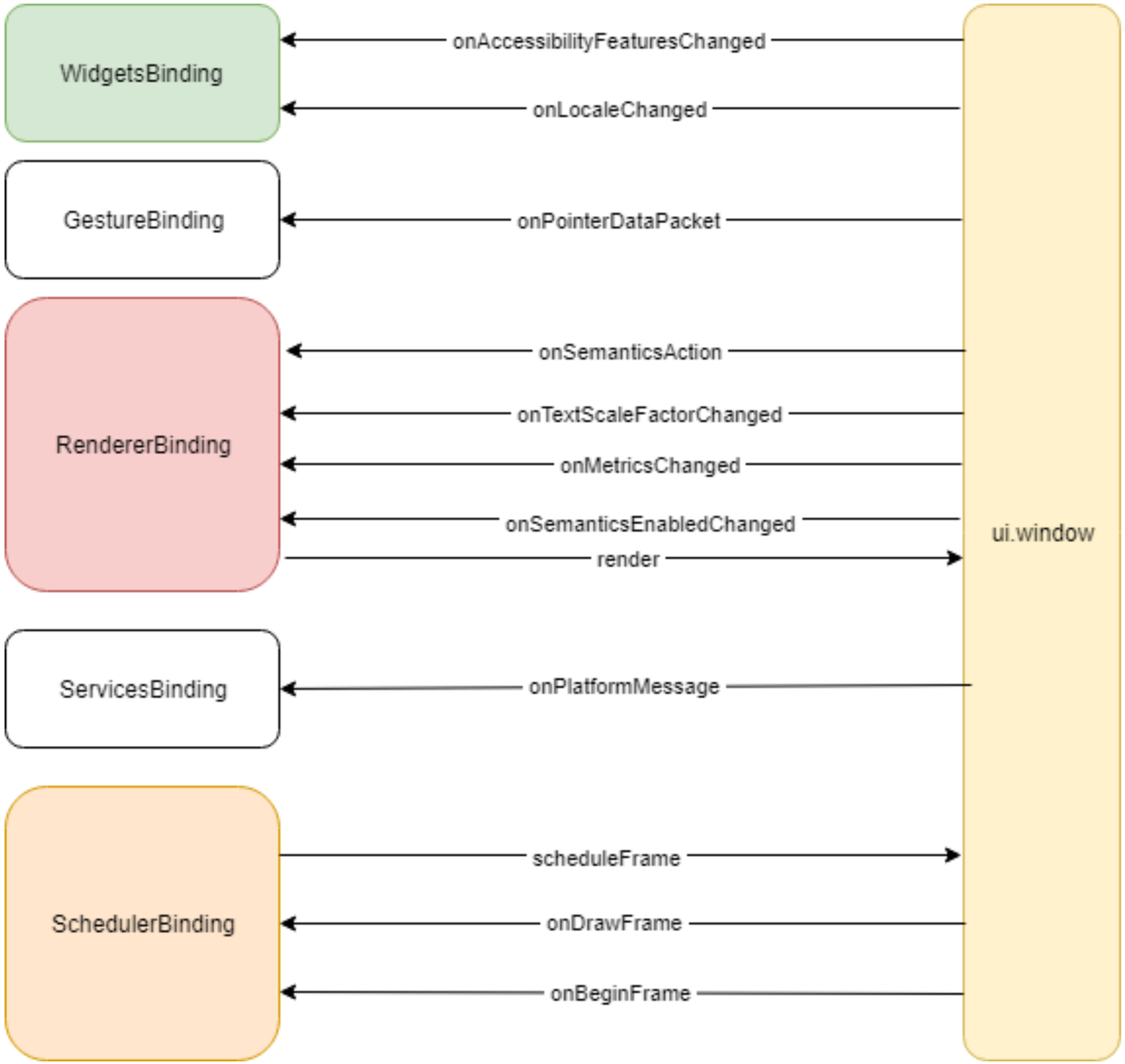
/// A concrete binding for applications based on the Widgets framework.
///
/// This is the glue that binds the framework to the Flutter engine.
class WidgetsFlutterBinding extends BindingBase with

1. GestureBinding,
2. SchedulerBinding,
3. ServicesBinding,
4. PaintingBinding,
5. SemanticsBinding,
6. RendererBinding,
7. WidgetsBinding {

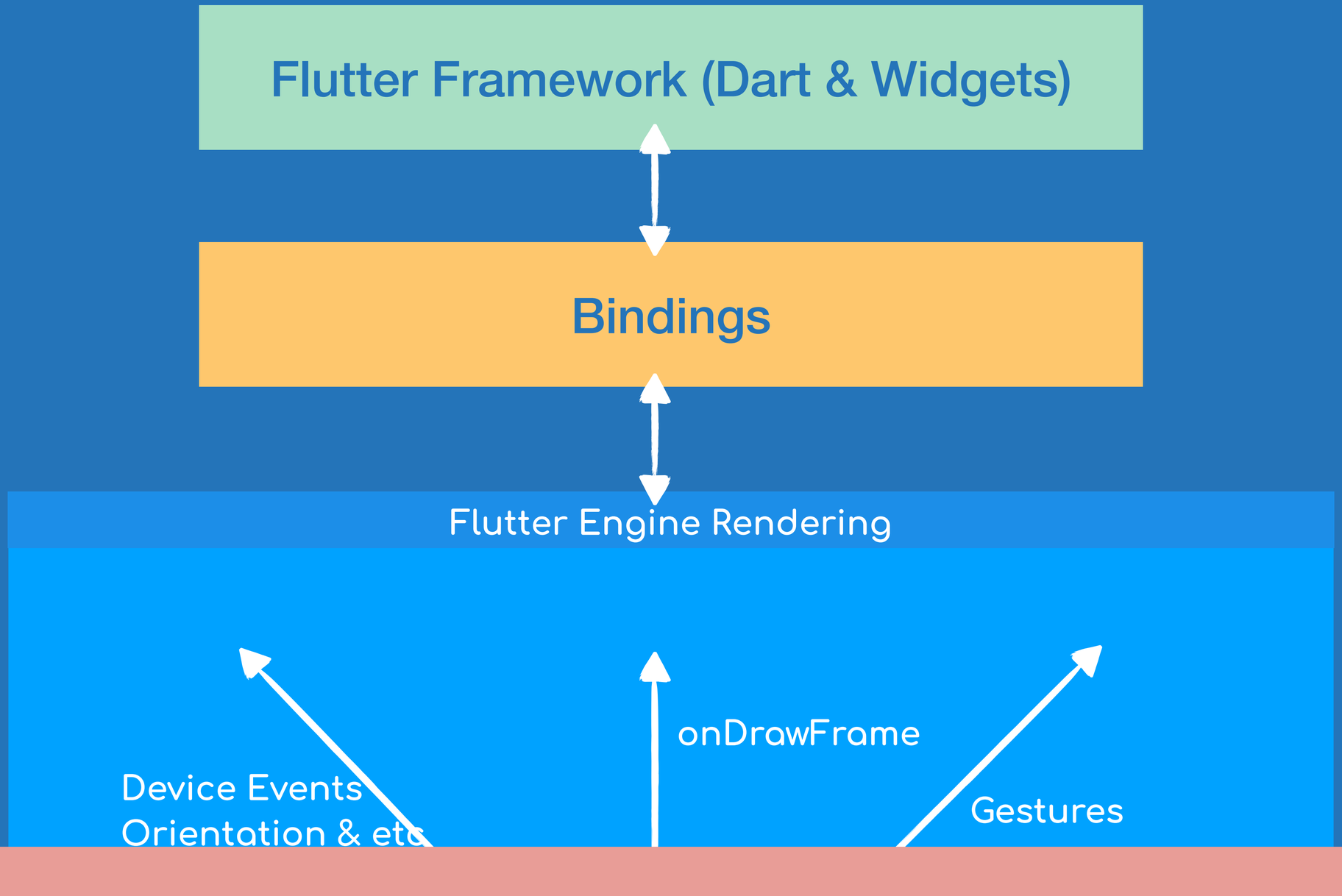
Bindings


abstract class BindingBase {
Choose Subclass of BindingBase (9 classes found)
RendererBinding Dart Packages
PaintingBinding Dart Packages
SchedulerBinding Dart Packages
GestureBinding Dart Packages
ServicesBinding Dart Packages
RenderingFlutterBinding Dart Packages
WidgetsBinding Dart Packages
WidgetsFlutterBinding Dart Packages
SemanticsBinding Dart Packages

Bindings


WidgetsBinding
onAccessibilityFeaturesChanged
onLocaleChanged
GestureBinding
onPointerDataPacket
RendererBinding
onSemanticsAction
onTextScaleFactorChanged
onMetricsChanged
onSemanticsEnabledChanged
render
ServicesBinding
onPlatformMessage
SchedulerBinding
scheduleFrame
onDrawFrame
onBeginFrame
ui.window

Bindings


Flutter Framework (Dart & Widgets)
Bindings
Flutter Engine Rendering
Device Events
Orientation & etc
onDrawFrame
Gestures

Where is relation between Widget & Render Object?

Widget class

```
@immutable
abstract class Widget extends DiagnosticableTree

...
/// Inflates this configuration to a concrete instance.
///
/// A given widget can be included in the tree zero or more times. In particular
/// a given widget can be placed in the tree multiple times. Each time a widget
/// is placed in the tree, it is inflated into an [Element], which means a
/// widget that is incorporated into the tree multiple times will be inflated
/// multiple times.
@protected
@factory
Element createElement();
```

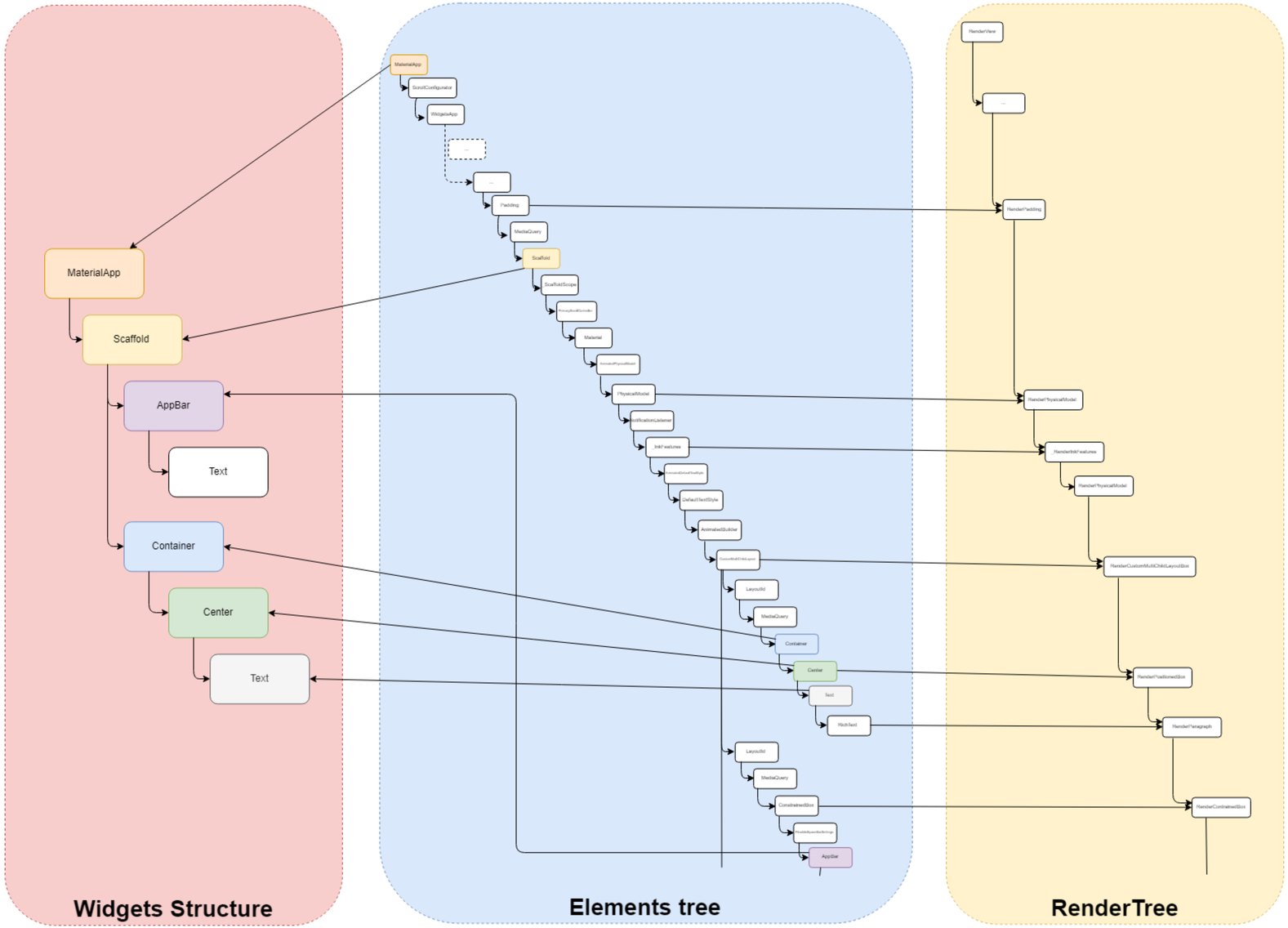
Element class

```
/// An instantiation of a [Widget] at a particular location in the tree.
///
/// Widgets describe how to configure a subtree but the same widget can be used
/// to configure multiple subtrees simultaneously because widgets are immutable.
/// An [Element] represents the use of a widget to configure a specific location
/// in the tree. Over time, the widget associated with a given element can
/// change, for example, if the parent widget rebuilds and creates a new widget
/// for this location.


abstract class Element extends DiagnosticableTree implements BuildContext {
```

Element class

```
abstract class Element extends DiagnosticableTree implements BuildContext {


..
/// Create an element for the given widget and add it as a child of this
/// element in the given slot.
@protected
Element inflateWidget(Widget newWidget, dynamic newSlot) {




/// The render object at (or below) this location in the tree.
///
/// If this object is a [RenderObjectElement], the render object is the one at
/// this location in the tree. Otherwise, this getter will walk down the tree
/// until it finds a [RenderObjectElement].
RenderObject get renderObject
```

Elements Tree



Element types - ComponentElement

ComponentElement
Эти элементы напрямую не отвечают за отрисовку какой-либо части отображения.

```
/// An [Element] that composes other [Element]s.
///
/// Rather than creating a [RenderObject] directly, a [ComponentElement] creates
/// [RenderObject]s indirectly by creating other [Element]s.
///
/// Contrast with [RenderObjectElement].
abstract class ComponentElement extends Element {
```

Choose Subclass of **ComponentElement** (7 classes found)

StatefulElement	Dart Packages
ProxyElement	Dart Packages
StatelessElement	Dart Packages
InheritedElement	Dart Packages
ParentDataElement	Dart Packages
InheritedNotifierElement	Dart Packages

Element types - RenderObjectElement

RenderObjectElement
Данные элементы отвечают за части отображаемого изображения на экране.

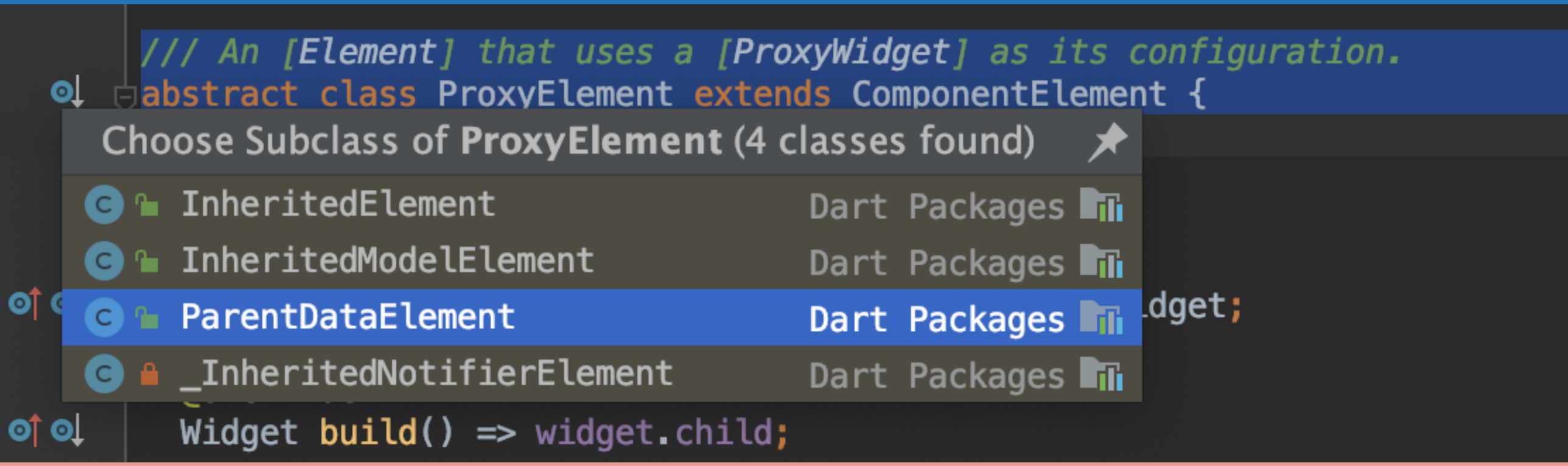
```
///
/// If a [RenderObjectElement] object has any children [Element]s, it must
/// expose them in its implementation of the [visitChildren] method. This method
/// is used by many of the framework's internal mechanisms, and so should be
/// fast. It is also used by the test framework and [debugDumpApp].
abstract class RenderObjectElement extends Element {
```

Choose Subclass of **RenderObjectElement** (22 classes found)

_TheatreElement	Dart Packages
_SliverPrototypeExtentListElement	Dart Packages
_SliverOffstageElement	Dart Packages
_TextSelectionToolbarItemsElement	Dart Packages
_DecorationElement	Dart Packages
_TableElement	Dart Packages

Element types - ProxyElement

Proxy - являясь наследниками ComponentElement - также не отвечают за рендеринг, а только за передачу состояния от родителей к своим детям.



Element.markNeedsBuild - invalidation of the state

```
abstract class Element extends DiagnosticableTree implements BuildContext {


..
/// Marks the element as dirty and adds it to the global list of widgets to
/// rebuild in the next frame.
///
/// Since it is inefficient to build an element twice in one frame,
/// applications and widgets should be structured so as to only mark
/// widgets dirty during event handlers before the frame begins, not during
/// the build itself.
void markNeedsBuild() {
```

Итоги

Что такое виджет Flutter?

Stateless & Stateful widgets

Inherited Widgets

Keys & Global Keys

Flutter Engine

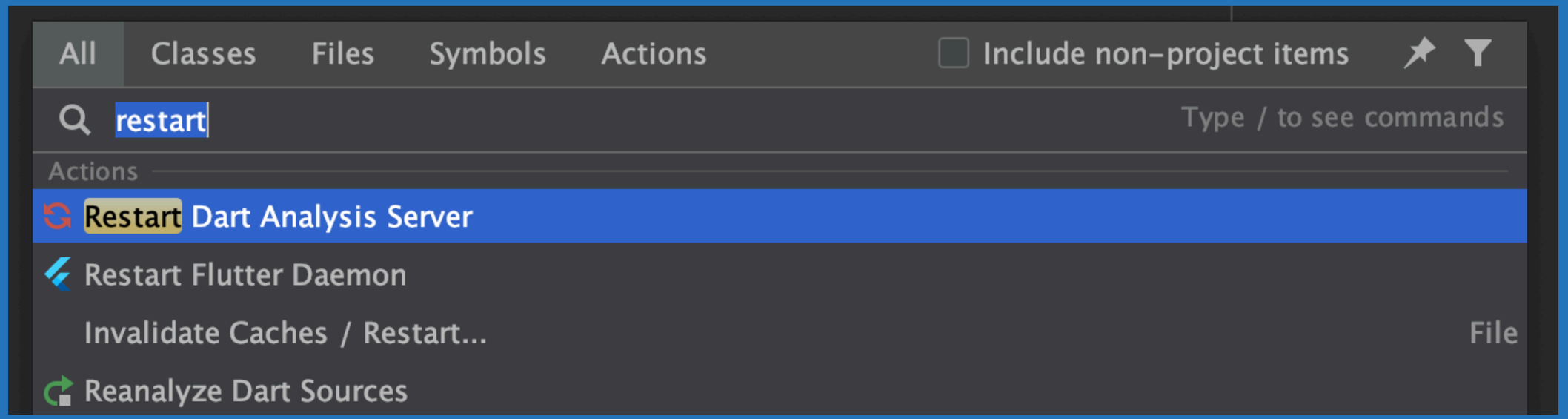
Заполните, пожалуйста,
опрос о занятии по ссылке в чате

Спасибо за внимание!
Приходите на следующие вебинары

Смирнов Андрей wrike

Курс **Мобильная разработка на Flutter**

Проблемы с Android Studio



Из полезного

https://material.io/components

https://flutterexamples.com/