ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо видно && слышно?

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

Flutter Mobile Developer

Тема 10. Flutter. Как устроена анимация во Flutter? Implicit animations

Цель урока

- Научиться работать со анимациями
- Научиться делать простые анимации, используя AnimatedContainer и производные от него
- Научиться использовать Curves для придания физических свойств анимации.

О чем будем говорить

- Что такое анимация и как она устроена
- Что есть для анимации в Flutter Framework
- Implicit Animation Widgets & AnimatedContainer
- Tween & TweenAnimationBuilder
- Animation physics. Curves.

Что такое анимация?

Википедия:

Технические приёмы создания иллюзии движущихся изображений с помощью последовательности неподвижных изображений (кадров), сменяющих друг друга с большой частотой (от 12 кадров в секунду для рисованной анимации до 30 кадров в секунду для компьютерной анимации).

Что такое анимация?

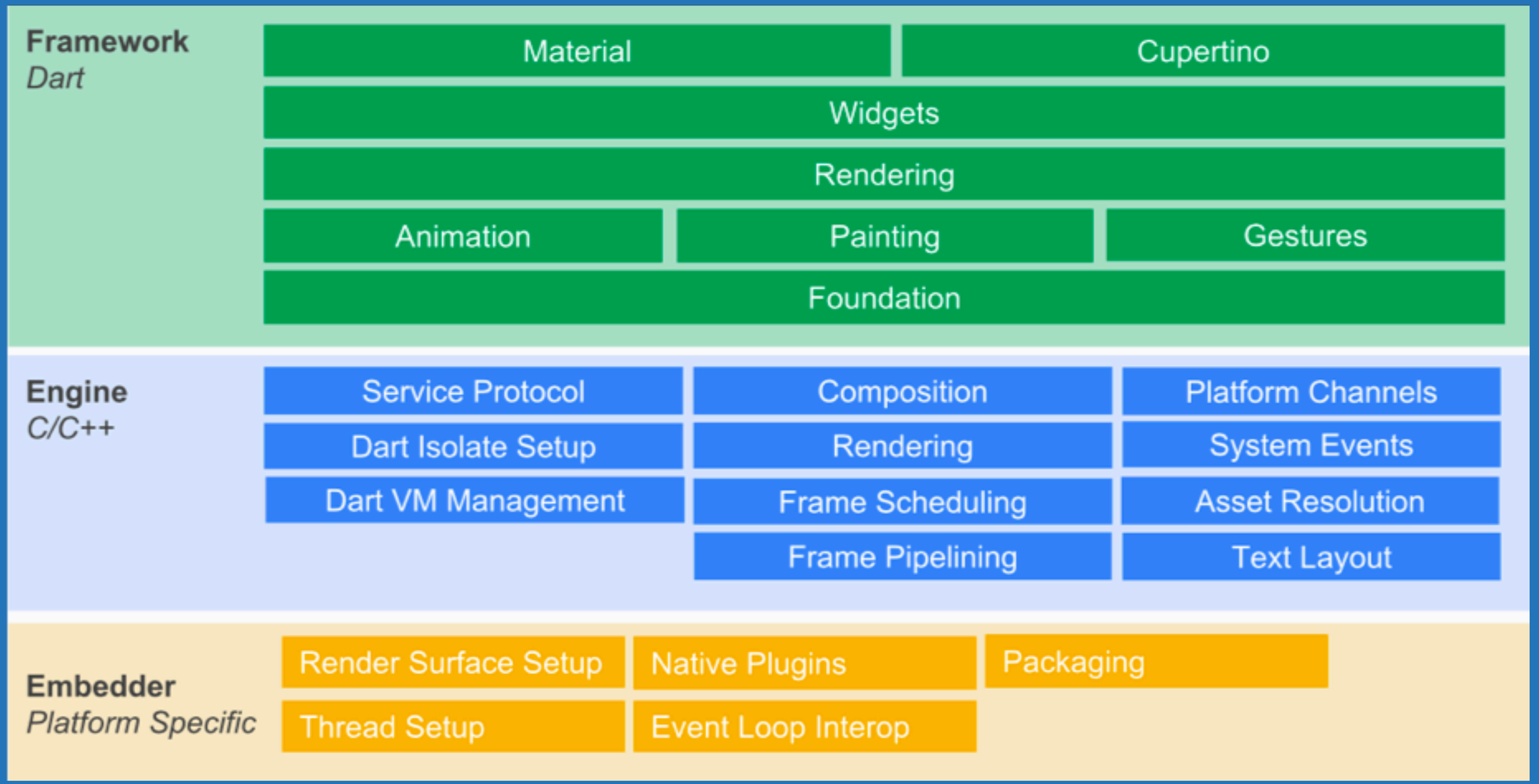
http://bit.ly/whatisanimation

Flutter
https://flutter.dev/docs/codelabs/explicit-animations#what-is-animation

Think about how animations work in a flip-book, or cartoons on TV, or a movie reel. What do these animation technologies have in common?

They create the illusion of motion by rapidly transitioning a single frame that you are viewing to other frames within a pre-defined sequence.
...

Flutter Engine



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:

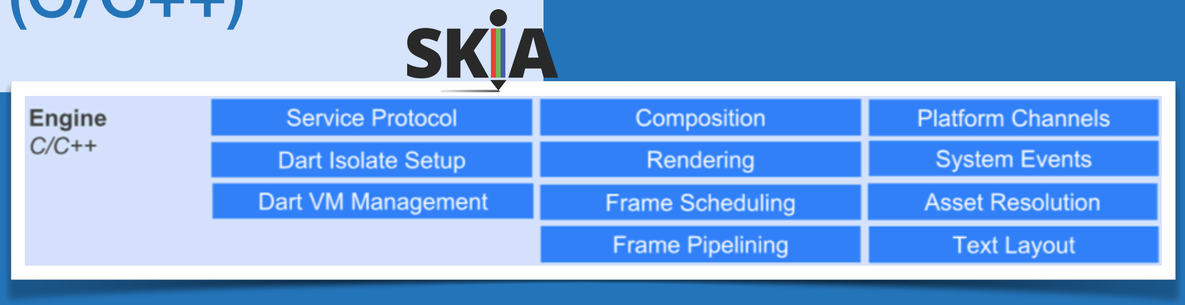
- **Skia** - a 2D graphics rendering library
- **Dart** - a VM for a garbage-collected object-oriented language

Flutter Engine - Window class & ui.window

Flutter Framework (Dart & Widgets)

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

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

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.

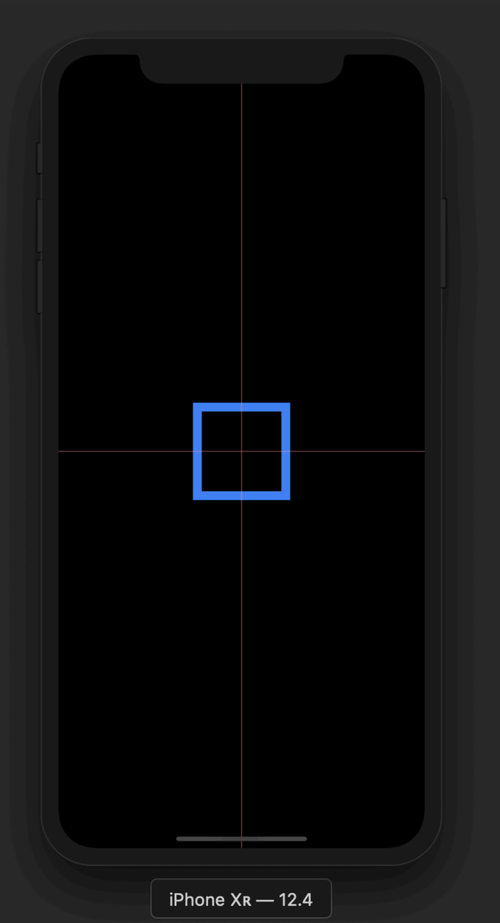
See also:

- SchedulerBinding, the Flutter framework class which manages the scheduling of frames.

main() -> without runApp() ?

main() -> without runApp() ?

```dart
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

drawLine, drawPath & etc.

Canvas

Picture

Scene Builder

Scene

window.**render(scene)**

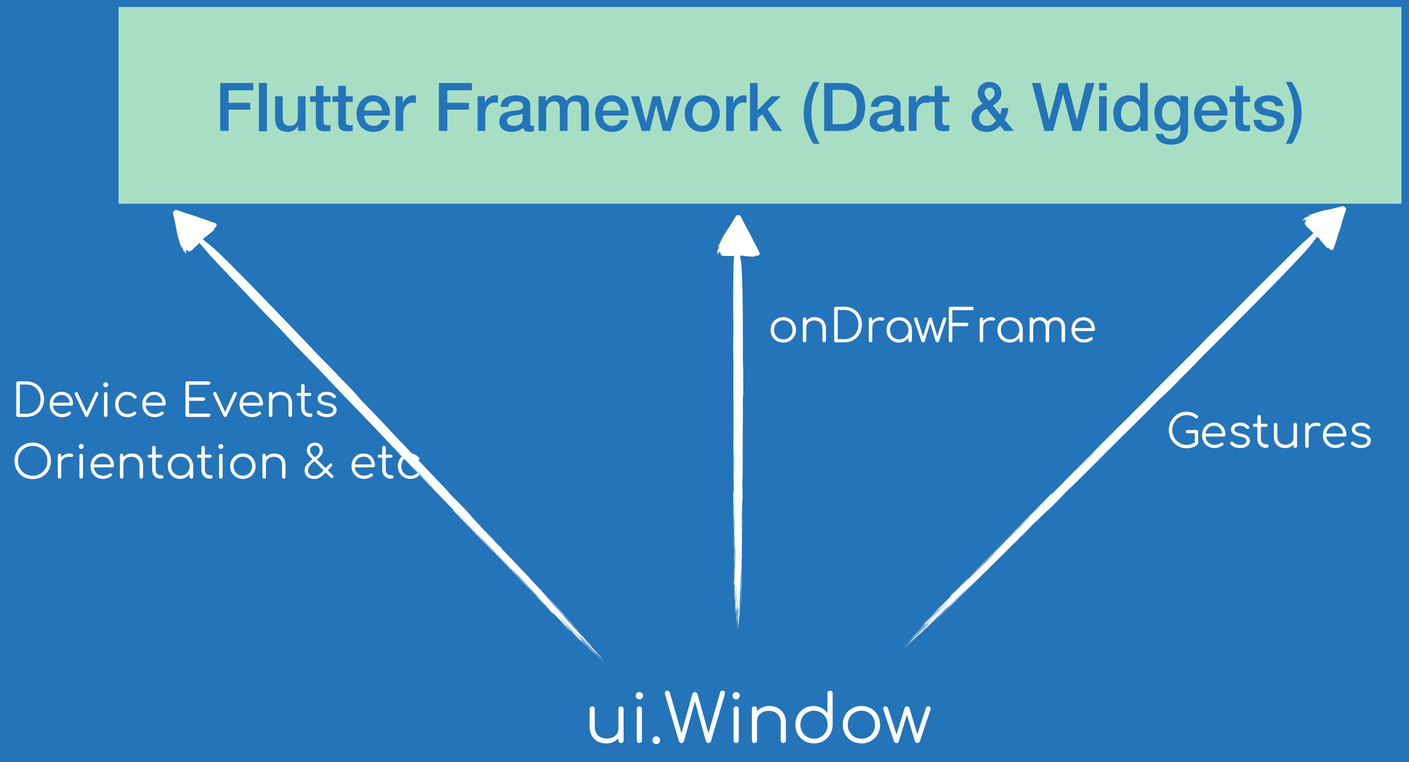
window.onBeginFrame =
callback(timeStamp)

window.render(scene)

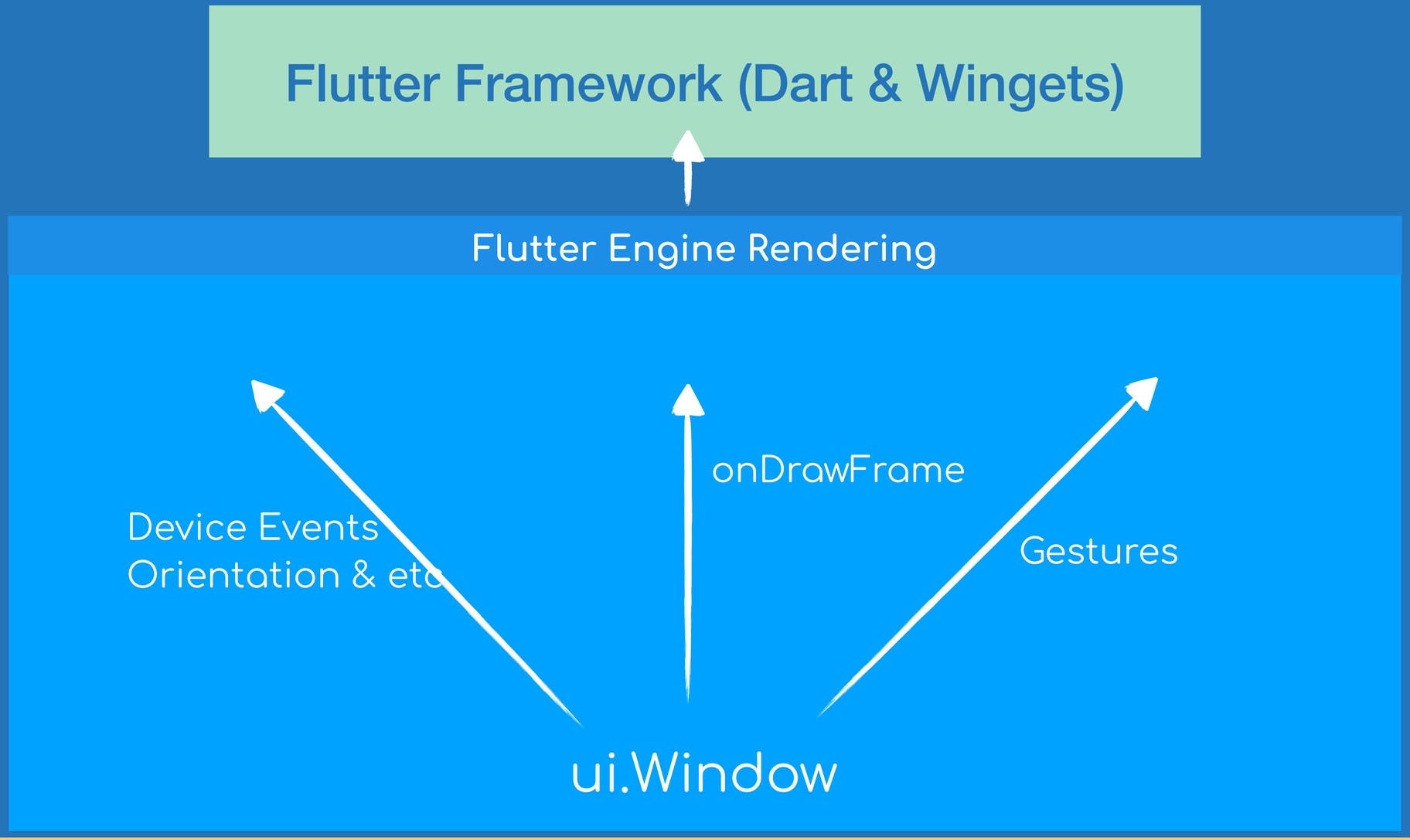
window.scheduleFrame()

ns

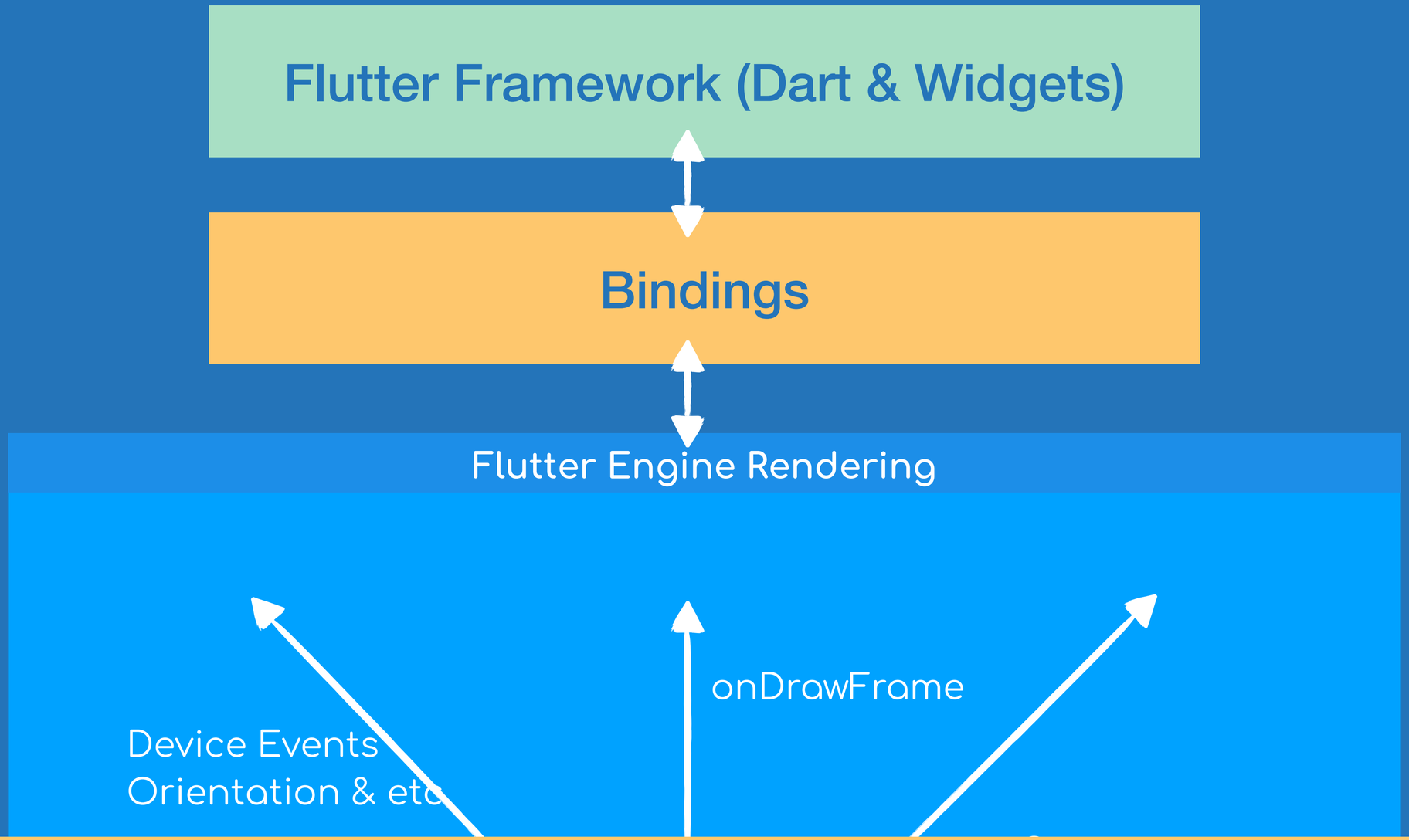
Widgets & Window



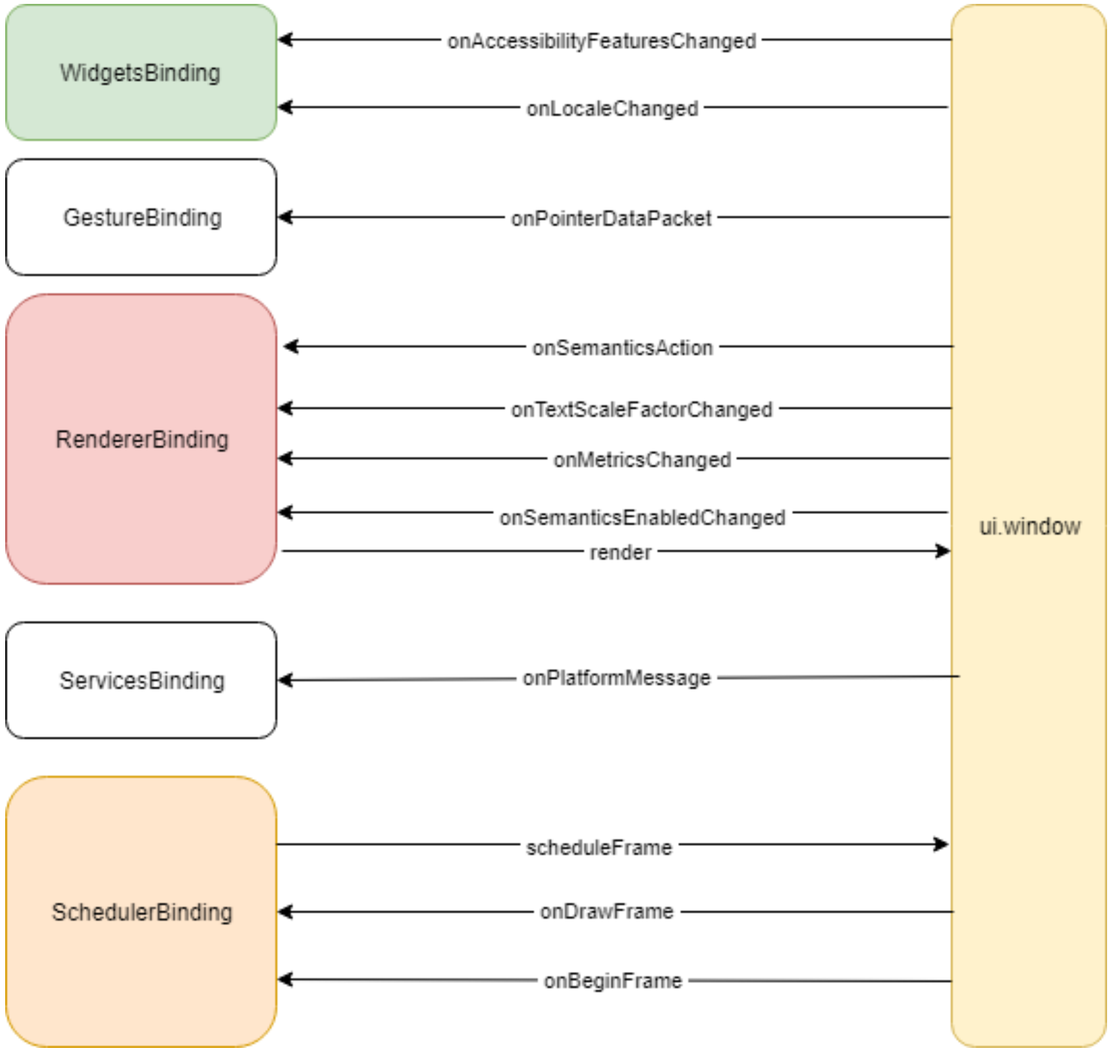
Flutter Engine Rendering



Bindings


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

Bindings


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

Window class - Animation Demo

```
  final Picture picture = recorder.endRecording();

  final double t = timeStamp.inMilliseconds / Duration.millisecondsPerSecond;
  final angleToRotate = math.pi * (t % 1.0);

  final yCenter = physicalBounds.height / 2;
  final xCenter = physicalBounds.width / 2;

  final SceneBuilder sceneBuilder = SceneBuilder()
    ..pushOffset(xCenter, yCenter)
    ..pushTransform((Matrix4.identity()..rotateZ(angleToRotate)).storage)
    ..addPicture(Offset.zero, picture);

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

Что такое Ticker?

Ticker: A ticker is a class that listens to frameCallback and calls a tick function that passes the elapsed duration between the current frame and the last frame to the ticker listener. In our case the controller.

VSync

Вертика́льная синхрониза́ция (англ. *V-Sync*) — синхронизация кадровой частоты в компьютерной игре с частотой вертикальной развёртки монитора. При этом максимальный FPS с вертикальной синхронизацией приравнивается к частоте обновления монитора. Если FPS ниже частоты обновления монитора, то во избежание ещё большей потери производительности следует включить тройную буферизацию.

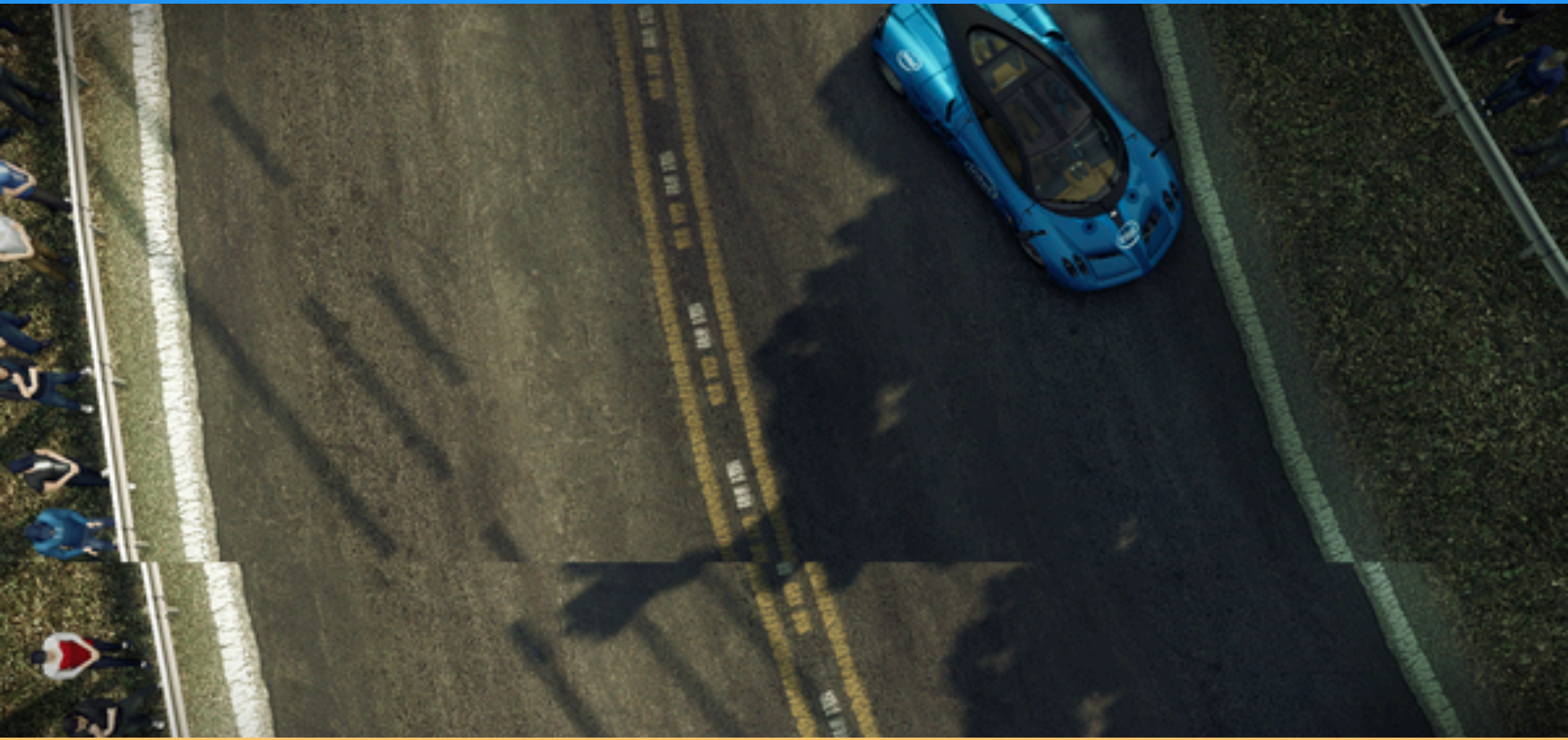
Применение [править | править код]

- возможно избавление от «рывков», когда FPS резко скачет
- убирает подёргивания изображения
- в играх позволяет видеокарте работать не на полную мощь, тем самым значительно снизив температуру видеокарты и шум, особенно если установленная система охлаждения на видеокарте слабая и шумная
- в нетребовательных приложениях снижает максимальное энергопотребление видеокарты

Включение функции V-Sync [править | править код]

Вертикальную синхронизацию можно включить как непосредственно в настройках большинства игр, так и в настройках драйвера видеокарты, причём можно задать V-Sync для какого-либо конкретного приложения либо для всех.

VSync



Что такое Ticker?

Flutter › scheduler › Ticker class

CLASSES

FrameTiming

Priority

Ticker

TickerFuture

TickerProvider

MIXINS

SchedulerBinding

PROPERTIES

debugPrintBeginFrameBa...

debugPrintEndFrameBan...

debugPrintScheduleFram...

timeDilation

Ticker class

Null safety

Calls its callback once per animation frame.

When created, a ticker is initially disabled. Call start to enable the ticker.

A Ticker can be silenced by setting muted to true. While silenced, time still elapses, and start and stop can still be called, but no callbacks are called.

By convention, the start and stop methods are used by the ticker's consumer, and the muted property is controlled by the TickerProvider that created the ticker.

Tickers are driven by the SchedulerBinding. See SchedulerBinding.scheduleFrameCallback.

Constructors

Ticker(TickerCallback _onTick, {String? debugLabel})
Creates a ticker that will call the provided callback once per frame while running. [...]

Что такое Ticker?

Flutter › scheduler › TickerProvider abstract class

CLASSES

FrameTiming

Priority

Ticker

TickerFuture

TickerProvider

MIXINS

SchedulerBinding

PROPERTIES

debugPrintBeginFrameBa...

debugPrintEndFrameBan...

debugPrintScheduleFram...

timeDilation

FUNCTIONS

debugAssertAllScheduler...

TickerProvider class Null safety

An interface implemented by classes that can vend Ticker objects.

Tickers can be used by any object that wants to be notified whenever a frame triggers, but are most commonly used indirectly via an AnimationController. AnimationControllers need a TickerProvider to obtain their Ticker. If you are creating an AnimationController from a State, then you can use the TickerProviderStateMixin and SingleTickerProviderStateMixin classes to obtain a suitable TickerProvider. The widget test framework WidgetTester object can be used as a ticker provider in the context of tests. In other contexts, you will have to either pass a TickerProvider from a higher level (e.g. indirectly from a State that mixes in TickerProviderStateMixin), or create a custom TickerProvider subclass.

Implementers

TestVSync, WidgetTester

Constructors

TickerProvider()

Abstract const constructor. This constructor enables subclasses to provide const constructors so that they can be used in const expressions.

Идем дальше

Что такое анимация и как она устроена

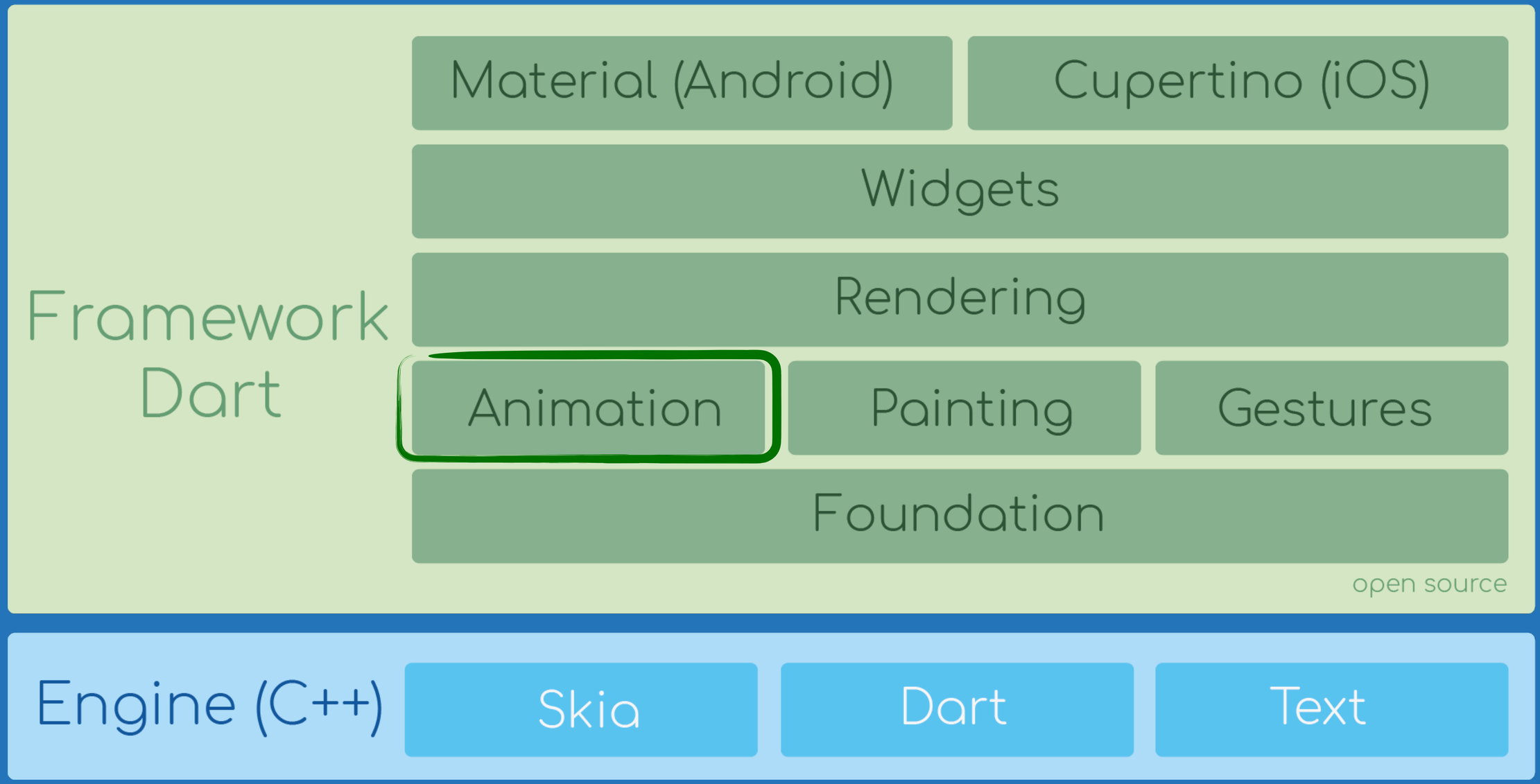
Что есть для анимации в Flutter Framework

Implicit Animation Widgets & AnimatedContainer

Tween & TweenAnimationBuilder

Animation physics. Curves.

Что есть для анимации во Flutter?



Что есть для анимации во Flutter?

Animation library - The Flutter animation system

To use, import package:flutter/animation.dart.

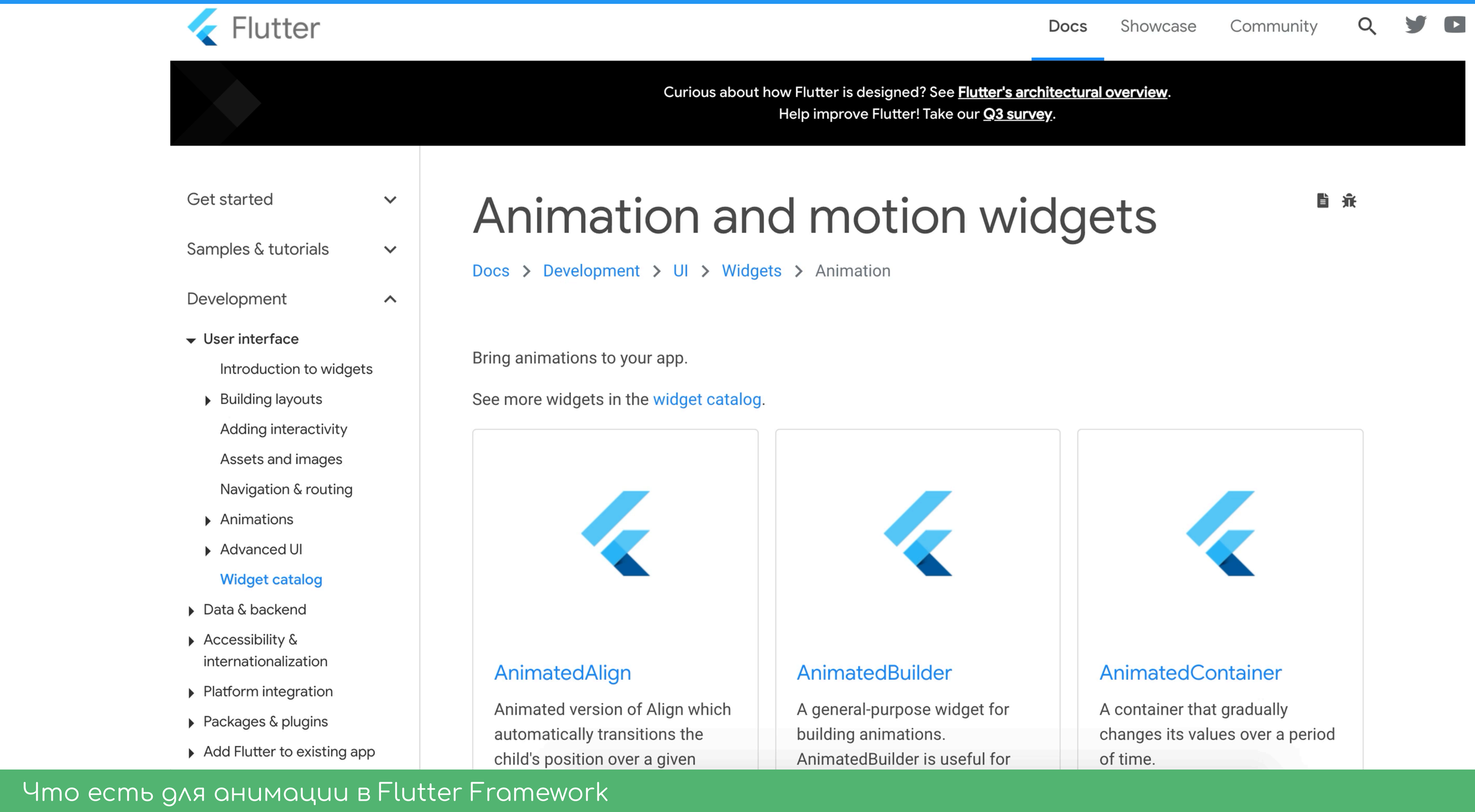
This library provides basic building blocks for implementing animations in Flutter. Other layers of the framework use these building blocks to provide advanced animation support for applications.

For example, the widget library includes ImplicitlyAnimatedWidgets and AnimatedWidgets that make it easy to animate certain properties of a Widget.

If those animated widgets are not sufficient for a given use case, the basic building blocks provided by this library can be used to implement custom animated effects.

This library depends only on core Dart libraries and the physics.dart library.

Что есть для анимации во Flutter?



Принятие решения

https://flutter.dev/docs/development/ui/animations

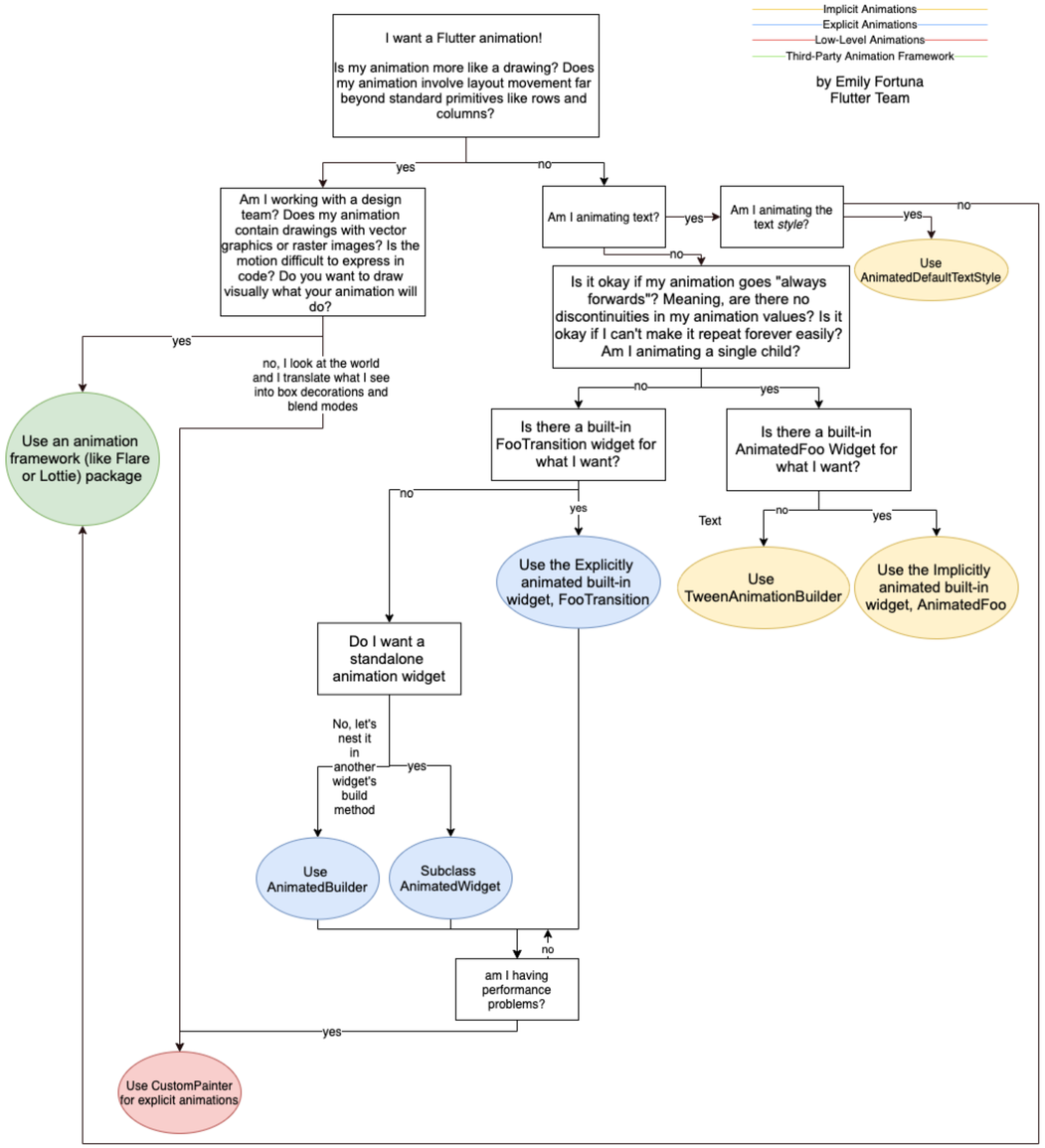


Схема принятия решения

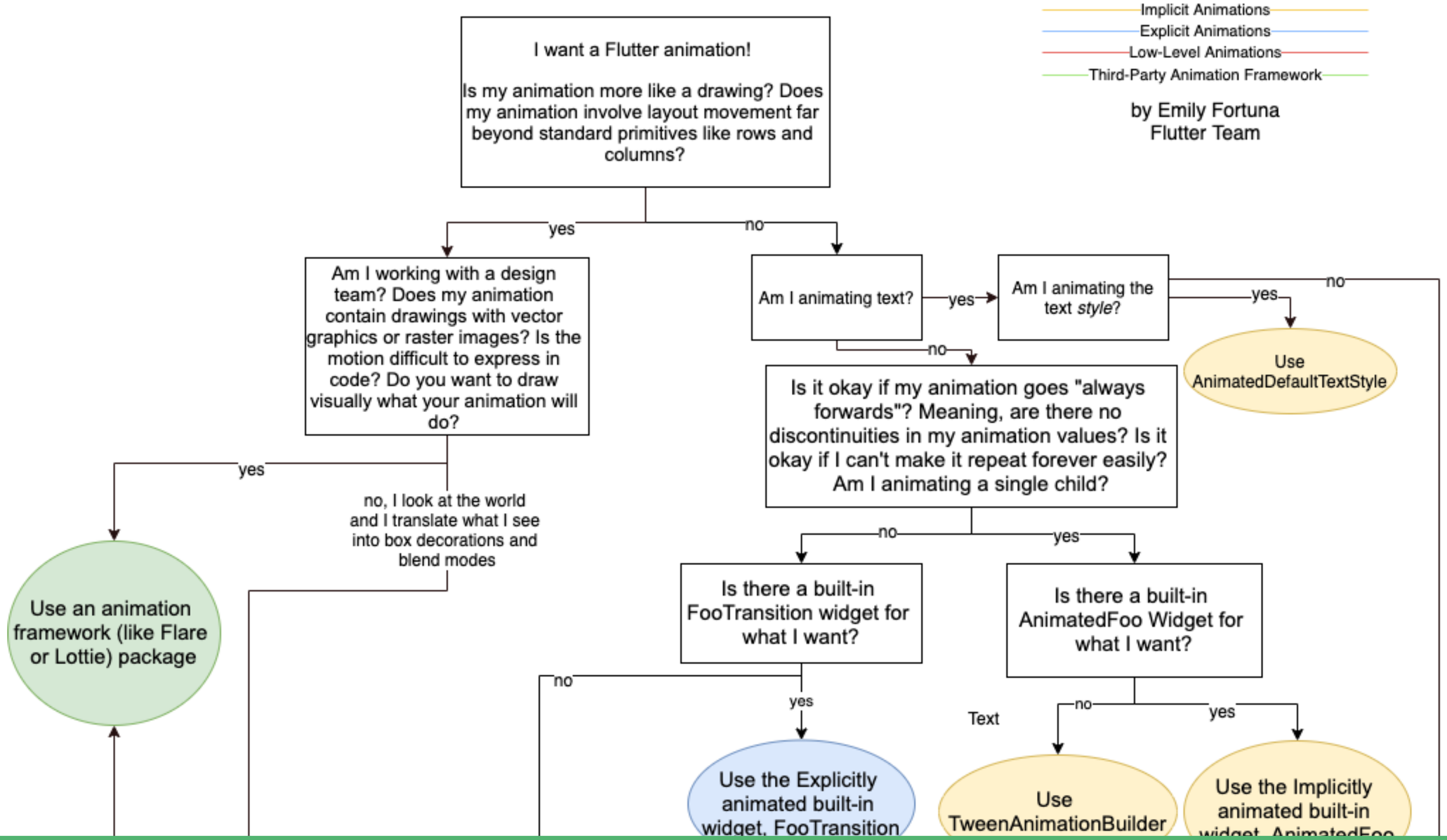


Схема принятия решения

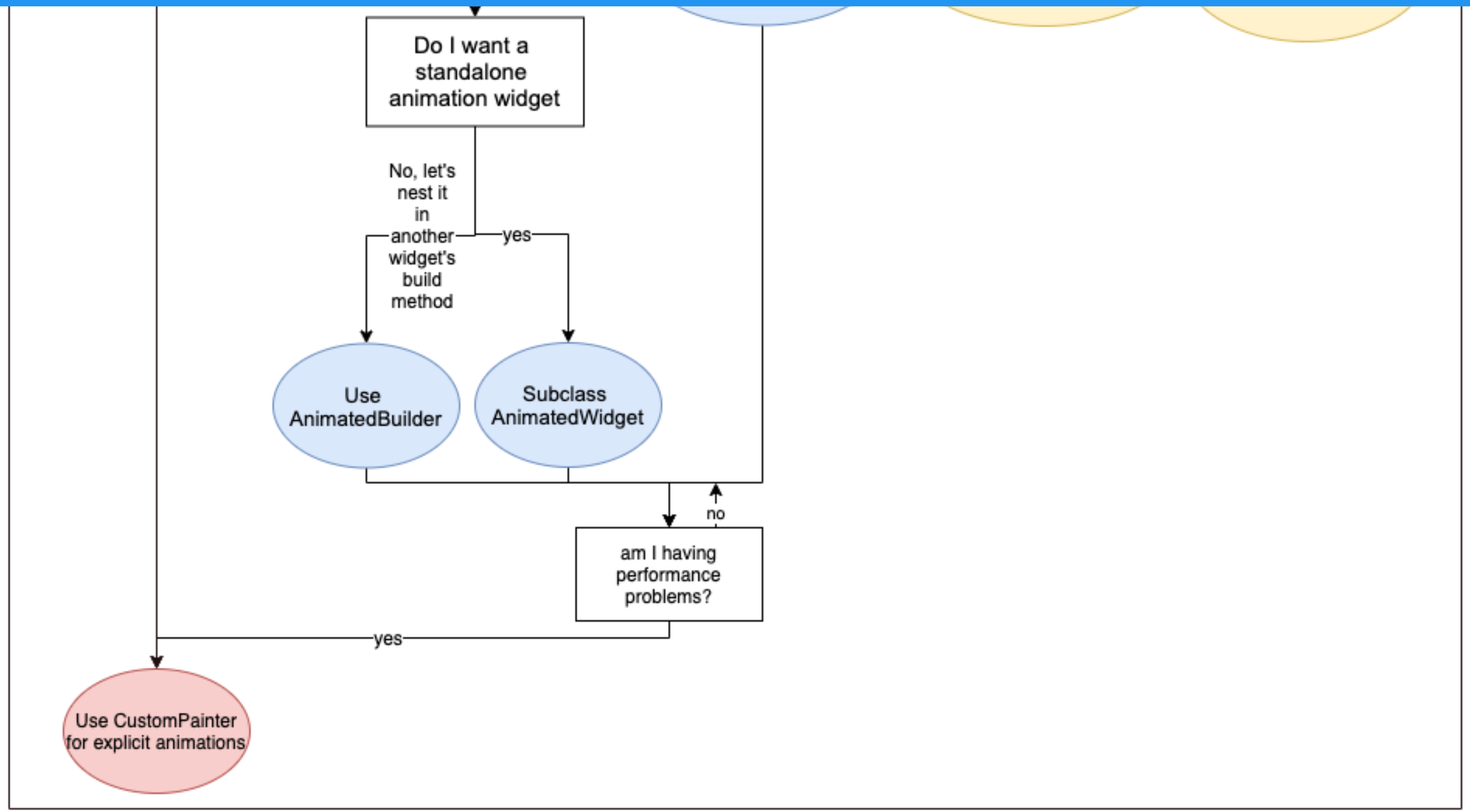


Схема принятия решения

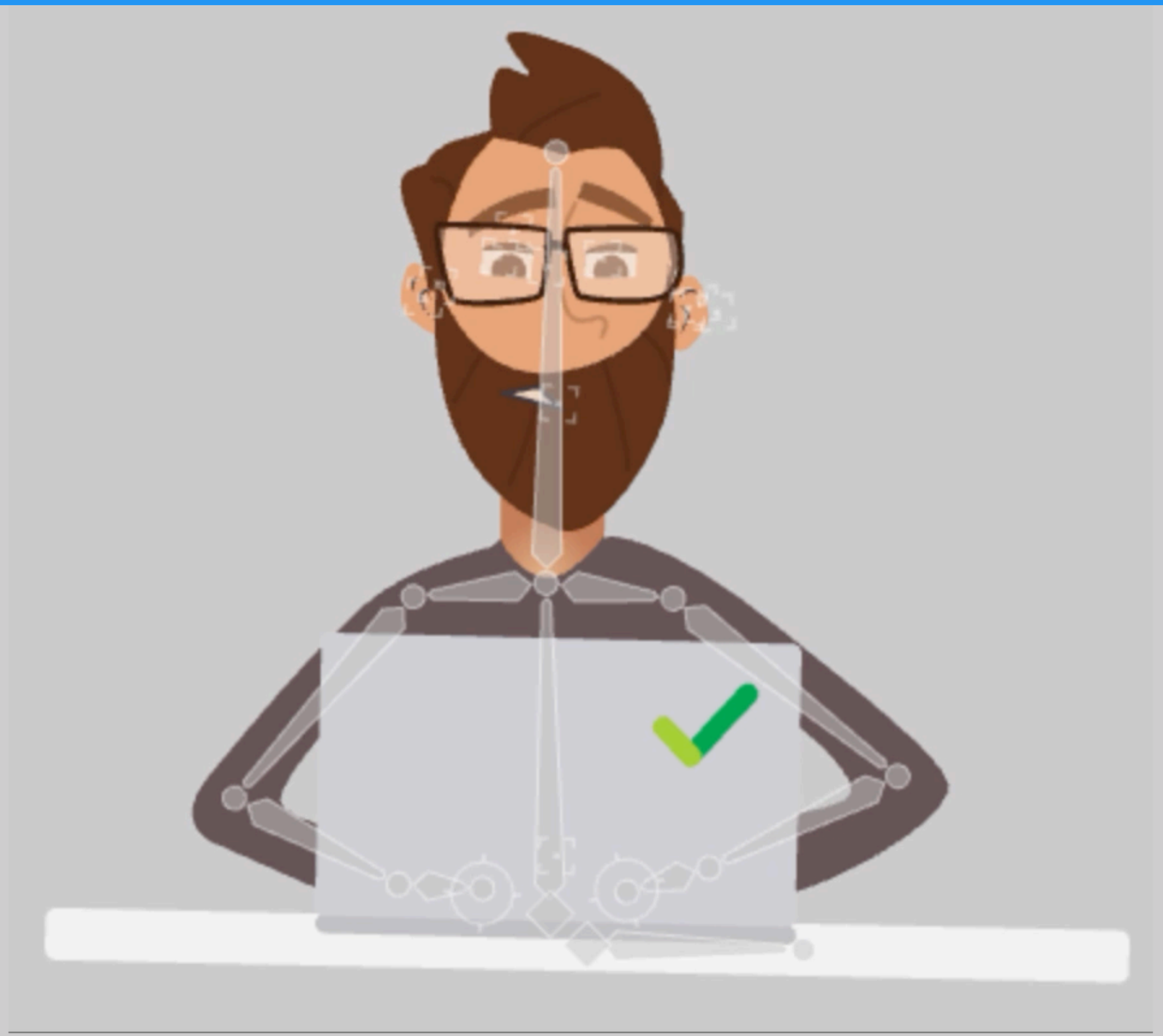
Первый вопрос:

Моя анимация выглядит как мультипликация? (Выглядит так, что каким-либо алгоритмом такая анимация вообще не описывается)

ответили Да? Ваш выбор - Drawing Based Animation - RiveApp (Flare) / Lottie (Adobe After Effects)

Нет? Смотрим на пакет package:flutter/animation.dart

Drawing Based Animation

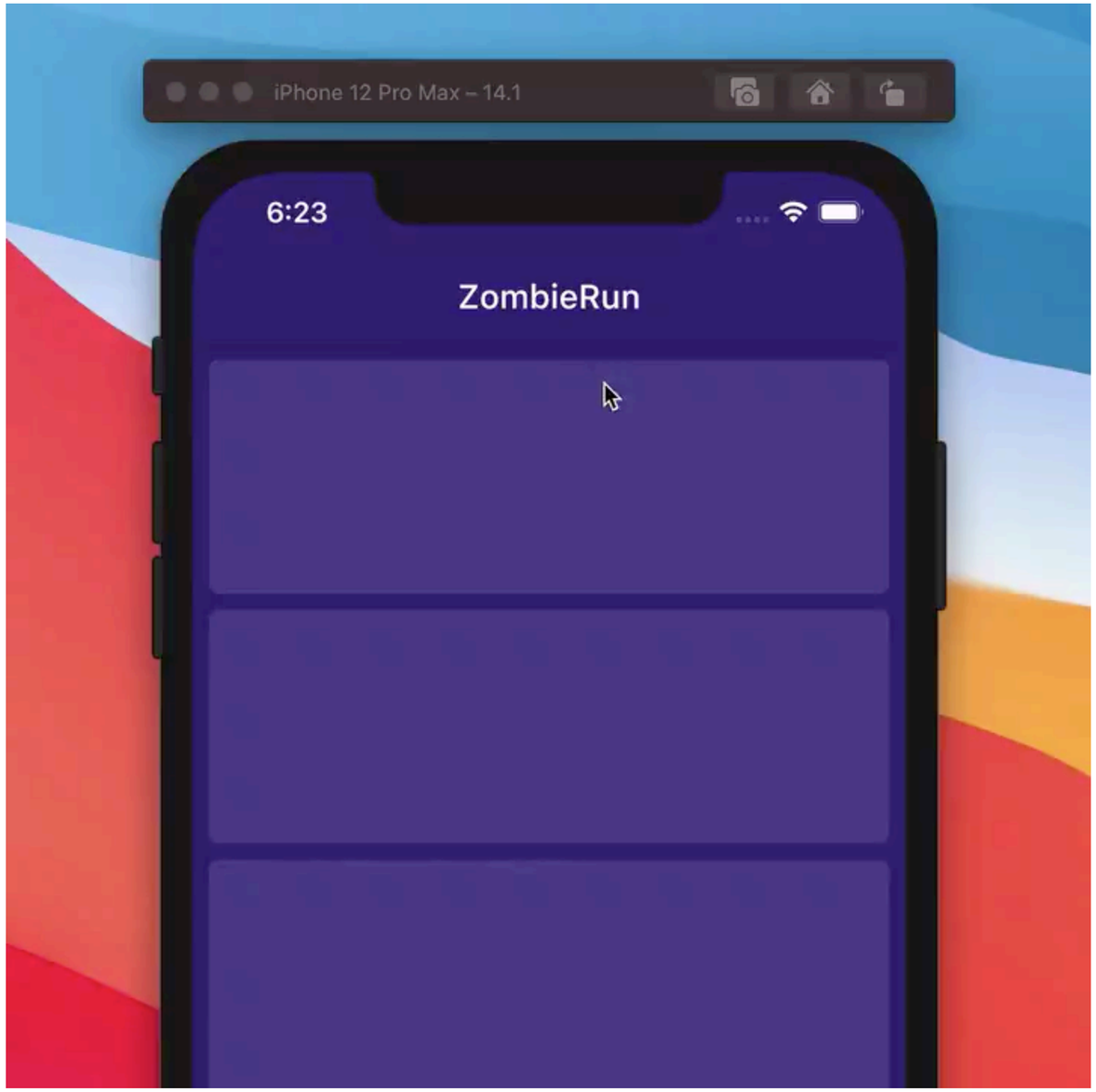


2D

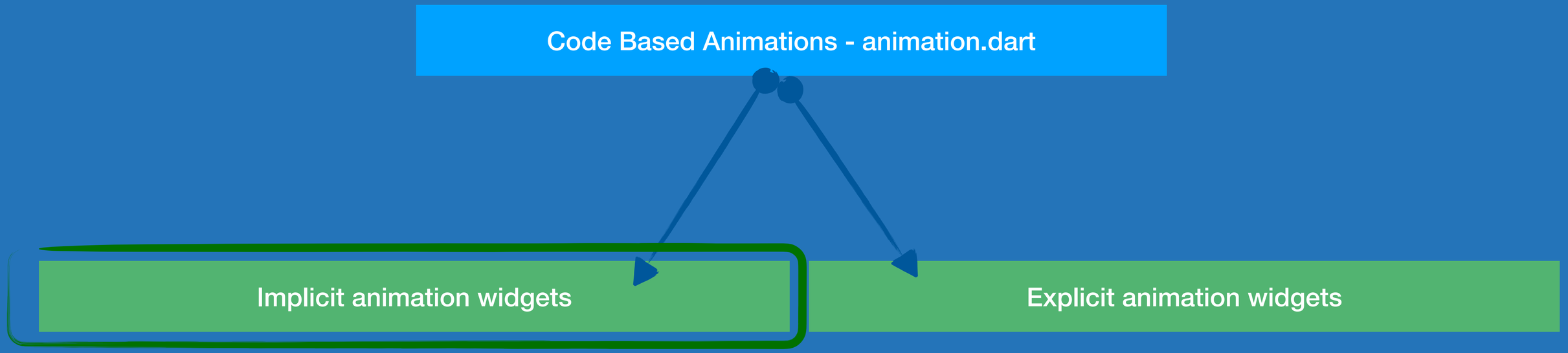
Rive App

Drawing Based Animation

Drawing Based Animation



Code Based Animations - animation.dart



Идем дальше

Что такое анимация и как она устроена

Что есть для анимации в Flutter Framework

Implicit Animation Widgets & AnimatedContainer

Tween & TweenAnimationBuilder

Animation physics. Curves.

Implicit Animations

What are implicit animations?

With Flutter's animation library, you can add motion and create visual effects for the widgets in your UI.
One widget set in the library manages animations for you.
These widgets are collectively referred to as implicit animations, or implicitly animated widgets, deriving their name from the ImplicitlyAnimatedWidget class that they implement.

With implicit animations, you can animate a widget property by setting a target value; whenever that target value changes, the widget animates the property from the old value to the new one. In this way, implicit animations trade control for convenience—they manage animation effects so that you don't have to.

https://flutter.dev/docs/codelabs/implicit-animations

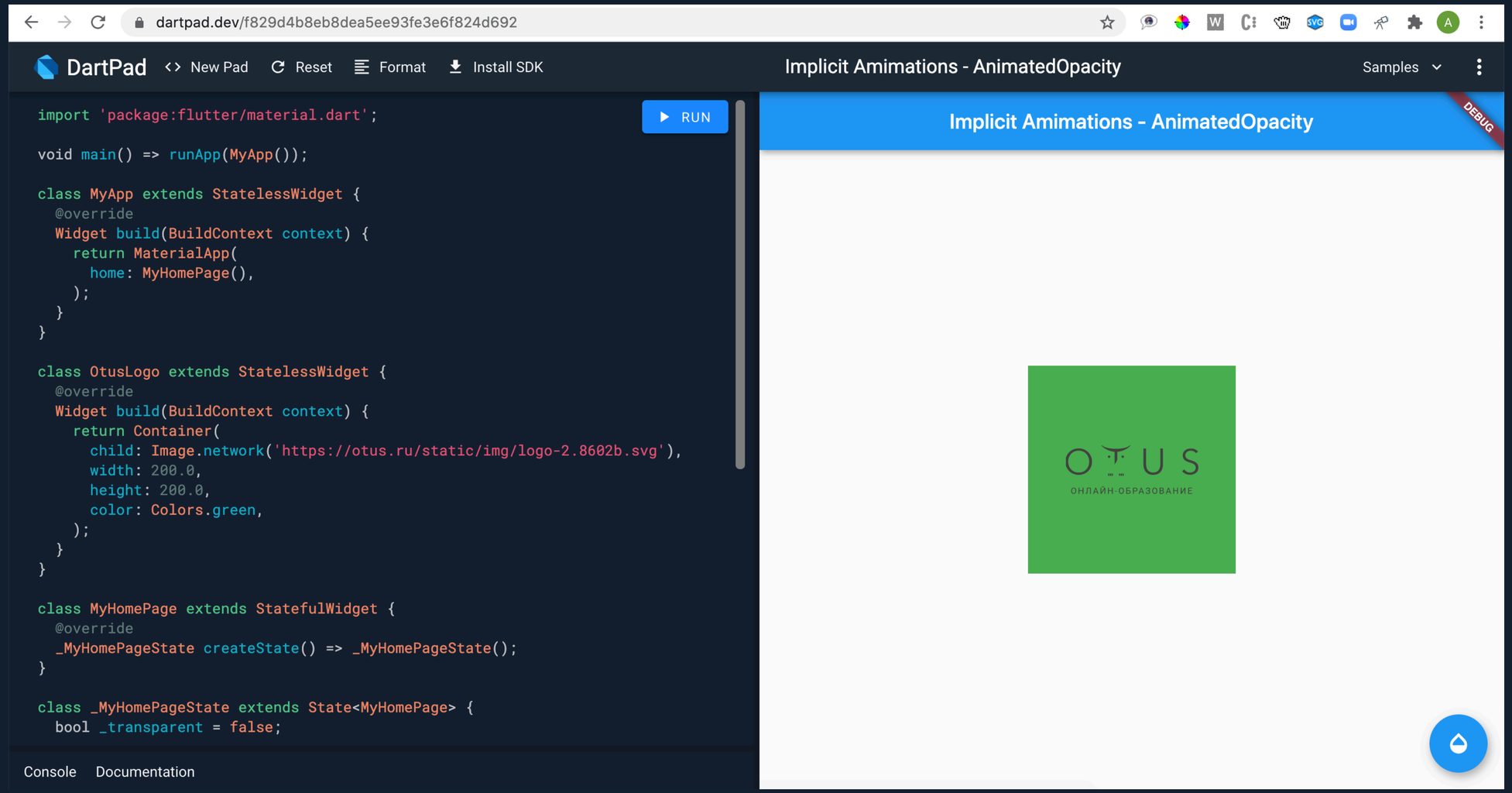
Common implicitly animated widgets

Common implicitly animated widgets

A number of implicitly animated widgets ship with the framework. They are usually named `AnimatedFoo`, where `Foo` is the name of the non-animated version of that widget. Commonly used implicitly animated widgets include:

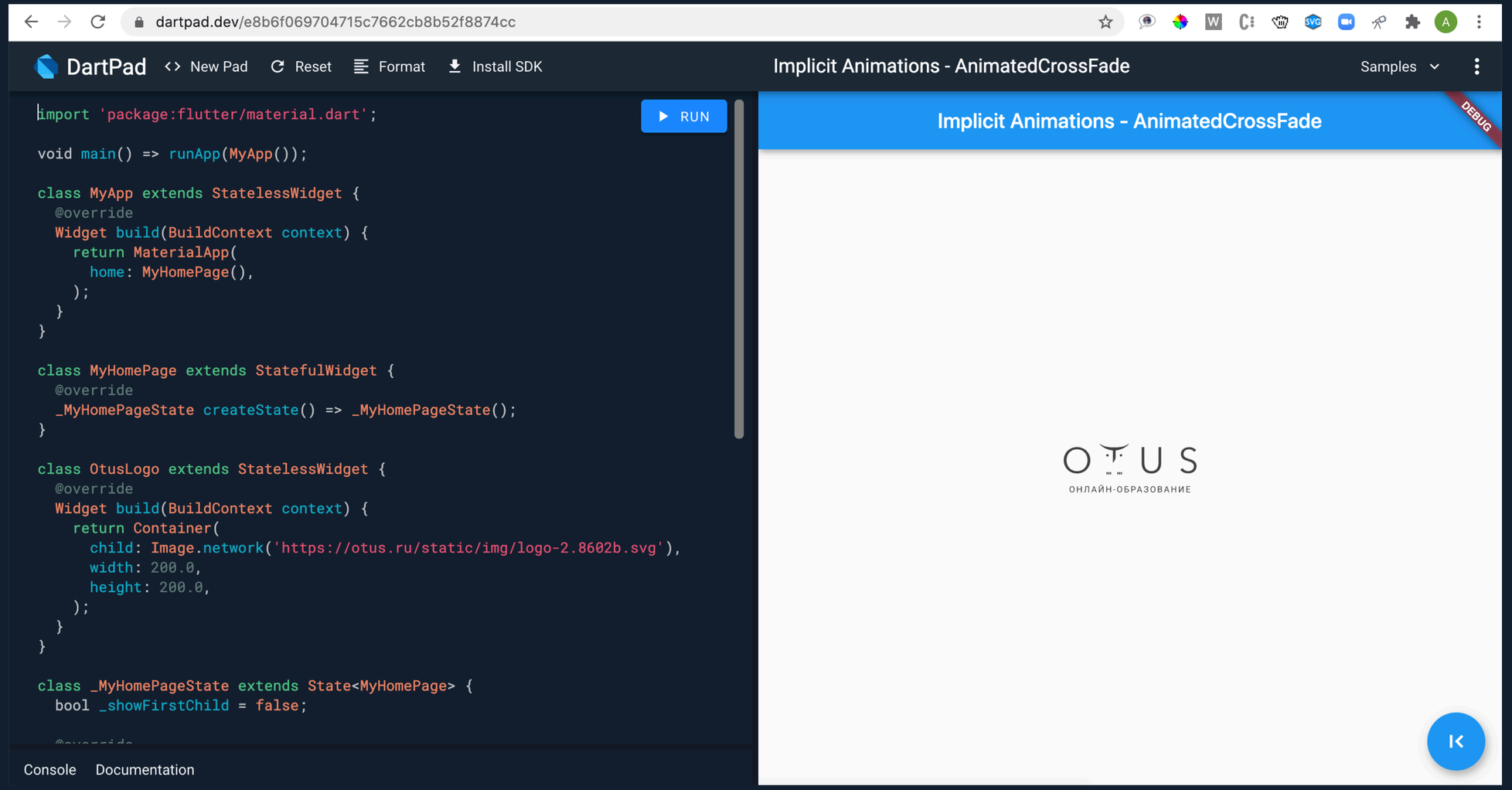
- TweenAnimationBuilder, which animates any property expressed by a Tween to a specified target value.
- AnimatedAlign, which is an implicitly animated version of Align.
- AnimatedContainer, which is an implicitly animated version of Container.
- AnimatedDefaultTextStyle, which is an implicitly animated version of DefaultTextStyle.
- AnimatedOpacity, which is an implicitly animated version of Opacity.
- AnimatedPadding, which is an implicitly animated version of Padding.
- AnimatedPhysicalModel, which is an implicitly animated version of PhysicalModel.
- AnimatedPositioned, which is an implicitly animated version of Positioned.
- AnimatedPositionedDirectional, which is an implicitly animated version of PositionedDirectional.
- AnimatedTheme, which is an implicitly animated version of Theme.
- AnimatedCrossFade, which cross-fades between two given children and animates itself between their sizes.
- AnimatedSize, which automatically transitions its size over a given duration.
- AnimatedSwitcher, which fades from one widget to another.

Implicit Animations - AnimatedOpacity



http://bit.ly/AnimatedOpacity

Implicit Animations - AnimatedCrossFade



http://bit.ly/AnimatedCrossFade

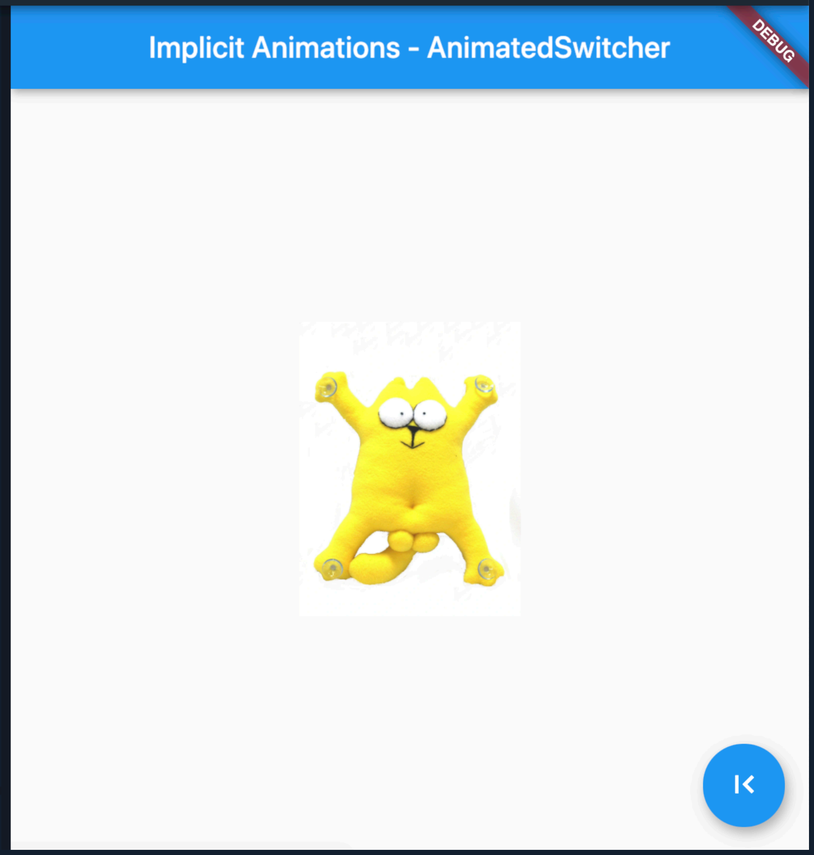
Implicit Animations - AnimatedSwitcher

DartPad New Pad Reset Format Install SDK Implicit Animations - AnimatedSwitcher Samples

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class CatLogo extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      child:
Image.network('https://cdn1.ozone.ru/multimedia/wc1200/1027466419.jpg'),
      width: 200.0,
      height: 200.0,
    );
  }
}
```

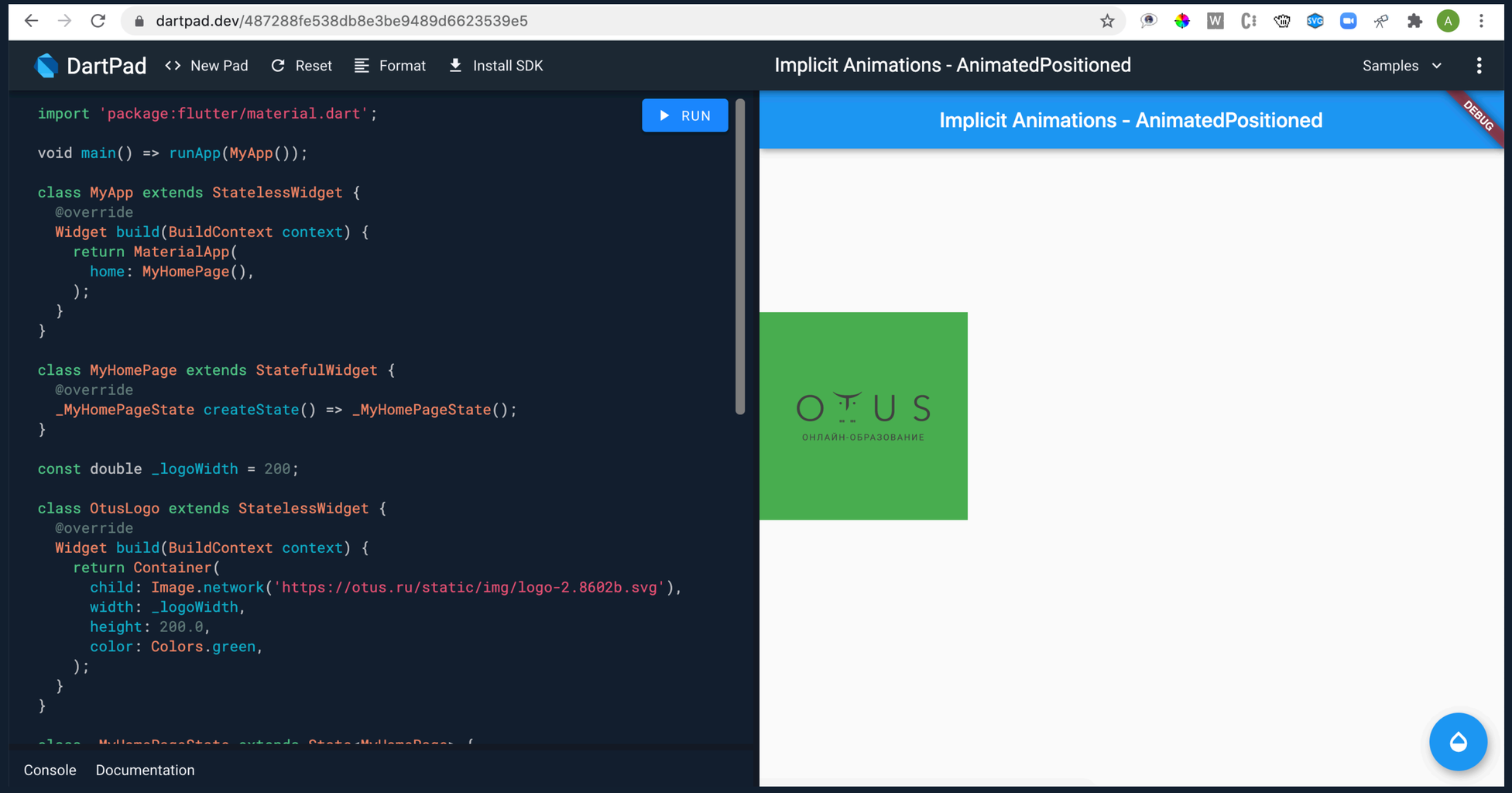
RUN

Console Documentation



http://bit.ly/AnimatedSwitcher

Implicit Animations - AnimatedPositioned

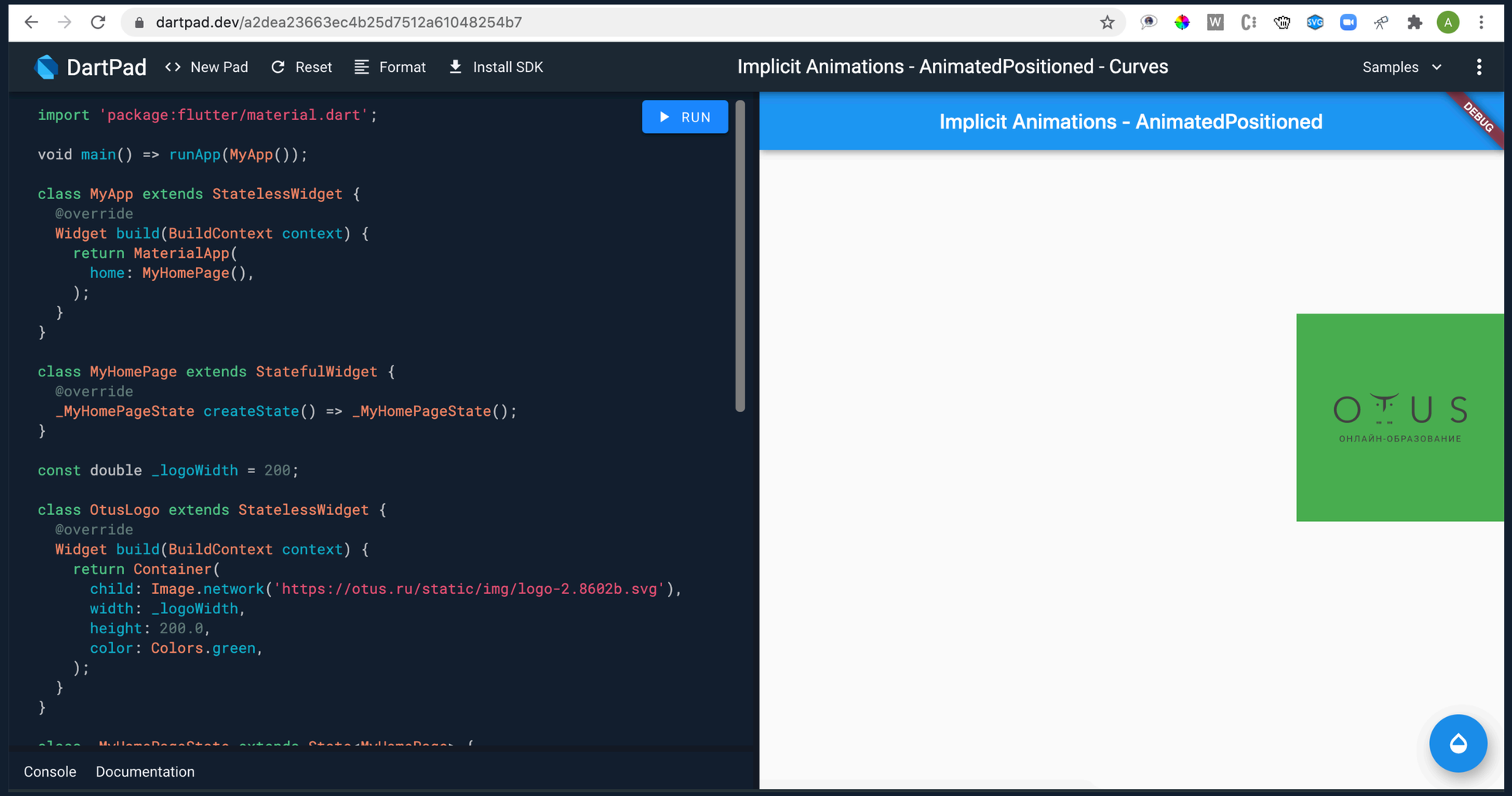


http://bit.ly/AnimatedPositioned

ImplicitlyAnimatedWidget

```
abstract class ImplicitlyAnimatedWidget extends
StatefulWidget {

  /// Initializes fields for subclasses.
  /// The [curve] and [duration] arguments must not be null.
  const ImplicitlyAnimatedWidget({
    Key key,
    this.curve = Curves.linear,
    @required this.duration,
    this.onEnd,
  })

  /// The curve to apply when animating the parameters of this container.
  final Curve curve;

  /// The duration over which to animate the parameters of this container.
  final Duration duration;

  /// Called every time an animation completes.
  /// This can be useful to trigger additional actions (e.g. another animation)
  /// at the end of the current animation.
  final VoidCallback onEnd;
```

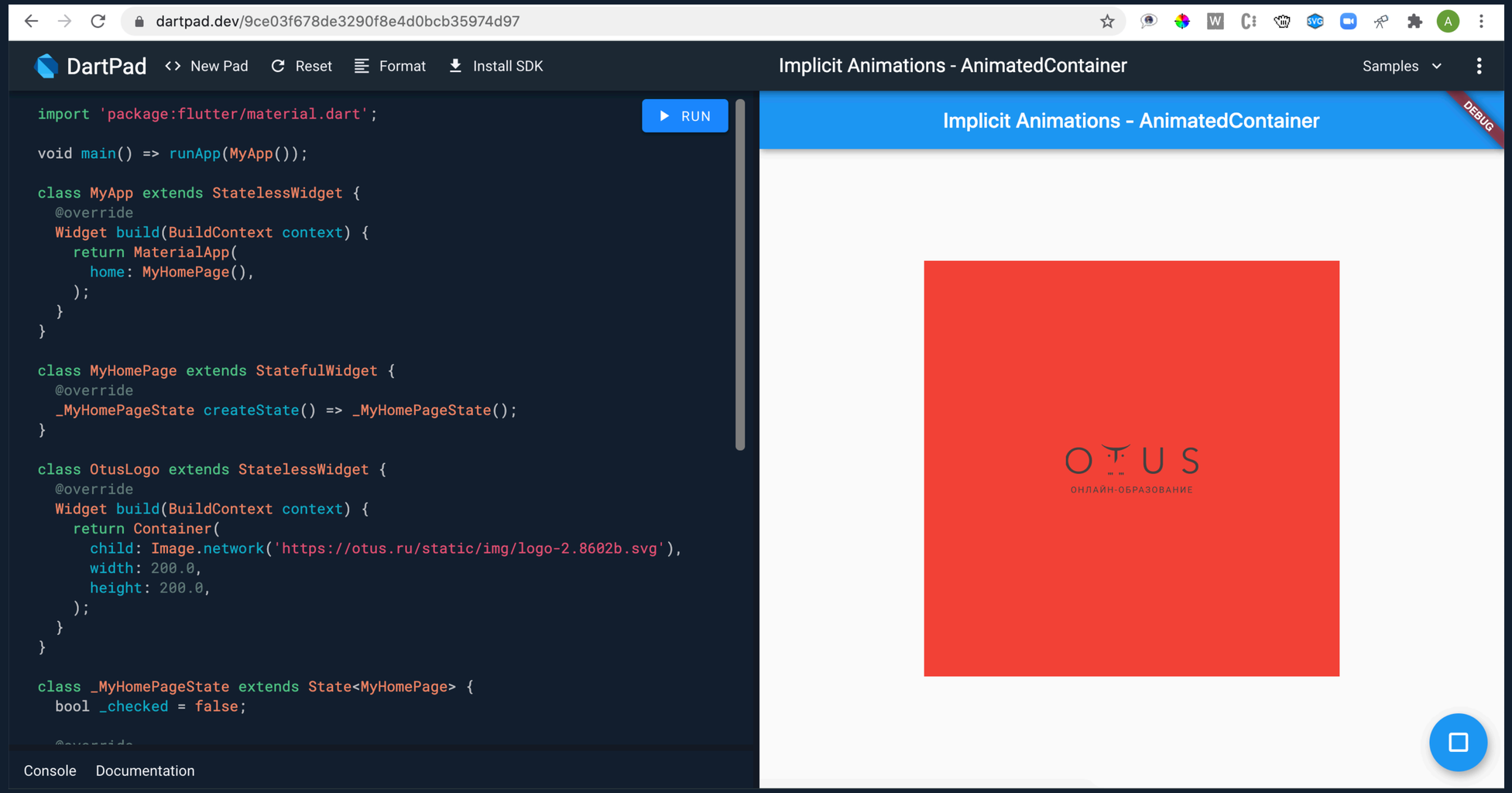
Implicit Animations - AnimatedPositioned - Curves



http://bit.ly/AnimatedPositionedCurves

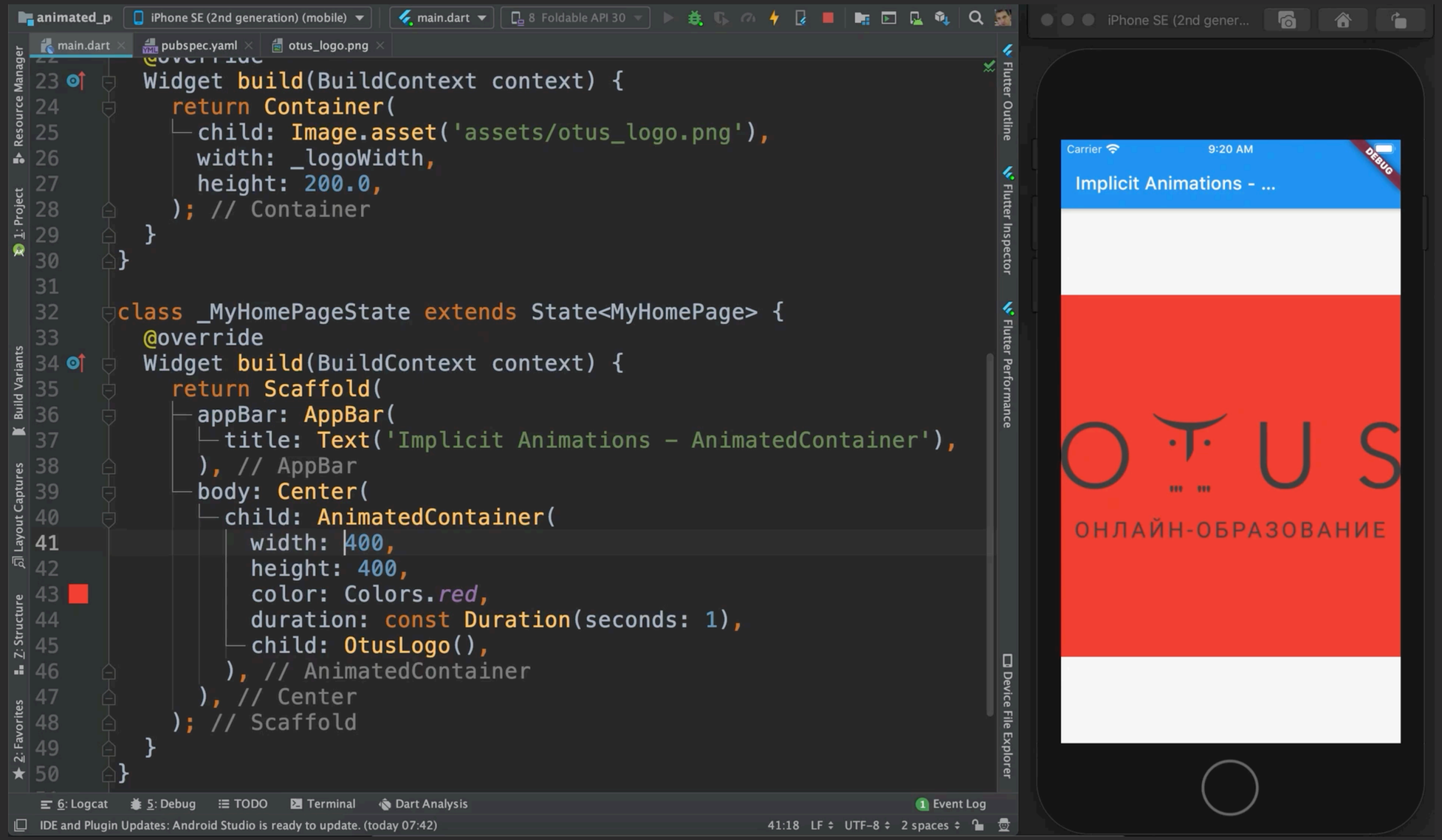
Curves - о них чуть позже

Implicit Animations - AnimatedContainer



http://bit.ly/AnimatedContainer

Implicit Animations - AnimatedContainer



Идем дальше

Что такое анимация и как она устроена

Что есть для анимации в Flutter Framework

Implicit Animation Widgets & AnimatedContainer

Tween & TweenAnimationBuilder

Animation physics. Curves.

Lerp & Tween

А если я не нахожу Animated<Foo> widget?

Animated<Color? Some Behaviour?> widget

Lerp & Tween - пример на lerp - /// todo

http://bit.ly/lerp-interpolation

Interpolation

Have you ever wondered how computer graphics animators draw each and every frame of your favorite CGI movies?

Well, they don't! Instead, animators set an initial position and a final position for the object they are animating.

Next, they rely on software to compute all of the positions for the object between the initial and the final position that they defined.

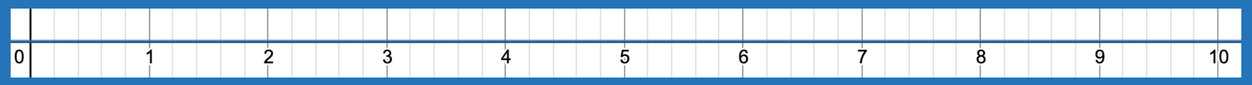
The process of computing animation values between a starting and ending position is called interpolation.

Lerp & Tween

Tween<T extends dynamic> class

A linear interpolation between a beginning and ending value.

Tween is useful if you want to interpolate across a range.
To use a Tween object with an animation, call the Tween object's animate method and pass it the Animation object that you want to modify.



IntTween(begin: 0, end: 10)

Lerp & Tween - Int Tween - Demo

DartPad New Pad Reset Format Install SDK Implicit Animations - IntTween

```dart
import 'package:flutter/animation.dart';

void main() {

  IntTween tween = IntTween(begin: 0,  end: 10);

  final duration = Duration(seconds: 1);
  final step = Duration.millisecondsPerSecond / 60;
  print(duration.inMilliseconds);
  for(var currentTime = 0.0; currentTime < duration.inMilliseconds; currentTime += step) {
    final value = tween.lerp(currentTime / duration.inMilliseconds);
    print('current time is $currentTime, tween is $value');
```

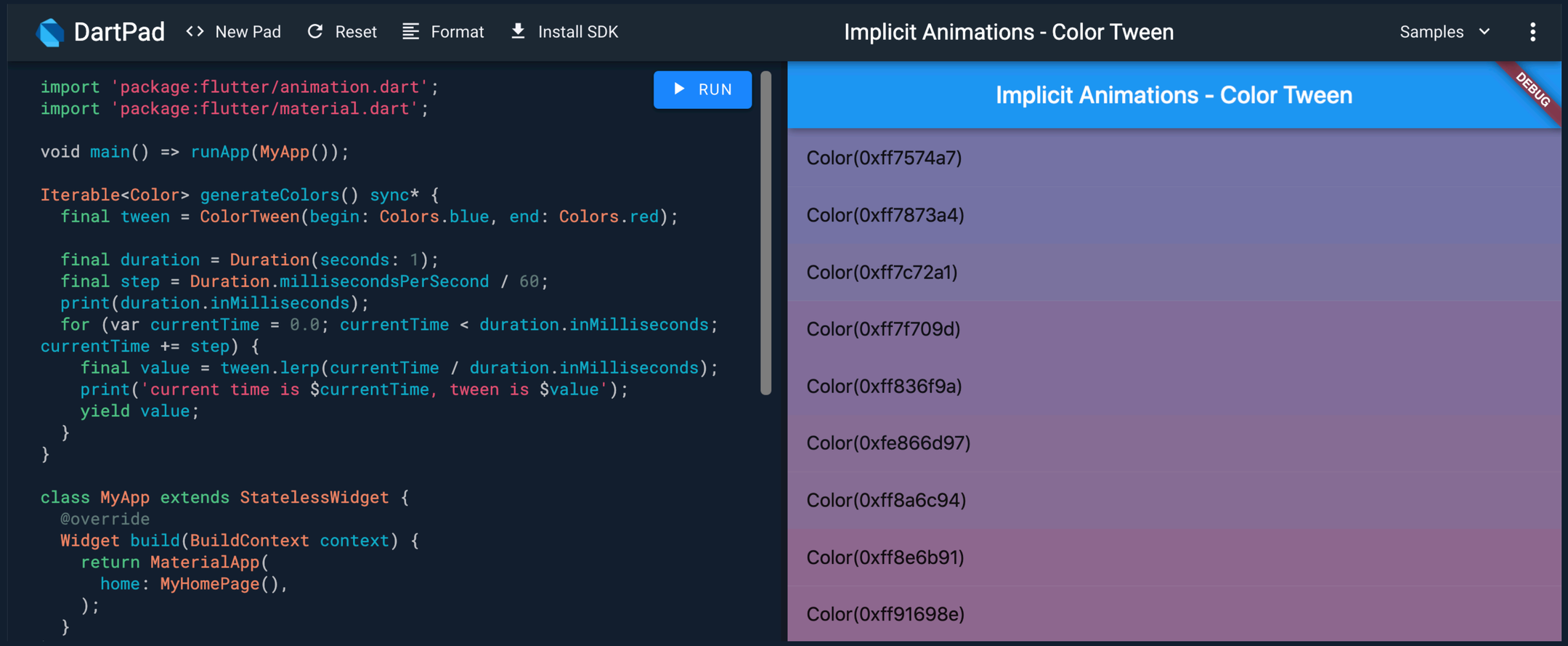
RUN

Console Documentation

```
1000
current time is 0, tween is 0
current time is 16.666666666666668, tween is 0
current time is 33.333333333333336, tween is 0
current time is 50, tween is 1
current time is 66.66666666666667, tween is 1
current time is 83.33333333333334, tween is 1
current time is 100.00000000000001, tween is 1
current time is 116.66666666666669, tween is 1
current time is 133.33333333333334, tween is 1
current time is 150, tween is 2
current time is 166.66666666666666, tween is 2
```

http://dartpad.dev/6145199454a6c612f5a9f577c281bded

Lerp & Tween - Color Tween - Demo



https://dartpad.dev/98a0a67063faf645b5ee536d84c12ab6

Tween

```dart
class Tween<T extends dynamic> extends Animatable<T> {
  Tween({ this.begin, this.end });

  /// The value this variable has at the beginning of the animation.
  /// See the constructor for details about whether this property may be null
  /// (it varies from subclass to subclass).
  T begin;

  /// The value this variable has at the end of the animation.
  /// See the constructor for details about whether this property may be null
  /// (it varies from subclass to subclass).
  T end;

  /// Returns the value this variable has at the given animation clock value.
  /// The default implementation of this method uses the [+], [-], and [*]
  /// operators on `T`. The [begin] and [end] properties must therefore be
  /// non-null by the time this method is called.
  @protected
  T lerp(double t) => begin + (end - begin) * t as T;
```

Tween

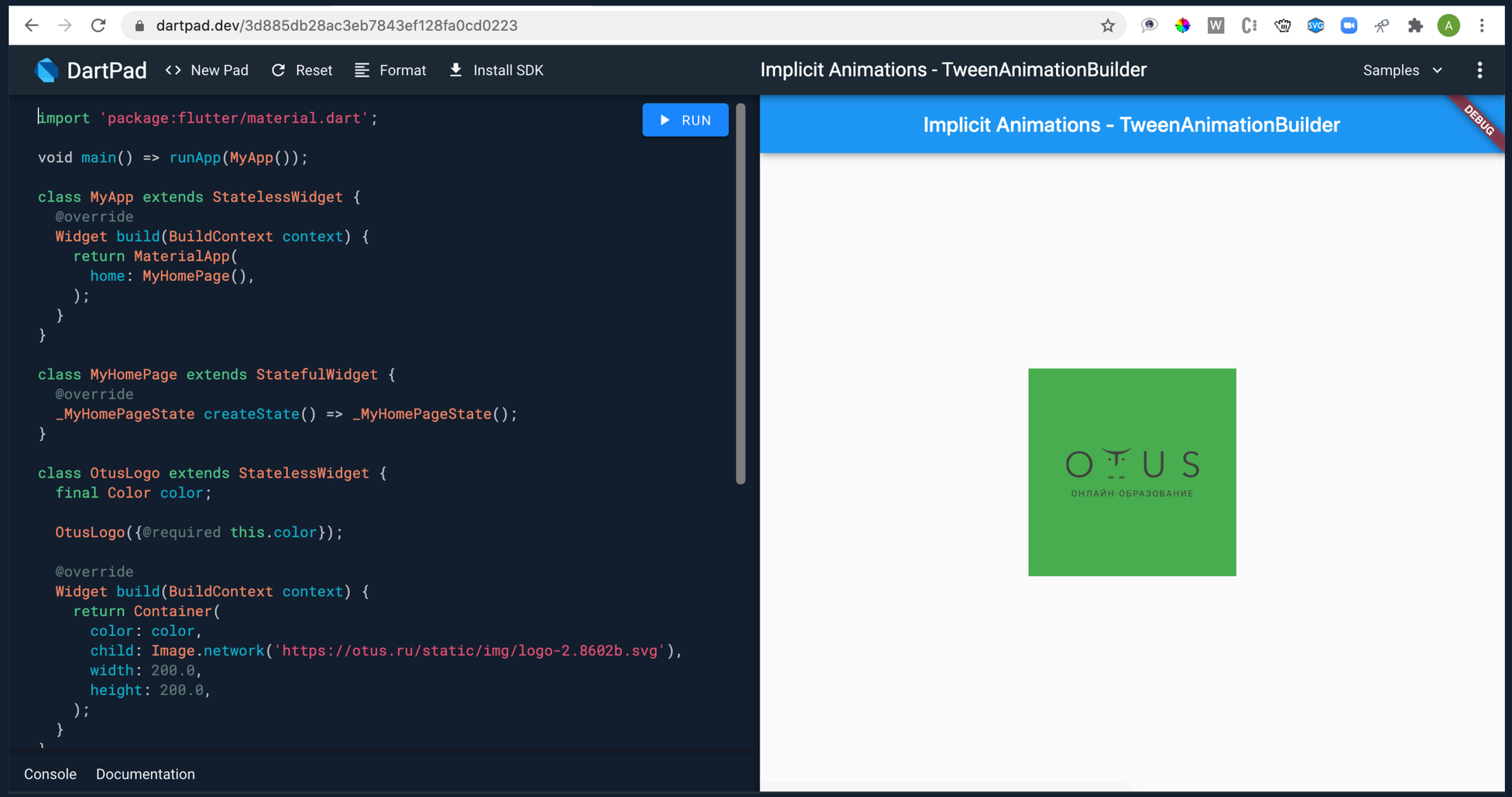
Tween Implementers

- AlignmentGeometryTween
- AlignmentTween
- BorderRadiusTween
- BorderTween
- BoxConstraintsTween
- ColorTween
- ConstantTween
- DecorationTween
- EdgeInsetsGeometryTween
- EdgeInsetsTween
- FractionalOffsetTween
- IntTween
- MaterialPointArcTween
- Matrix4Tween
- RectTween
- RelativeRectTween
- ReverseTween
- ShapeBorderTween
- SizeTween
- StepTween
- TextStyleTween

TweenAnimationBuilder

```
class TweenAnimationBuilder<T> extends
                                ImplicitlyAnimatedWidget {

  const TweenAnimationBuilder({
    Key key,
    @required this.tween,
    @required Duration duration,
    Curve curve = Curves.linear,
    @required this.builder,
    VoidCallback onEnd,
    this.child,
  })

  /// Defines the target value for the animation.
  final Tween<T> tween;

  /// Called every time the animation value changes.
  /// The current animation value is passed to the builder along with the
  /// [child]. The builder should build a [Widget] based on the current
  /// animation value and incorporate the [child] into it, if it is non-null.
  final ValueWidgetBuilder<T> builder;

  /// The child widget to pass to the builder.
  final Widget child;
```

Implicit Animations - TweenAnimationBuilder



http://bit.ly/TweenAnimationBuilder

Implicit Animations - TweenSequence - на следующем уроке

```dart
final Animation<double> animation = TweenSequence(
  <TweenSequenceItem<double>>[
    TweenSequenceItem<double>(
      tween: Tween<double>(begin: 5.0, end: 10.0)
        .chain(CurveTween(curve: Curves.ease)),
      weight: 40.0,
    ), // TweenSequenceItem
    TweenSequenceItem<double>(
      tween: ConstantTween<double>(10.0),
      weight: 20.0,
    ), // TweenSequenceItem
    TweenSequenceItem<double>(
      tween: Tween<double>(begin: 10.0, end: 5.0)
        .chain(CurveTween(curve: Curves.ease)),
      weight: 40.0,
    ), // TweenSequenceItem
  ], // <TweenSequenceItem<double>>[]
).animate(myAnimationController); // TweenSequence
```

ImplicitlyAnimatedWidget

```
abstract class ImplicitlyAnimatedWidget extends StatefulWidget {

  /// Initializes fields for subclasses.
  /// The [curve] and [duration] arguments must not be null.
  const ImplicitlyAnimatedWidget({
    Key key,
    this.curve = Curves.linear,
    @required this.duration,
    this.onEnd,
  })

  /// The curve to apply when animating the parameters of this container.
  final Curve curve;

  /// The duration over which to animate the parameters of this container.
  final Duration duration;

  /// Called every time an animation completes.
  /// This can be useful to trigger additional actions (e.g. another animation)
  /// at the end of the current animation.
  final VoidCallback onEnd;
```

Implementers - Common implicitly animated widgets

- AnimatedAlign, which is an implicitly animated version of Align.
- AnimatedContainer, which is an implicitly animated version of Container.
- AnimatedDefaultTextStyle, which is an implicitly animated version of DefaultTextStyle.
- AnimatedOpacity, which is an implicitly animated version of Opacity.
- AnimatedPadding, which is an implicitly animated version of Padding.
- AnimatedPhysicalModel, which is an implicitly animated version of PhysicalModel.
- AnimatedPositioned, which is an implicitly animated version of Positioned.
- AnimatedPositionedDirectional, which is an implicitly animated version of PositionedDirectional.
- AnimatedTheme, which is an implicitly animated version of Theme.
- AnimatedCrossFade, which cross-fades between two given children and animates itself between their sizes.
- AnimatedSize, which automatically transitions its size over a given duration.
- AnimatedSwitcher, which fades from one widget to another.

- **TweenAnimationBuilder**, which animates any property expressed by a Tween to a specified target value.

Implicit animation widgets

Если требуется анимация одного или нескольких свойств, то

`Animated<FooWidget>`

Implicit animation widgets

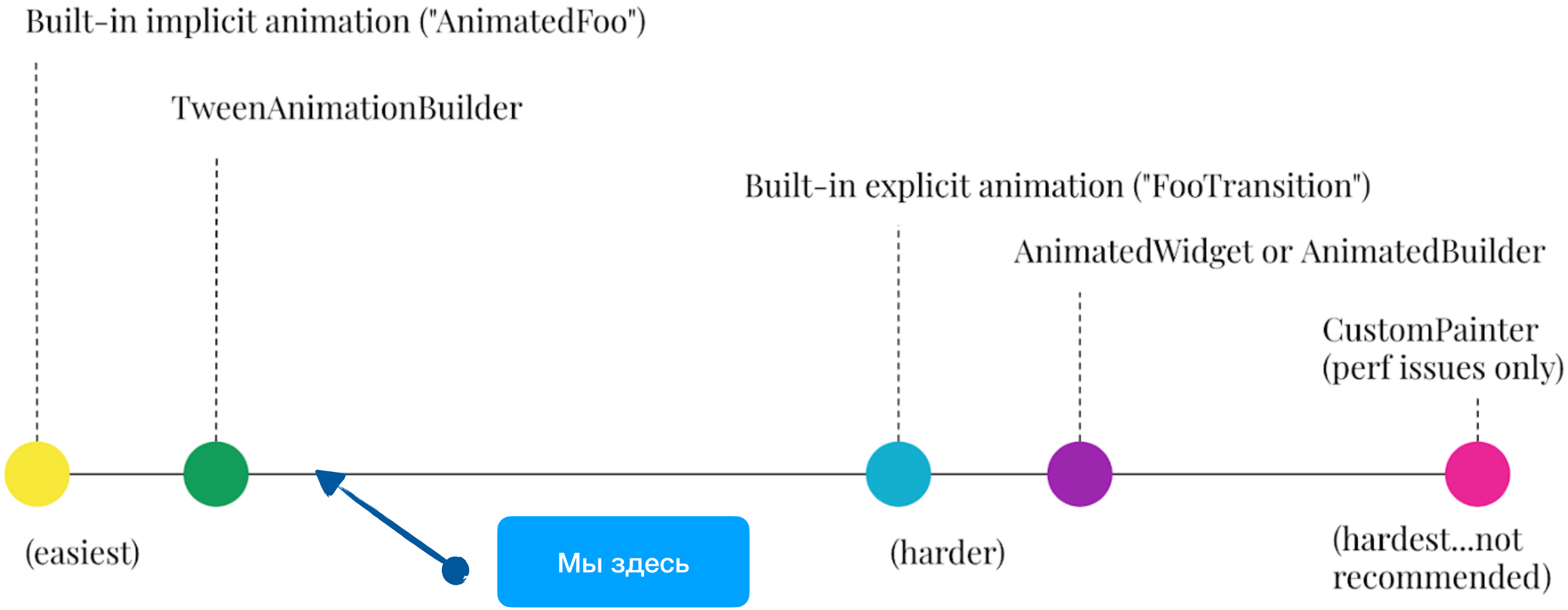
Если требуется анимация одного или нескольких свойств, то

Animated<FooWidget>

Если требуется анимация на основе Tween или своего custom Tween, то

TweenAnimationBuilder

Implicit animation widgets



Идем дальше

Что такое анимация и как она устроена

Что есть для анимации в Flutter Framework

Implicit Animation Widgets & AnimatedContainer

Tween & TweenAnimationBuilder

Animation physics. Curves.

Animation physics

Animation physics. Curves.

Animation physics

```
AnimatedPositioned(
  duration: const Duration(seconds: 1),
  curve: Curves.bounceOut, //  <- curve to set physics behaviour
  left: _left,
  child: FlutterIcon(),
),
```

http://bit.ly/AnimatedPositionedCurves

Animation physics

```dart
/// An parametric animation easing curve, i.e. a mapping of the unit interval to
/// the unit interval.
///
/// Easing curves are used to adjust the rate of change of an animation over
/// time, allowing them to speed up and slow down, rather than moving at a
/// constant rate.
///
/// A [Curve] must map t=0.0 to 0.0 and t=1.0 to 1.0.
///
/// See also:
///
///  * [Curves], a collection of common animation easing curves.
///  * [CurveTween], which can be used to apply a [Curve] to an [Animation].
///  * [Canvas.drawArc], which draws an arc, and has nothing to do with easing
///    curves.
///  * [Animatable], for a more flexible interface that maps fractions to
///    arbitrary values.
@immutable
abstract class Curve extends ParametricCurve<double> {
  /// Abstract const constructor to enable subclasses to provide
  /// const constructors so that they can be used in const expressions.
  const Curve();
```

Animation physics

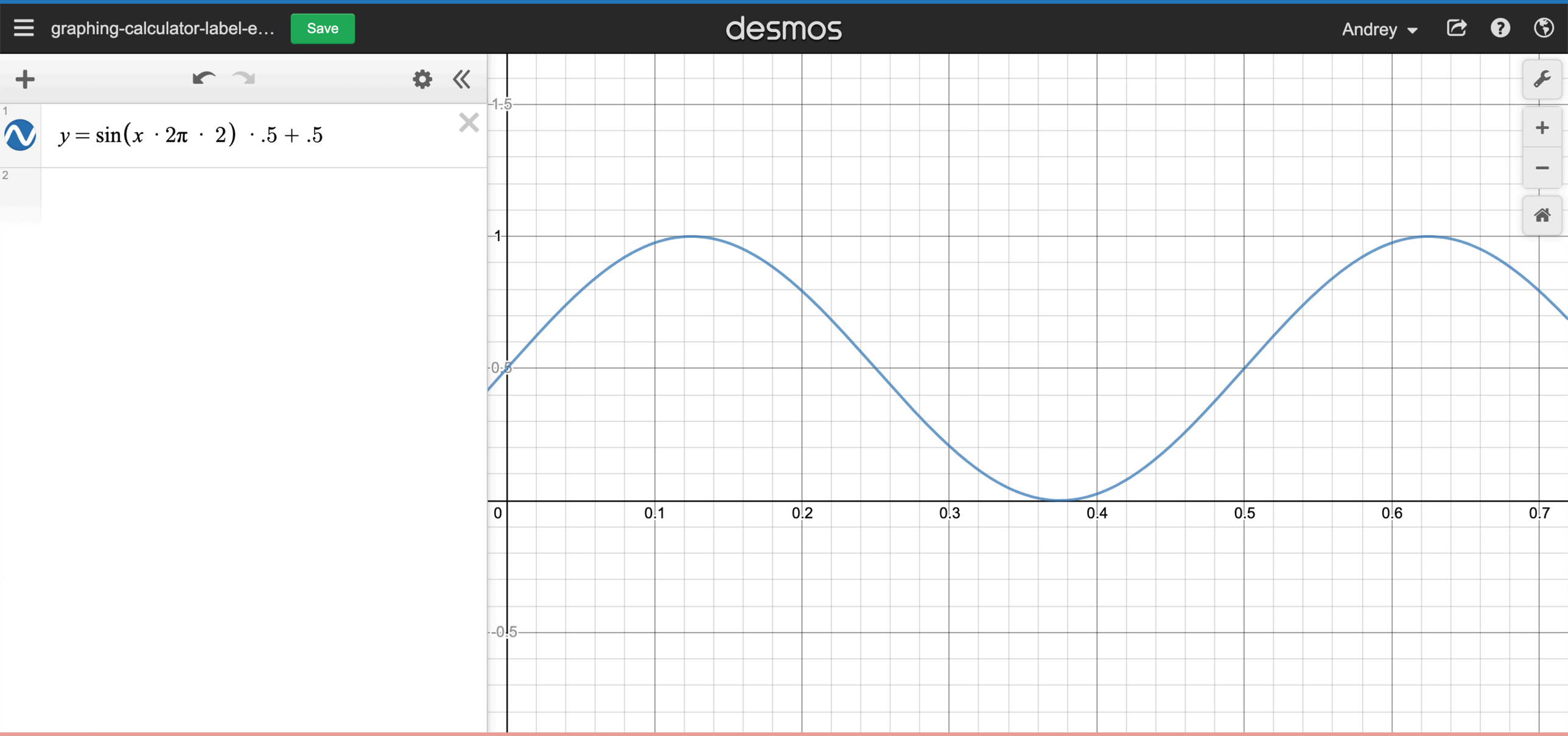
```dart
/// Returns the value of the curve at point `t`.
///
/// This function must ensure the following:
/// - The value of `t` must be between 0.0 and 1.0
/// - Values of `t`=0.0 and `t`=1.0 must be mapped to 0.0 and 1.0,
/// respectively.
///
/// It is recommended that subclasses override [transformInternal] instead of
/// this function, as the above cases are already handled in the default
/// implementation of [transform], which delegates the remaining logic to
/// [transformInternal].
@override
double transform(double t) {
  if (t == 0.0 || t == 1.0) {
    return t;
  }
  return super.transform(t);
```

Animation physics - Custom Physics - Demo

$$y = \sin(x \cdot 2\pi)$$

Custom Curves

https://www.desmos.com/calculator/stm0cm6q8t



Custom Curves - Реализуем свой алгоритм

```
class CustomCurve extends Curve {
  /// Returns the value of the curve at point `t`.
  ///
  /// The given parametric value `t` will be between 0.0 and 1.0, inclusive.

  @override
  double transform(double t) {
    ///
    /// y=sin(x * 2ρ * 2) * .5 + .5
    ///
    final y = math.sin(t * 5 * math.pi * 2) * .5 + .5;
    return y;
  }
}
```

Animation physics

```
// BOUNCE CURVES

double _bounce(double t) {
  if (t < 1.0 / 2.75) {
    return 7.5625 * t * t;
  } else if (t < 2 / 2.75) {
    t -= 1.5 / 2.75;
    return 7.5625 * t * t + 0.75;
  } else if (t < 2.5 / 2.75) {
    t -= 2.25 / 2.75;
    return 7.5625 * t * t + 0.9375;
  }
  t -= 2.625 / 2.75;
  return 7.5625 * t * t + 0.984375;
}
```

Animation physics

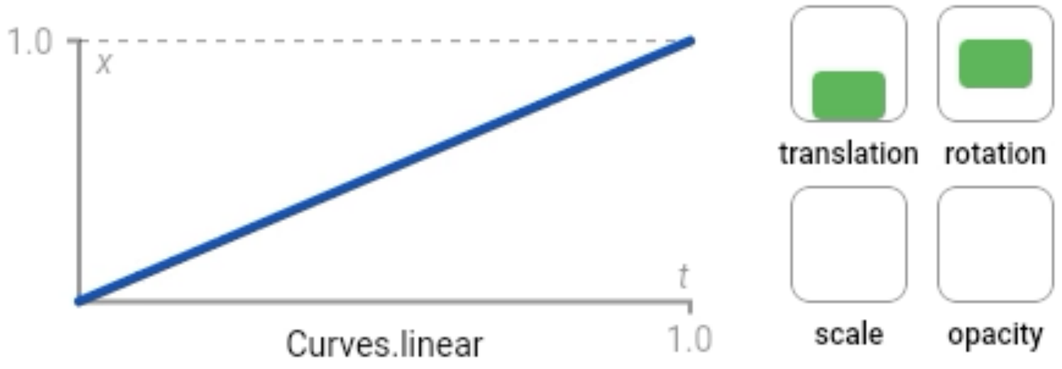
```
_animation = CurvedAnimation(
  curve: Curves.bounceOut,
  parent: _ar
);
```

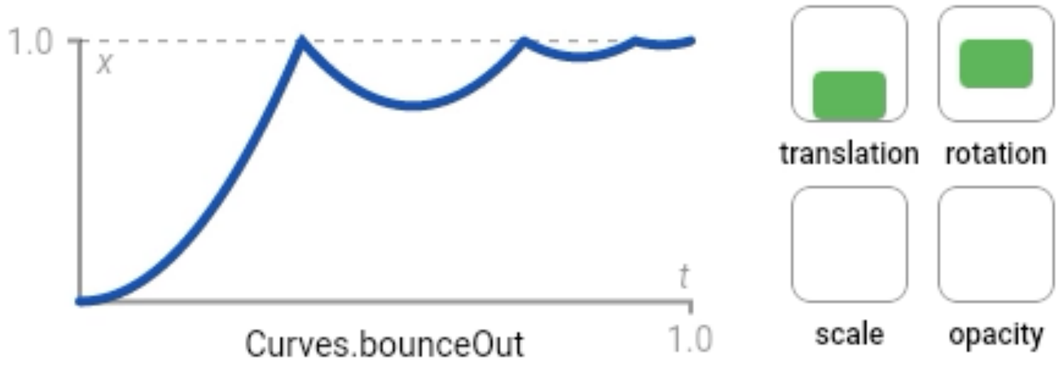
bounceOut Curve
decelerate Curve
bounceIn Curve
bounceInOut Curve
ease Cubic
easeIn Cubic
easeInBack Cubic
easeInCirc Cubic
easeInCubic Cubic
easeInExpo Cubic
easeInOut Cubic

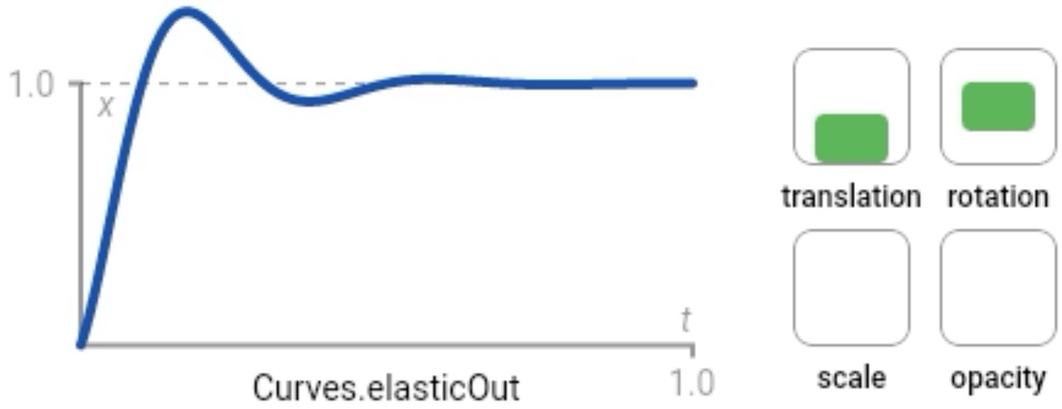
Use ⇥ to overwrite the current identifier with the chosen variant π

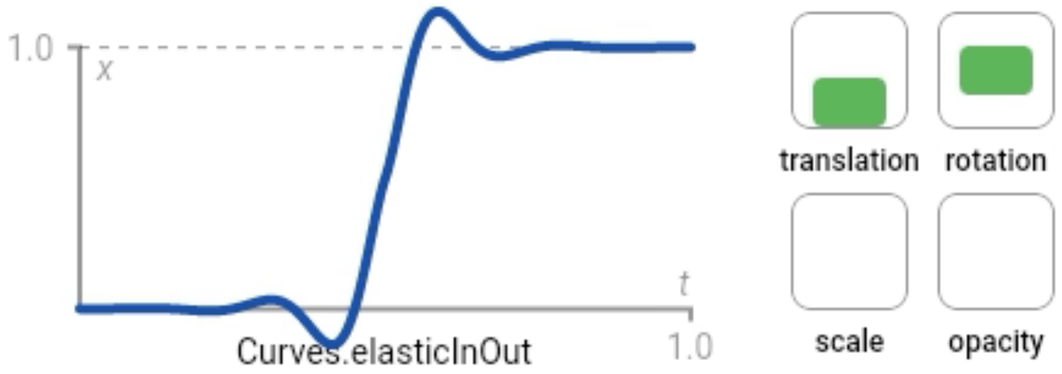
Animation physics

http://bit.ly/flutter-curves









Сегодня мы рассмотрели:

- Что такое анимация и как она устроена
- Что есть для анимации в Flutter Framework
- Implicit Animation Widgets & AnimatedContainer
- Tween & TweenAnimationBuilder
- Animation physics. Curves.

Заполните, пожалуйста,
опрос о занятии по ссылке в чате

Спасибо за внимание!
Приходите на следующие вебинары

Смирнов Андрей wrike

Курс **Мобильная разработка на Flutter**