ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо видно && слышно?

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

Flutter Mobile Developer

Тема 11. Flutter. Explicit animations

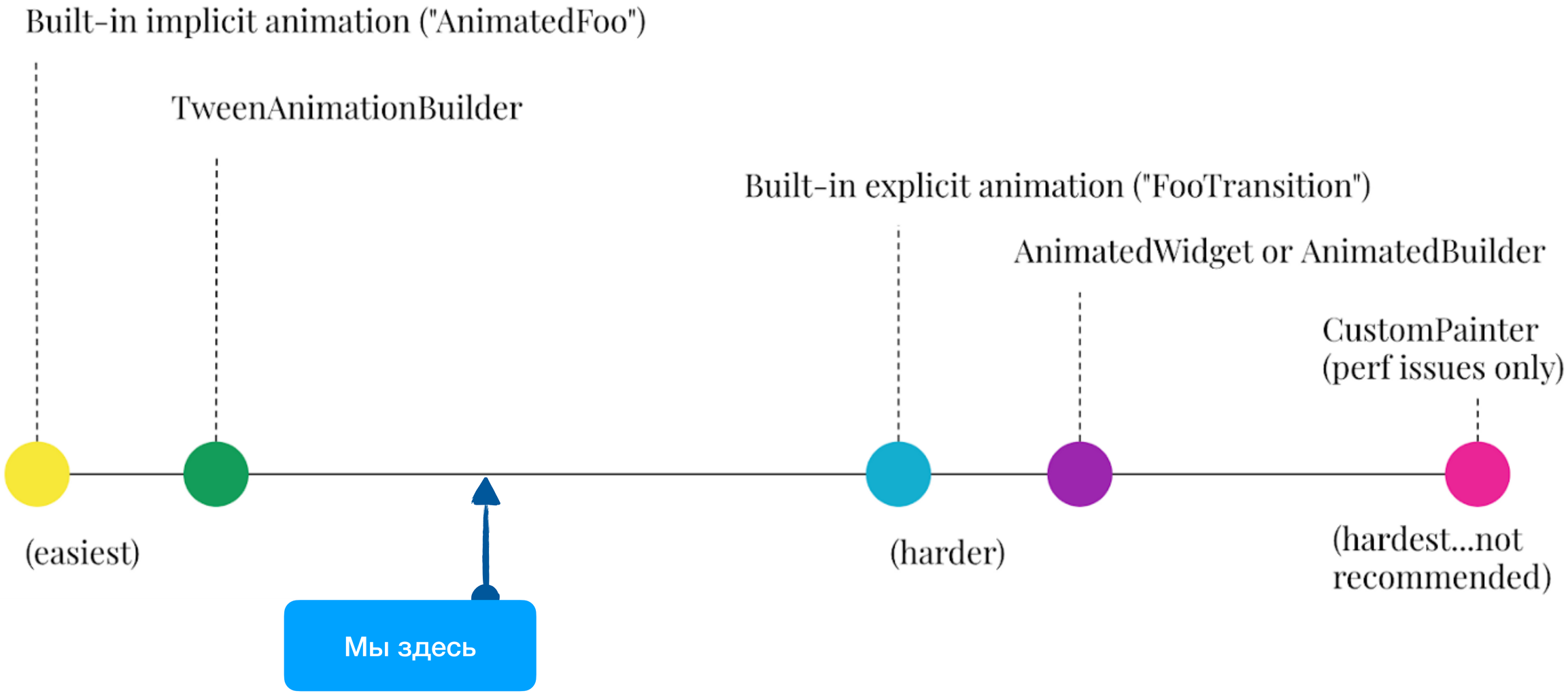
Цель урока

- делать анимации с помощью AnimatedBuilder и AnimatedWidget

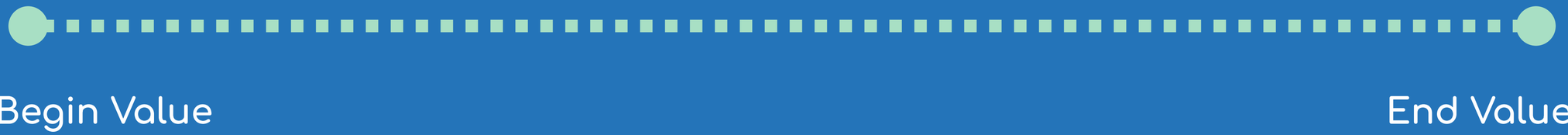
О чем будем говорить

- Implicit animation widgets - вспомним
- Animation Controller
- Explicit animation widgets
- Curves
- Tweens

Implicit animation widgets

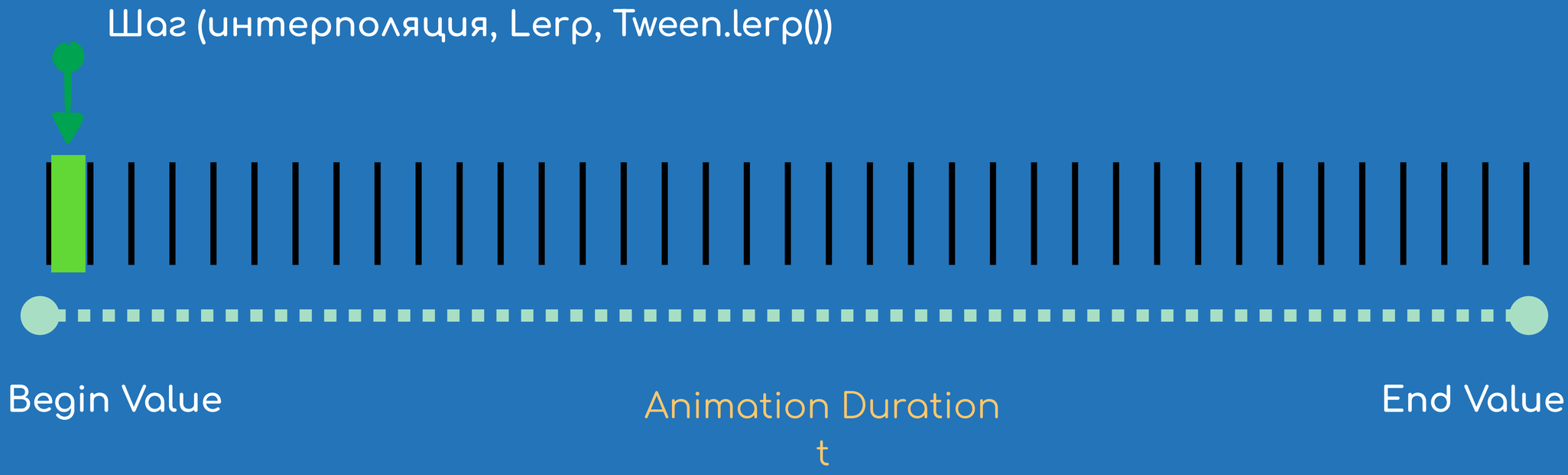


Implicit animation widgets

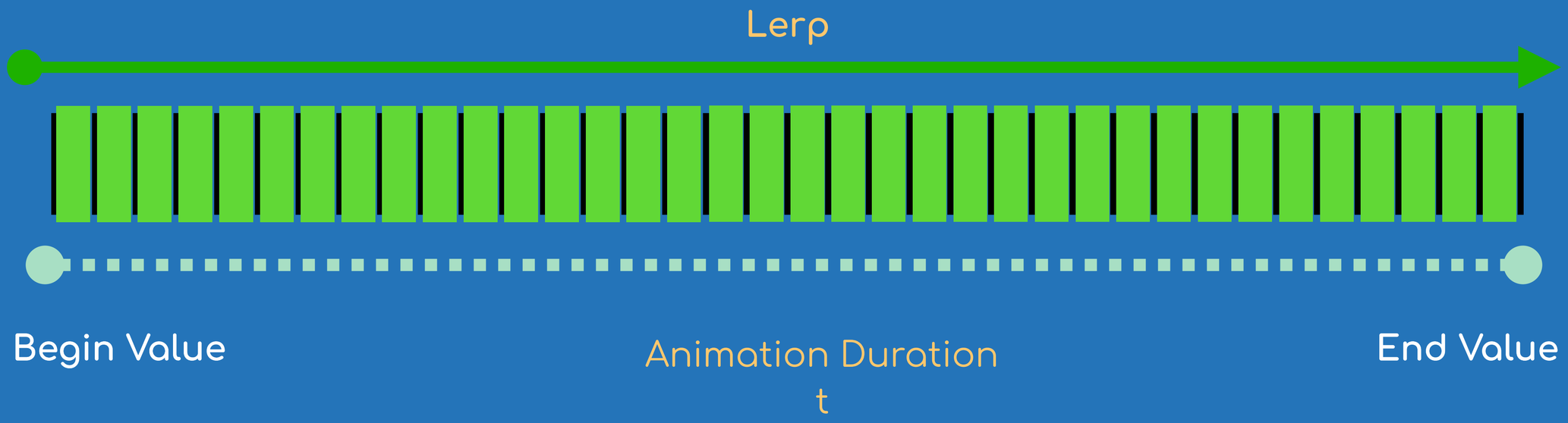


Implicit animation widgets

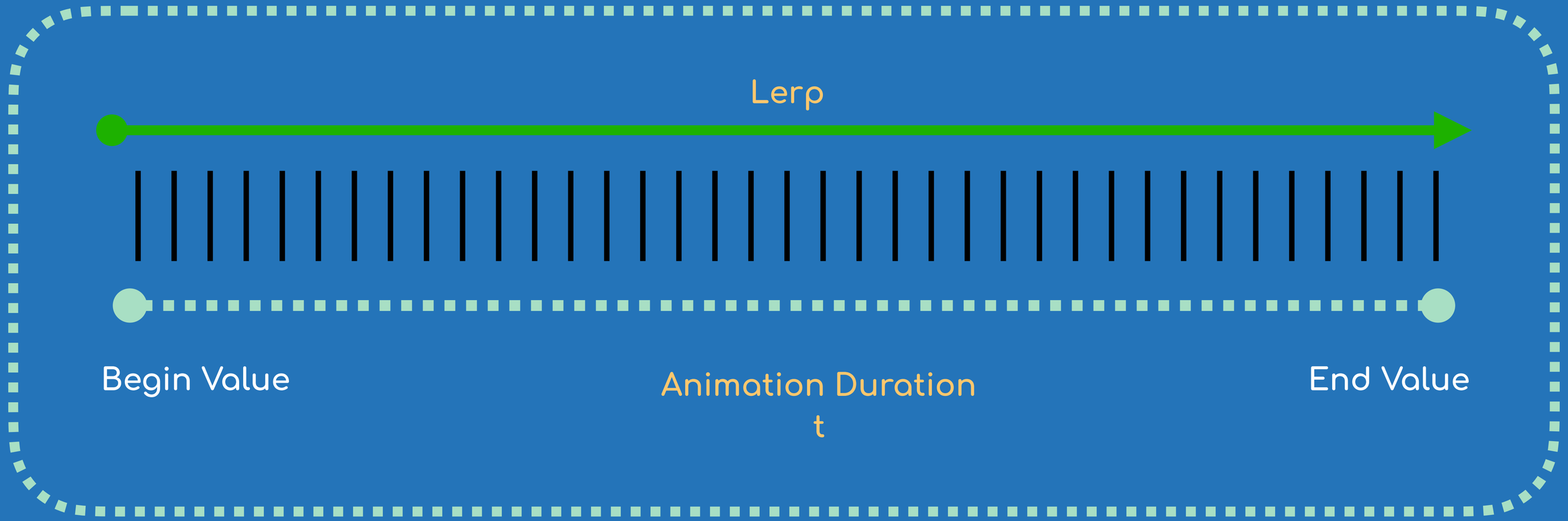
Implicit animation widgets



Implicit animation widgets

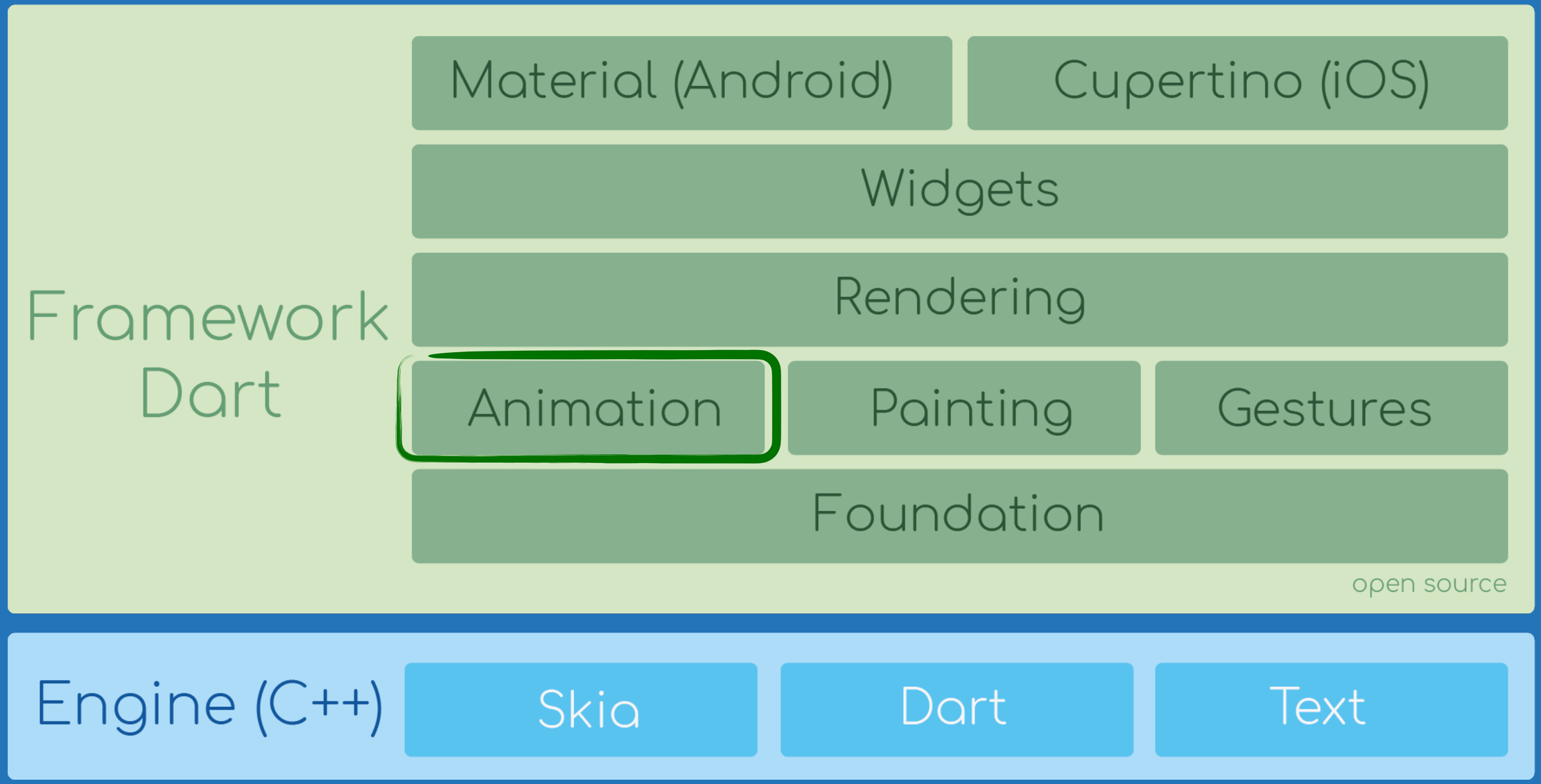


Implicit animation widgets



Нет возможности повлиять на анимацию во время ее выполнения

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.

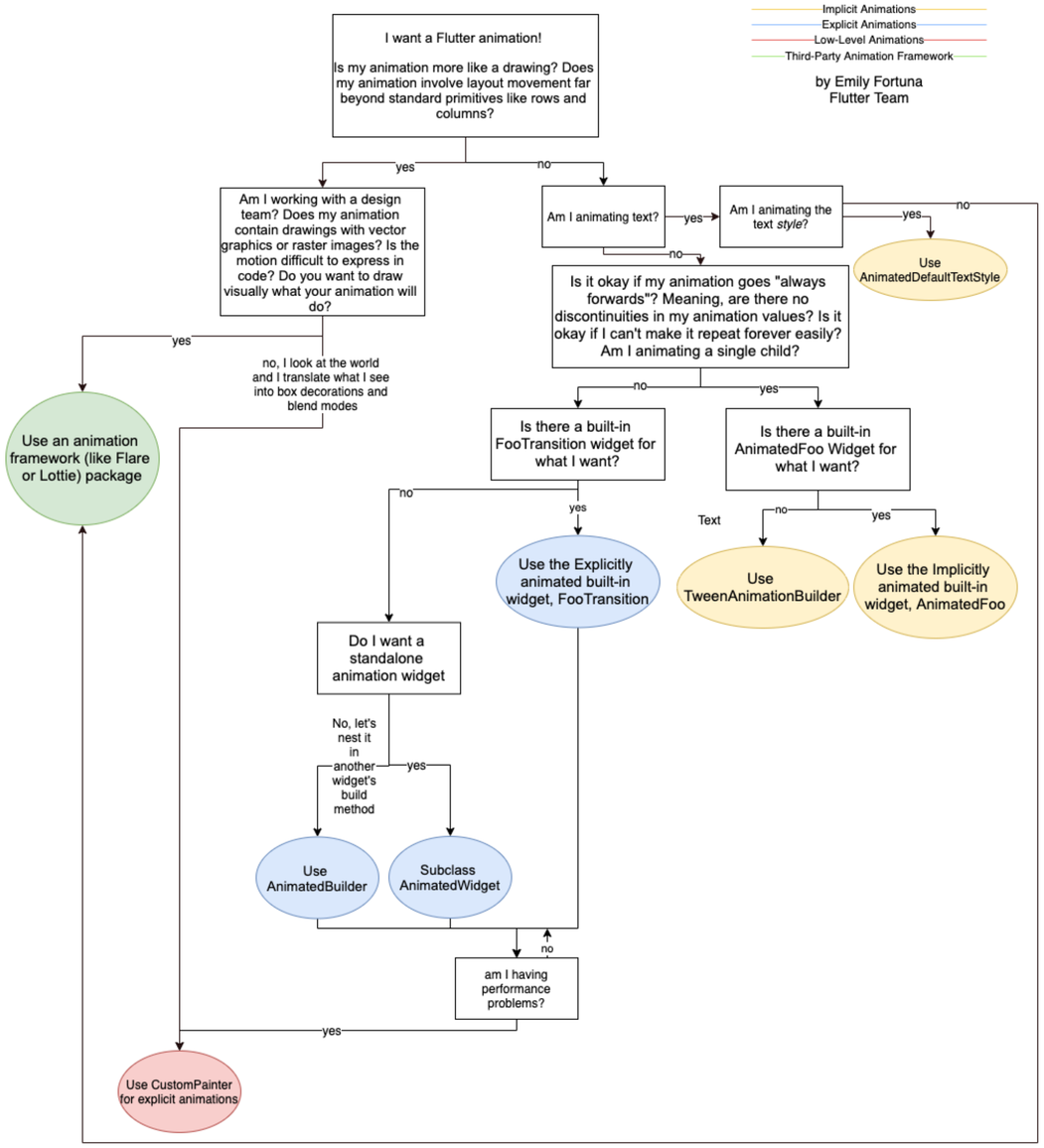
For example, the widget library includes ImplicitlyAnimatedWidgets and AnimatedWidgets that make it easy to animate certain properties of a Widget.

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.

Принятие решения

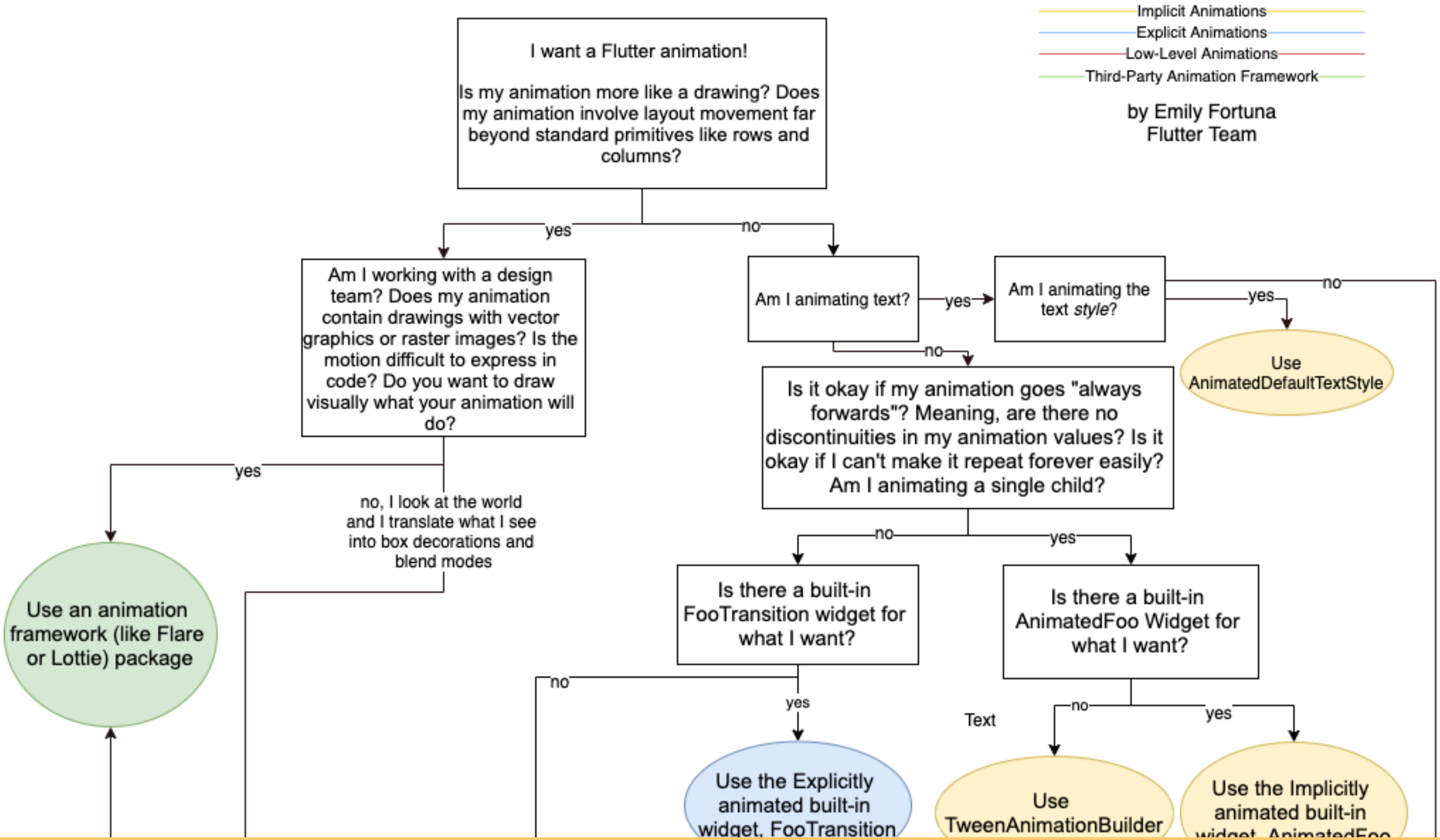
https://flutter.dev/docs/development/ui/animations



Implicit animation widgets - вспомним

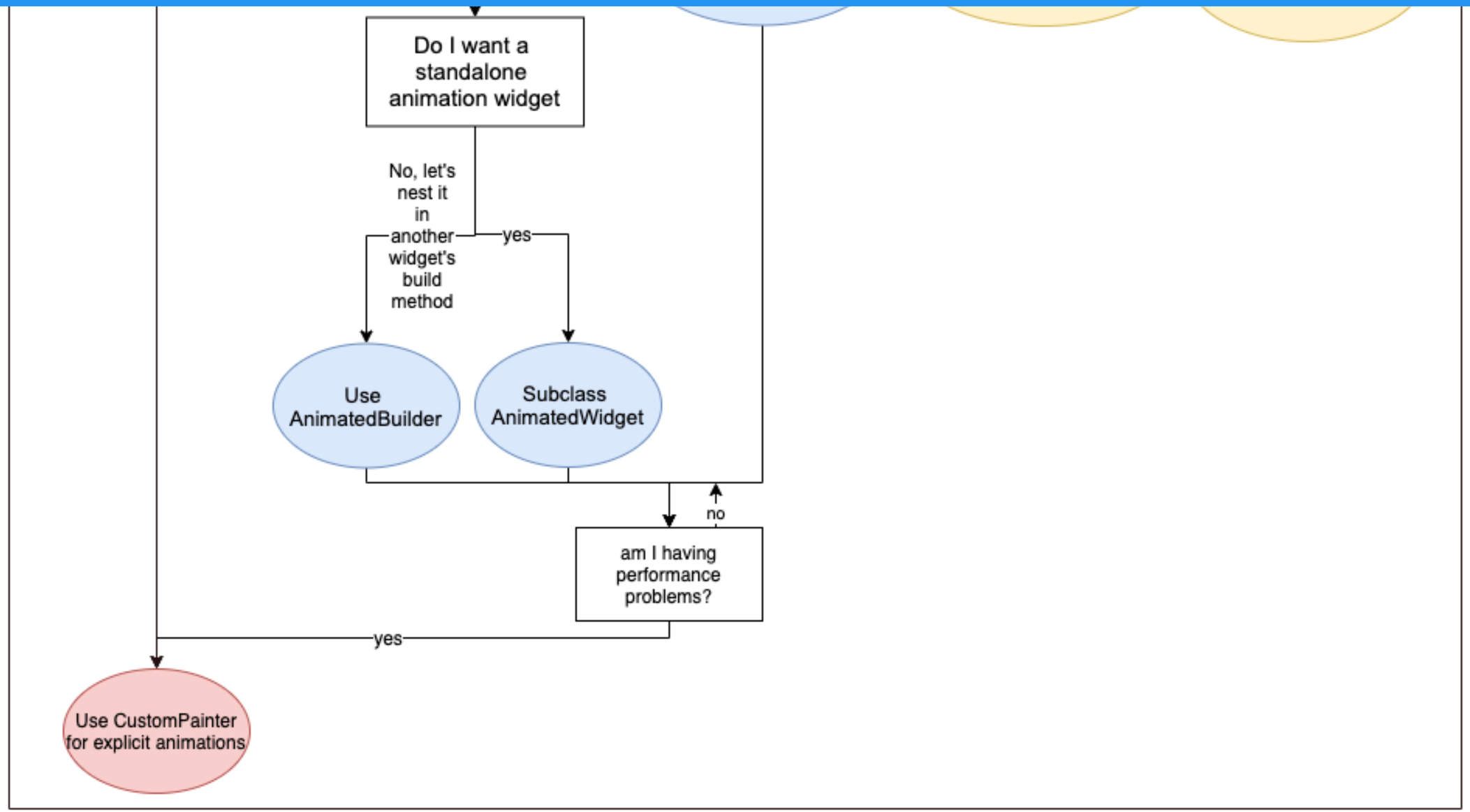
Принятие решения

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



Принятие решения

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



Принятие решения

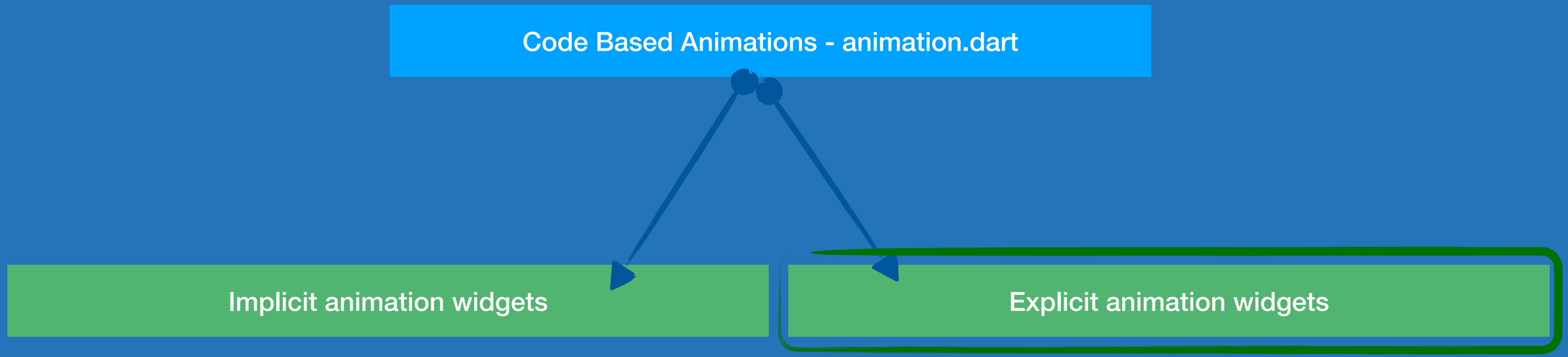
Первый вопрос:

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) - Третий вебинар в модуле 2 “Анимация”

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

Code Based Animations - animation.dart



What are explicit animations?

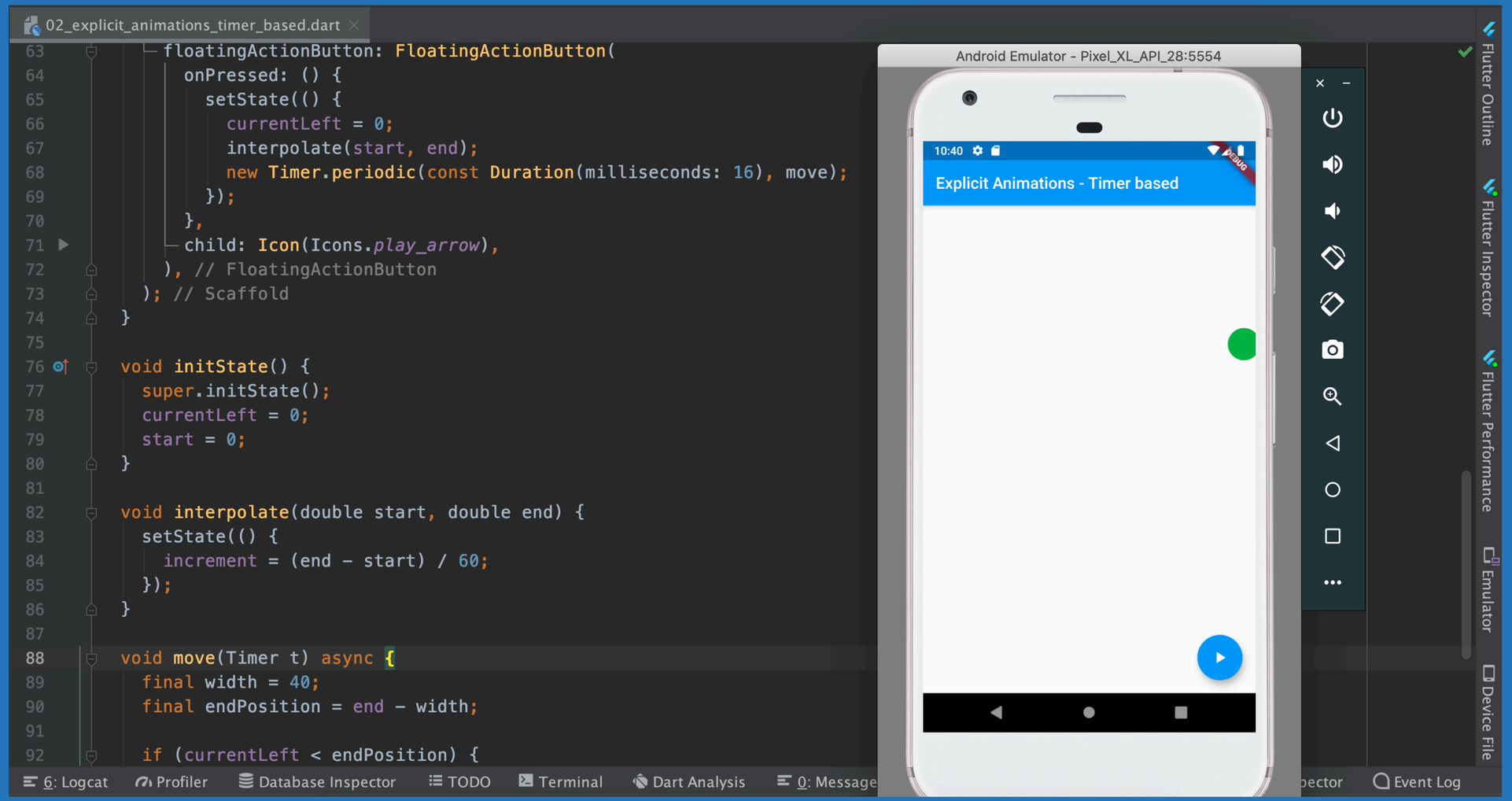
What are explicit animations?

Explicit animations are a set of controls for telling Flutter how to rapidly rebuild the widget tree while changing widget properties to create animation effects.

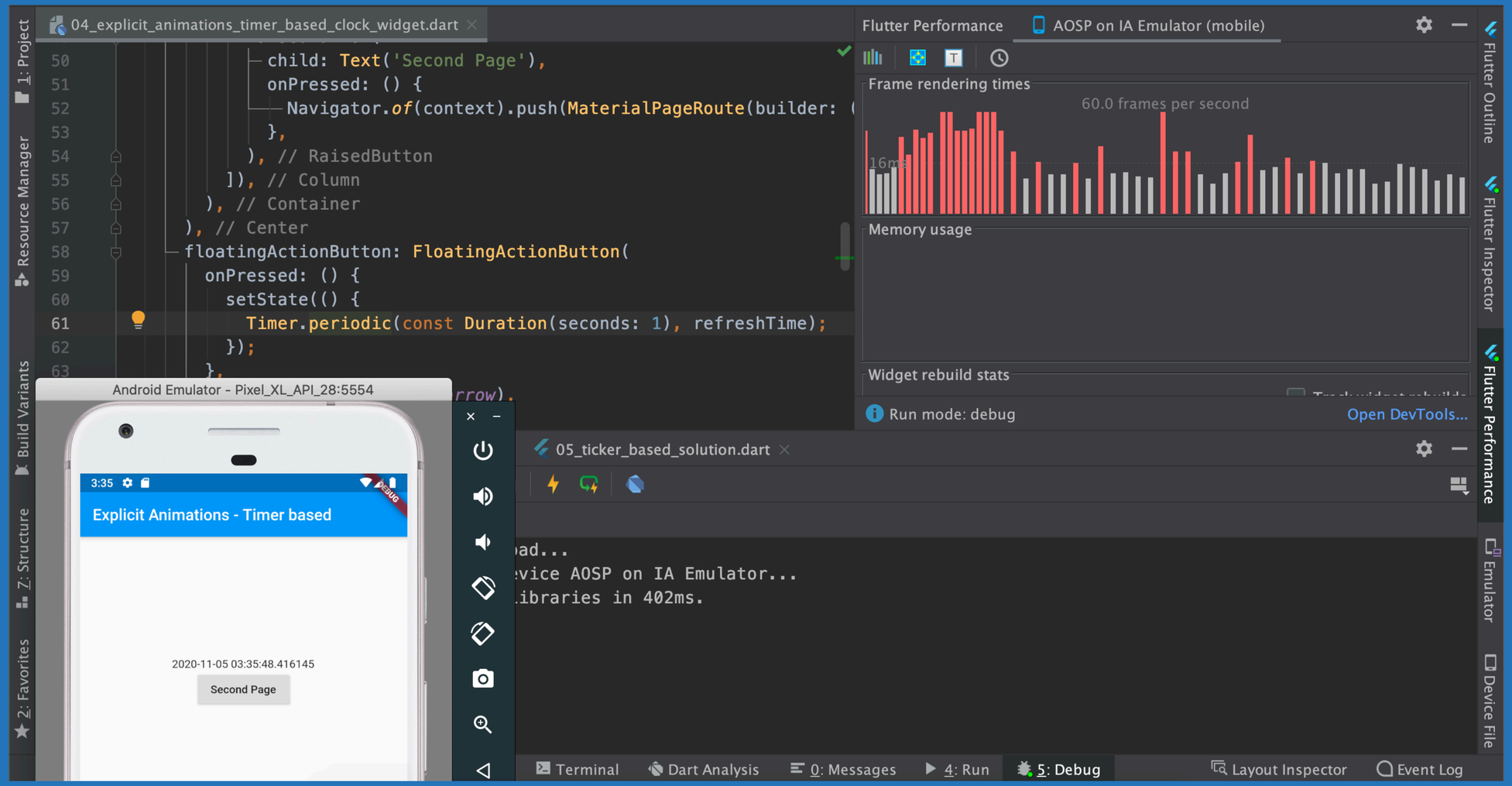
This approach enables you to create effects that you can't achieve using implicit animations.

Timer Based solution

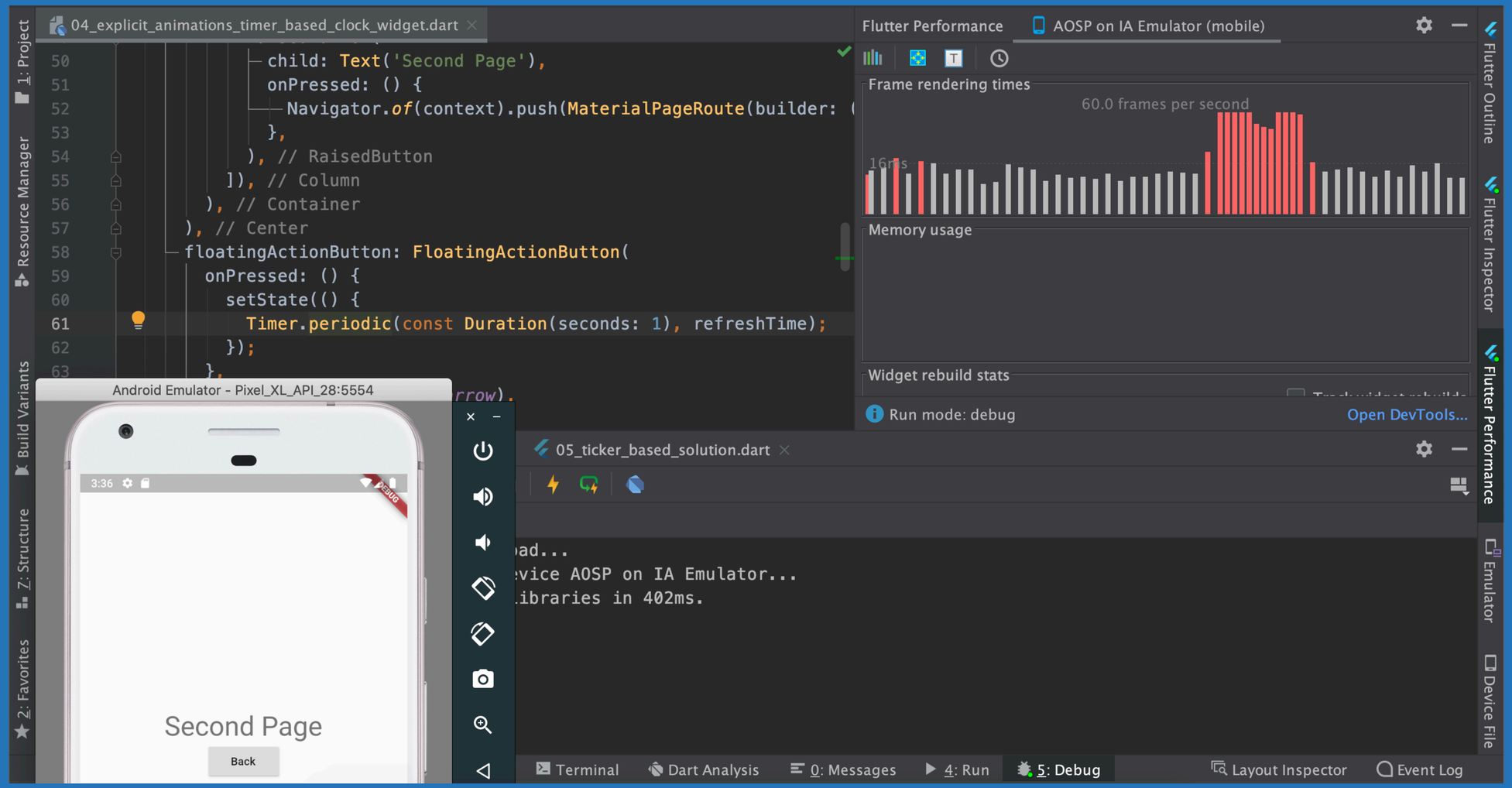
https://dartpad.dev/bbf00382c6d5f4f0fa34876c21b62b10



Timer Based solution



Timer Based solution



Базовый класс для всех анимаций

```
abstract class Animation<T> extends Listenable

  /// The current value of the animation.
  @override
  T get value;

  /// The current status of this animation.
  AnimationStatus get status;
```

Listeners на обновление значения анимации

```dart
abstract class Listenable {

  /// Register a closure to be called when the object notifies its listeners.
  void addListener(VoidCallback listener);

  /// Remove a previously registered closure from the list of closures that the
  /// object notifies.
  void removeListener(VoidCallback listener);
```

Контроллер анимации - AnimationController

```
class AnimationController extends Animation<double>

        /// The length of time this animation should last.
        Duration duration;

        TickerFuture forward()

        TickerFuture repeat()

        void stop()

        void reset()

        TickerFuture reverse()
```

What’s Animation Controller?

https://flutter.dev/docs/codelabs/explicit-animations#animationcontroller

AnimationController

The AnimationController is a special Animation object that generates a new value whenever the hardware is ready for a new frame. All explicit animations require an AnimationController.

What is an AnimationController?

The AnimationController class represents an interpolated range of values that define all possible frames for a particular animation. AnimationController has a value property that represents the current value of the animation within the range of other frame values. AnimationController is playable—it provides controls for triggering changes to its value property (between its lowerBound and upperBound) over a specified period of time (the duration parameter). Once triggered, AnimationController changes its value property over time to the other values in the range between upperBound and lowerBound. This change in value over time is what creates the animation effect. AnimationController is also highly configurable, allowing you to change the following:

Whether the animation should progress forward or backward through the range of values once triggered.

The amount(s) that an animation’s value changes between each frame.

Animation Controller

```
/// A controller for an animation.
///
/// This class lets you perform tasks such as:
///
/// * Play an animation [forward] or in [reverse], or [stop] an animation.
/// * Set the animation to a specific [value].
/// * Define the [upperBound] and [lowerBound] values of an animation.
/// * Create a [fling] animation effect using a physics simulation.
///
/// By default, an [AnimationController] linearly produces values that range
/// from 0.0 to 1.0, during a given duration. The animation controller generates
/// a new value whenever the device running your app is ready to display a new
/// frame (typically, this rate is around 60 values per second).
///
/// ## Ticker providers
///
/// An [AnimationController] needs a [TickerProvider], which is configured using
/// the `vsync` argument on the constructor.
```

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

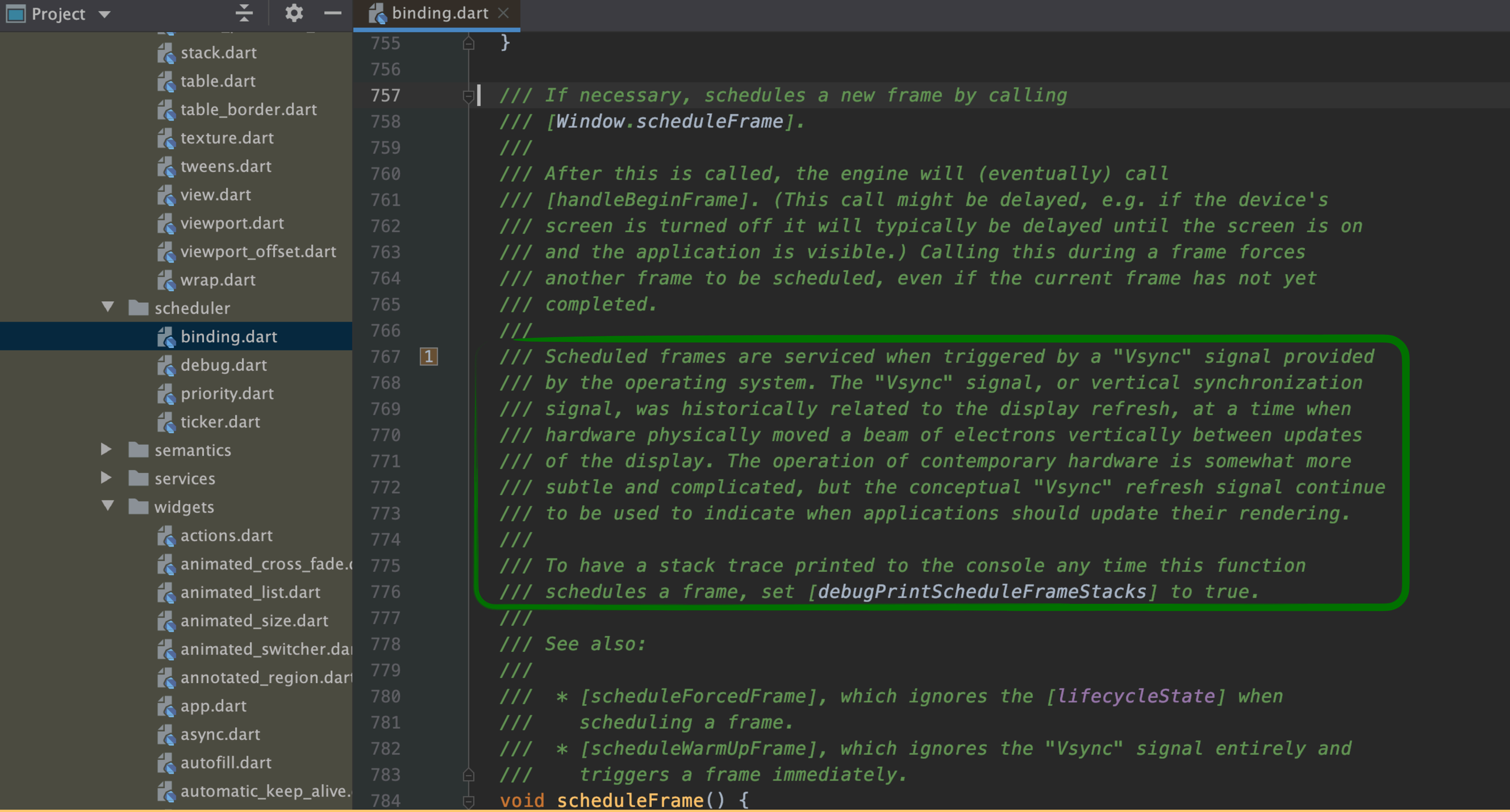
Вертика́льная синхрониза́ция (англ. *V-Sync*) — синхронизация кадровой частоты в компьютерной игре с частотой вертикальной развёртки монитора. При этом максимальный FPS с вертикальной синхронизацией приравнивается к частоте обновления монитора. Если FPS ниже частоты обновления монитора, то во избежание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


Project
binding.dart
stack.dart
table.dart
table_border.dart
texture.dart
tweens.dart
view.dart
viewport.dart
viewport_offset.dart
wrap.dart
scheduler
binding.dart
debug.dart
priority.dart
ticker.dart
semantics
services
widgets
actions.dart
animated_list.dart
animated_size.dart
app.dart
async.dart
autofill.dart
}
/// If necessary, schedules a new frame by calling
/// [Window.scheduleFrame].
///
/// After this is called, the engine will (eventually) call
/// [handleBeginFrame]. (This call might be delayed, e.g. if the device's
/// screen is turned off it will typically be delayed until the screen is on
/// and the application is visible.) Calling this during a frame forces
/// another frame to be scheduled, even if the current frame has not yet
/// completed.
///
/// Scheduled frames are serviced when triggered by a "Vsync" signal provided
/// by the operating system. The "Vsync" signal, or vertical synchronization
/// signal, was historically related to the display refresh, at a time when
/// hardware physically moved a beam of electrons vertically between updates
/// of the display. The operation of contemporary hardware is somewhat more
/// subtle and complicated, but the conceptual "Vsync" refresh signal continue
/// to be used to indicate when applications should update their rendering.
///
/// To have a stack trace printed to the console any time this function
/// schedules a frame, set [debugPrintScheduleFrameStacks] to true.
///
/// See also:
///
/// * [scheduleForcedFrame], which ignores the [lifecycleState] when
///   scheduling a frame.
/// * [scheduleWarmUpFrame], which ignores the "Vsync" signal entirely and
///   triggers a frame immediately.
void scheduleFrame() {

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.

Ticker

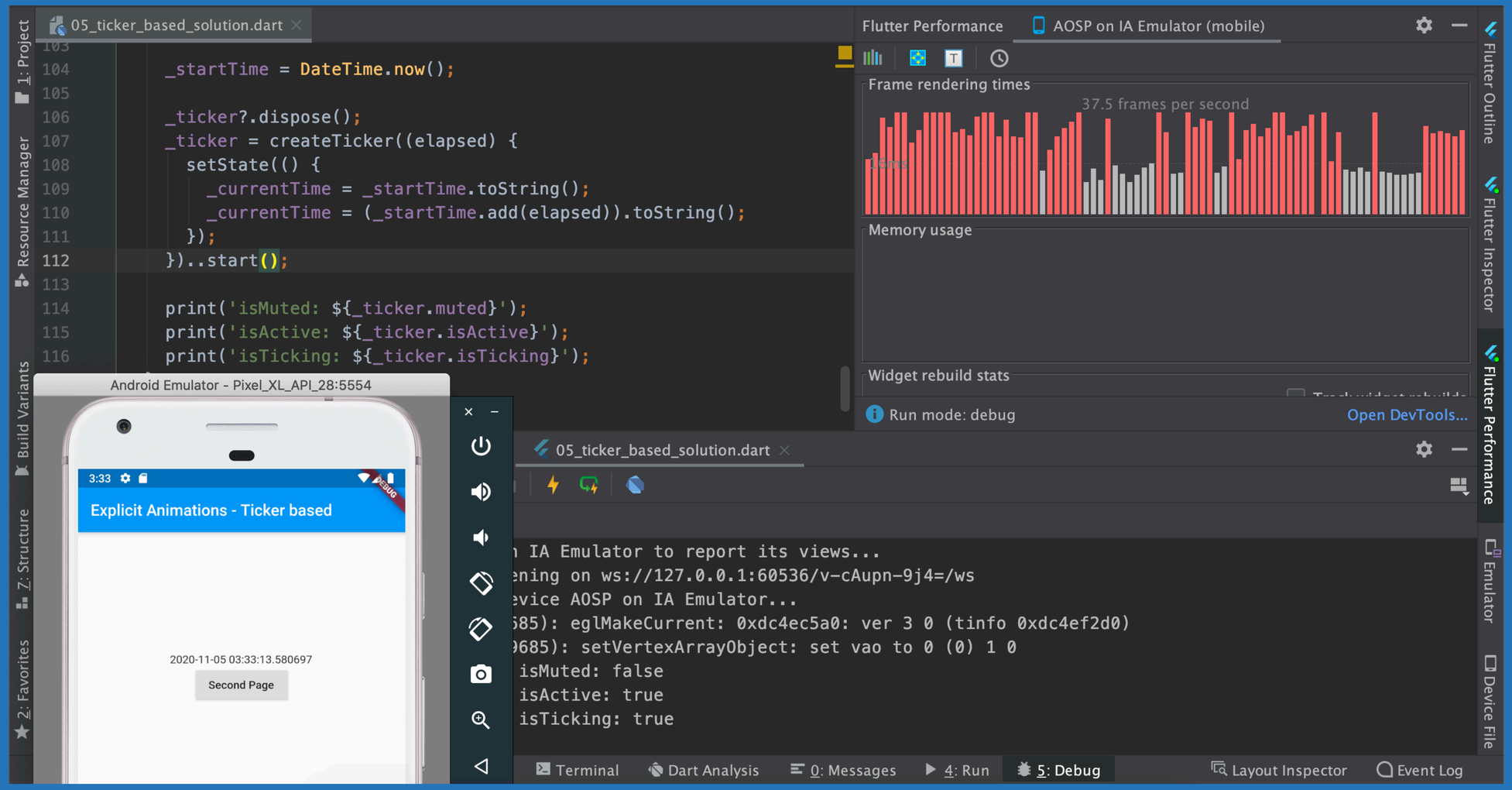
/// The methods that start animations return a [TickerFuture] object which
/// completes when the animation completes successfully, and never throws an
/// error; if the animation is canceled, the future never completes. This object
/// also has a [TickerFuture.orCancel] property which returns a future that
/// completes when the animation completes successfully, and completes with an
/// error when the animation is aborted.
///
/// This can be used to write code such as the `fadeOutAndUpdateState` method

Ticker

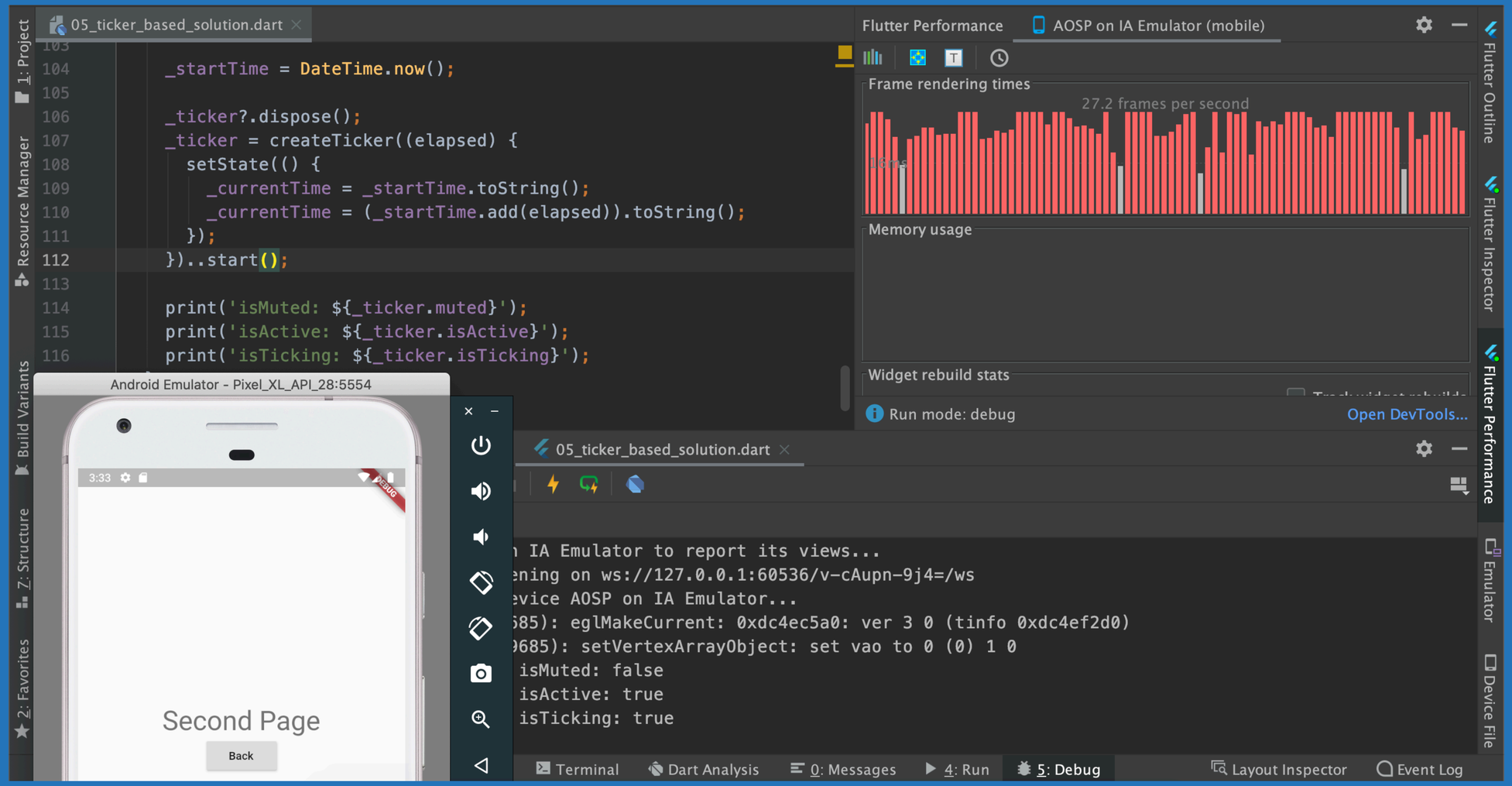
```
/// Calls its callback once per animation frame.
///
/// When created, a ticker is initially disabled. Call [start] to
/// enable the ticker.
///
/// A [Ticker] can be silenced by setting [muted] to true. While silenced, time
/// still elapses, and [start] and [stop] can still be called, but no callbacks
/// 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].
```

Ticker Based solution

Ticker Based solution



Ticker Based solution



Animation Controller - сценарий использования

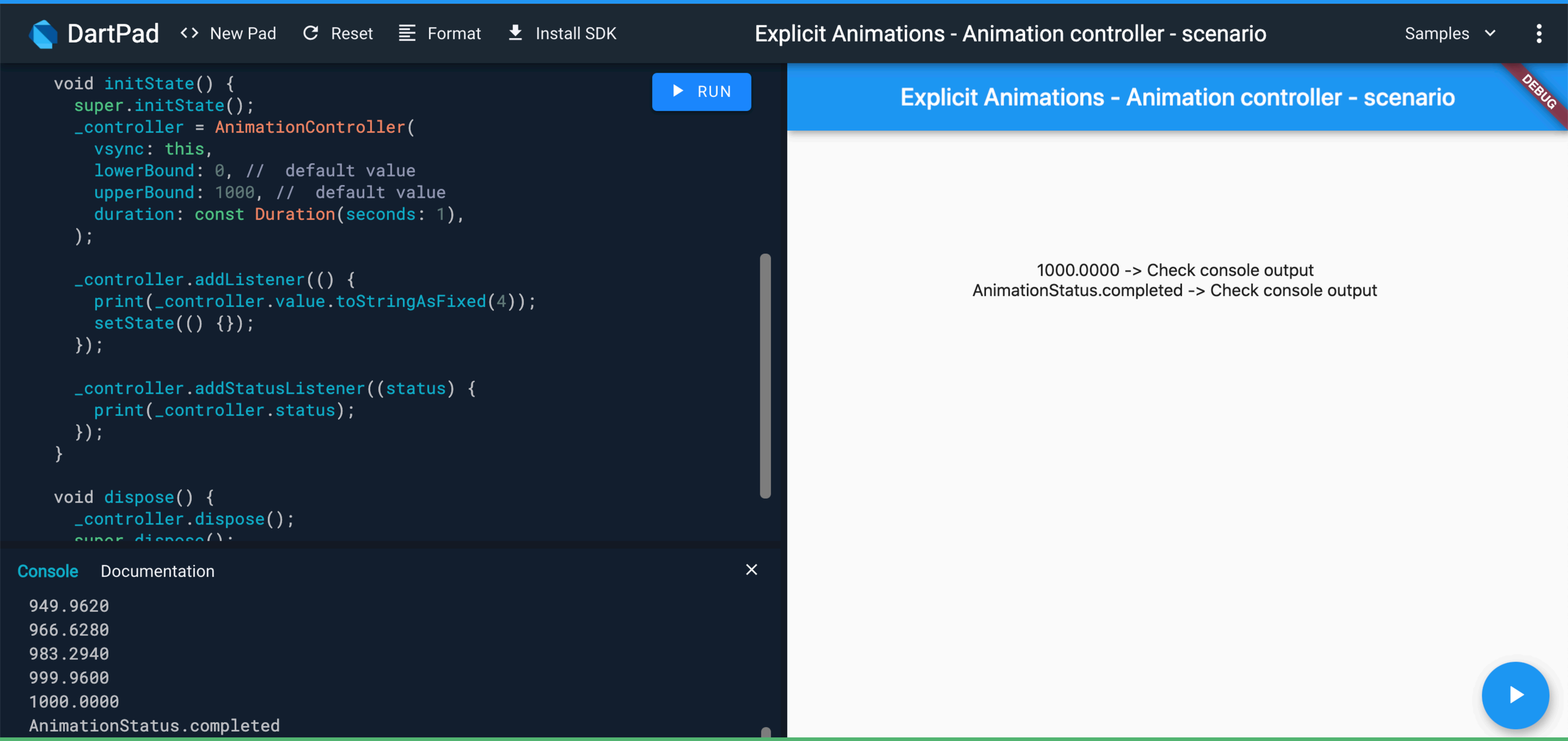
```
animationController = AnimationController(vsync: this,
                      duration: Duration(seconds: 2));

animationController.addListener(() {
  … здесь ваш код …
}

animationController.addStatusListener(() {
  … здесь ваш код …
}

animationController.repeat();
animationController.forward();
animationController.reverse();
animationController...();
```

Animation Controller - сценарий использования

```
class MyWidgetState extends State<MuWidget> with SingleTickerProviderStateMixin {
  AnimationController _controller;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(vsync: this, duration: const Duration(seconds: 1));
  }

  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) => ... using the _controller.value;
}
```

Scenario & Statuses

http://bit.ly/AnimationControllerScenario



Scenario & Statuses

http://dartpad.dev/f977a08d58c2b4567c3506793e18c992

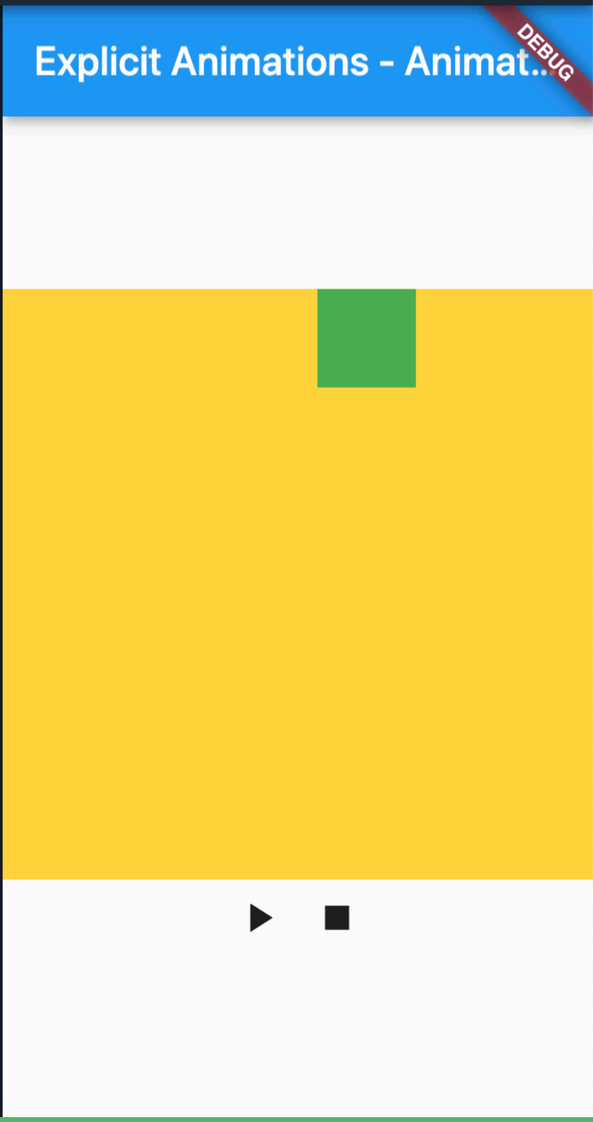
DartPad | New Pad | Reset | Format | Install SDK | Explicit Animations - Animation controller - statuses | Samples

```
@override
Widget build(BuildContext context) {
  _width = MediaQuery.of(context).size.width;
  return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: <Widget>[
      Row(
        children: [
          Expanded(
            child: Container(
              height: 300,
              color: Colors.amberAccent,
              child: Stack(children: <Widget>[
                Positioned(
                  child: Container(
                    width: 50,
                    height: 50,
                    color: Colors.green,
                  ),
                  left: _left,
                )
```

RUN

Console | Documentation

```
AnimationStatus.completed
AnimationStatus.reverse
AnimationStatus.dismissed
AnimationStatus.forward
AnimationStatus.completed
AnimationStatus.reverse
```



animateTo method

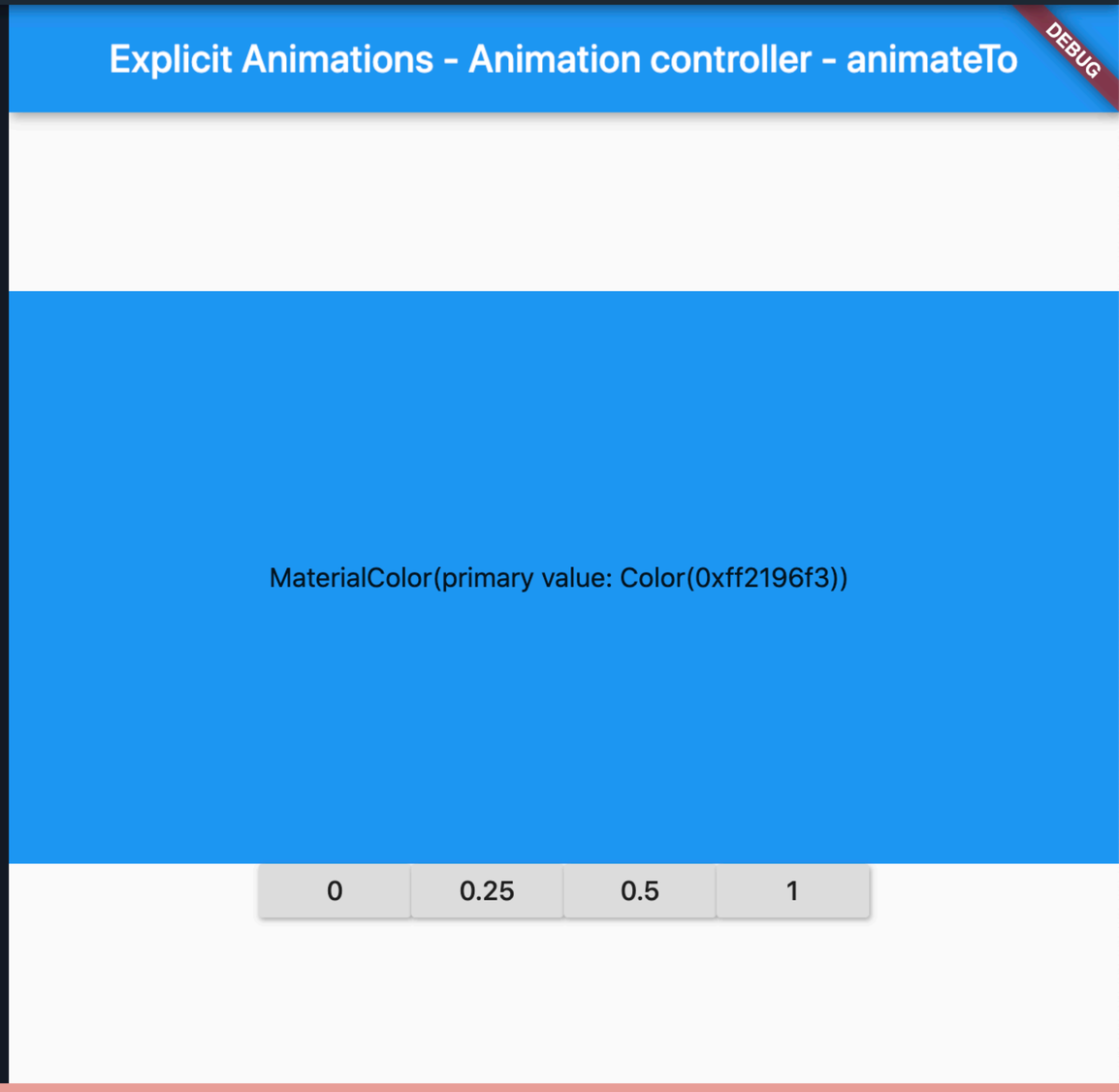
https://dartpad.dev/a73116bfe096308b91b6fce19261f439

DartPad | New Pad | Reset | Format | Install SDK | Explicit Animations - Animation controller - animateTo | Samples

RUN

```
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    RaisedButton(
      child: Text(0.toString()),
      onPressed: () {
        _controller.animateTo(0);
      },
    ),
    RaisedButton(
      child: Text(.25.toString()),
      onPressed: () {
        _controller.animateTo(.25);
      },
    ),
    RaisedButton(
      child: Text(.5.toString()),
      onPressed: () {
        _controller.animateTo(.5);
      },
    ),
    RaisedButton(
      child: Text(1.toString()),
      onPressed: () {
        _controller.animateTo(1);
      },
    ),
  ],
)
```

Console | Documentation



Explicit animation widgets

- AlignTransition, which is an animated version of Align.
- DecoratedBoxTransition, which is an animated version of DecoratedBox.
- DefaultTextStyleTransition, which is an animated version of DefaultTextStyle.
- PositionedTransition, which is an animated version of Positioned.
- RelativePositionedTransition, which is an animated version of Positioned.
- RotationTransition, which animates the rotation of a widget.
- ScaleTransition, which animates the scale of a widget.
- SizeTransition, which animates its own size.
- SlideTransition, which animates the position of a widget relative to its normal position.
- FadeTransition, which is an animated version of Opacity.
- AnimatedModalBarrier, which is an animated version of ModalBarrier.

Implicit & Explicit animation widgets

Implicit animation widgets

Animated<FooWidget>

Explicit animation widgets

<FooWidget>Transition

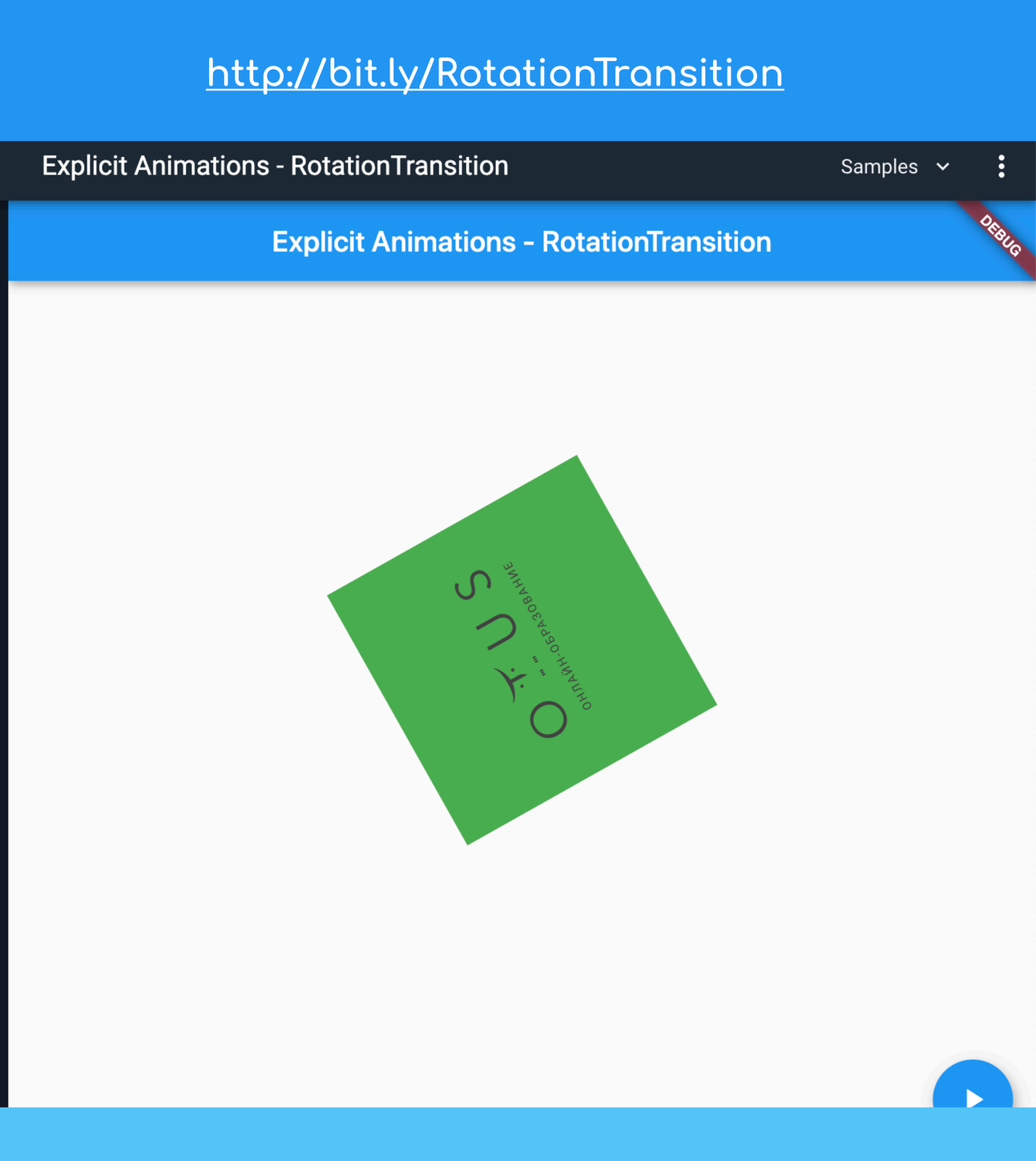
RotationTransition

http://bit.ly/RotationTransition

DartPad | New Pad | Reset | Format | Install SDK | Explicit Animations - RotationTransition | Samples

RUN

```
        title: Text('Explicit Animations - RotationTransition'),
      ),
      body: Center(
        child: Container(
          height: 300,
          child: Stack(
            children: <Widget>[
              RotationTransition(
                turns: _controller,
                child: OtusLogo(),
              ),
            ],
          ),
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          if (_controller.isAnimating) {
            _controller.stop();
          } else {
            _controller.repeat();
          }

          setState(() {});
        },
        child:
            _controller.isAnimating ? Icon(Icons.stop) :
Icon(Icons.play_arrow),
      ),
    );
  }
}
```



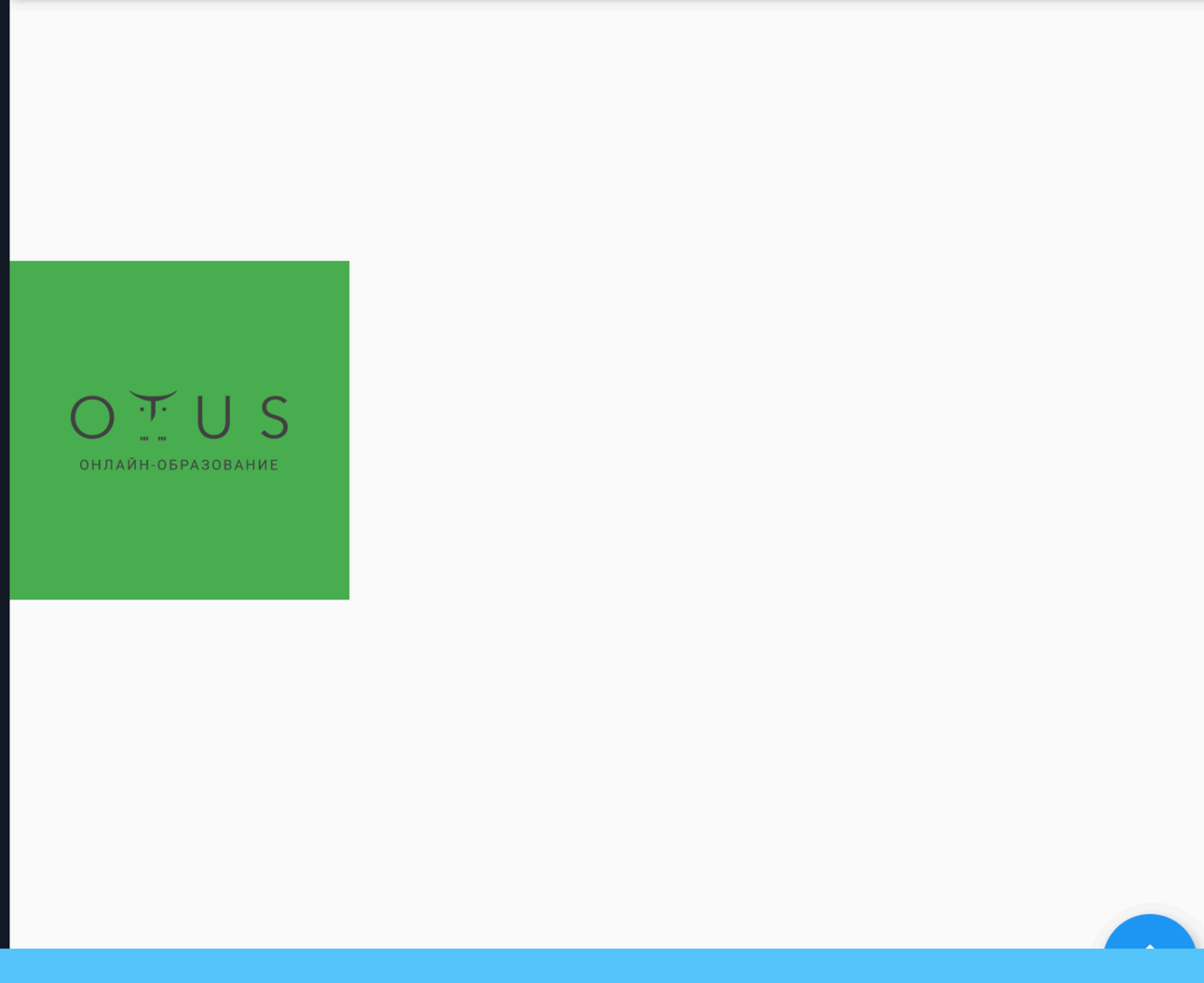
Прыжок в прошлое
Implicit Animations - AnimatedPositioned

http://bit.ly/AnimatedPositioned

DartPad | New Pad | Reset | Format | Install SDK | Implicit Animations - AnimatedPositioned | Samples

RUN

```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();
}

const double _logoWidth = 200;

class OtusLogo extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      child: Image.network('https://otus.ru/static/img/logo-2.8602b.svg'),
      width: _logoWidth,
      height: 200.0,
      color: Colors.green,
    );
  }
}
```



PositionedTransition

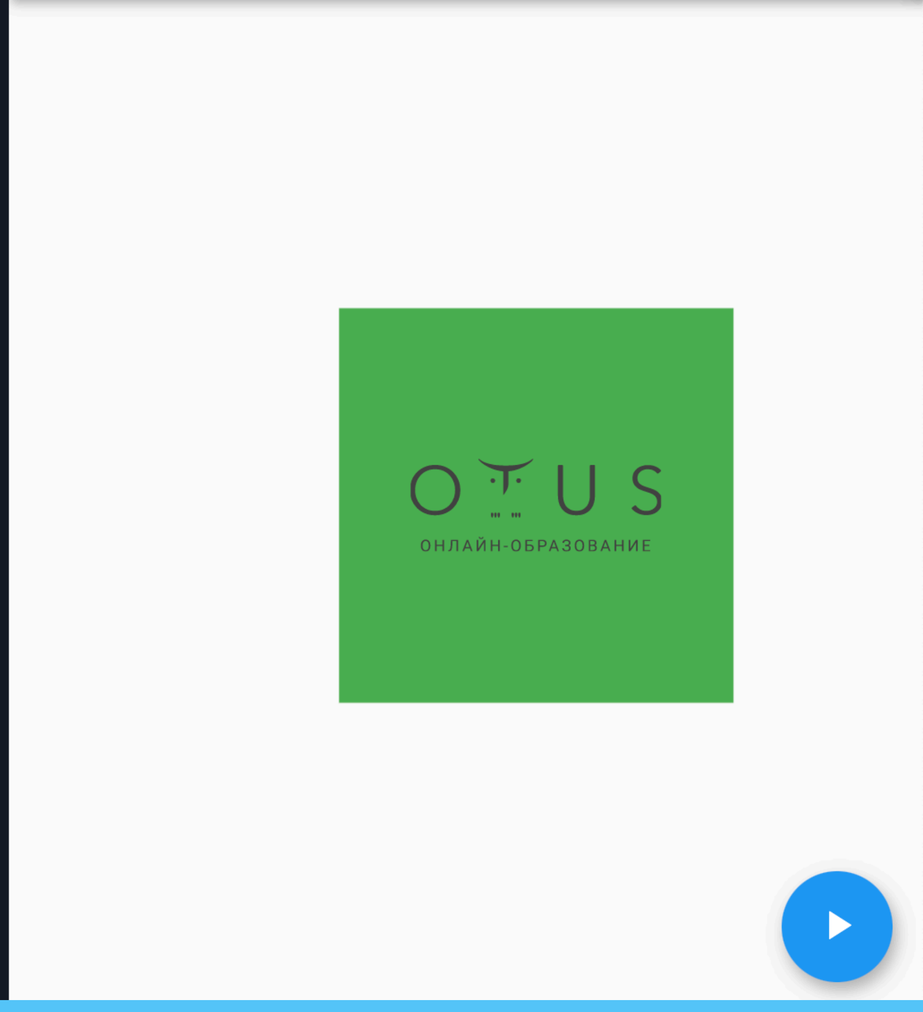
http://dartpad.dev/c4c07d816834f70ffb4ad7bfb00b66f6

DartPad | New Pad | Reset | Format | Install SDK | Explicit Animations - PositionedTransition | Samples

```
final padding = 50.0;
return Scaffold(
  appBar: AppBar(
    title: Text('Explicit Animations - PositionedTransition'),
  ),
  body: Center(
    child: Container(
      height: 300,
      child: Stack(
        children: <Widget>[
          PositionedTransition(
            rect: RelativeRectTween(
              begin: RelativeRect.fromLTRB(0, padding, size.width - _logoSize, padding),
              end: RelativeRect.fromLTRB((size.width - _logoSize), padding, 0, padding),
            ).animate(_controller),
            child: OtusLogo(),
          ),
        ],
      ),
    ),
  ),
  floatingActionButton: FloatingActionButton(
    onPressed: () {
      if (_controller.isAnimating) {
        _controller.stop();
      } else {
        _controller.repeat();
      }
```

RUN

Console | Documentation



PositionedTransition & Curves

https://dartpad.dev/f043afdcb805152f7bf7

DartPad | New Pad | Reset | Format | Install SDK | Explicit Animations - PositionedTransition + Curves | Samples

RUN

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

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

const double _logoSize = 200.0;

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

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

class OtusLogo extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      child: Image.network('https://otus.ru/static/img/logo-
2.8602b.svg'),
      width: _logoSize,
      height: _logoSize,
```

Console 1 | Documentation

Explicit Animations - PositionedTransition + Curves

DEBUG

Explicit animation widgets

- AlignTransition, which is an animated version of Align.
- DecoratedBoxTransition, which is an animated version of DecoratedBox.
- DefaultTextStyleTransition, which is an animated version of DefaultTextStyle.
- PositionedTransition, which is an animated version of Positioned.
- RelativePositionedTransition, which is an animated version of Positioned.
- RotationTransition, which animates the rotation of a widget.
- ScaleTransition, which animates the scale of a widget.
- SizeTransition, which animates its own size.
- SlideTransition, which animates the position of a widget relative to its normal position.
- FadeTransition, which is an animated version of Opacity.
- AnimatedModalBarrier, which is an animated version of ModalBarrier.

- **AnimatedBuilder**, which is useful for complex animation use cases and a notable exception to the naming scheme of AnimatedWidget subclasses.

Explicit animation widgets - Custom Transition

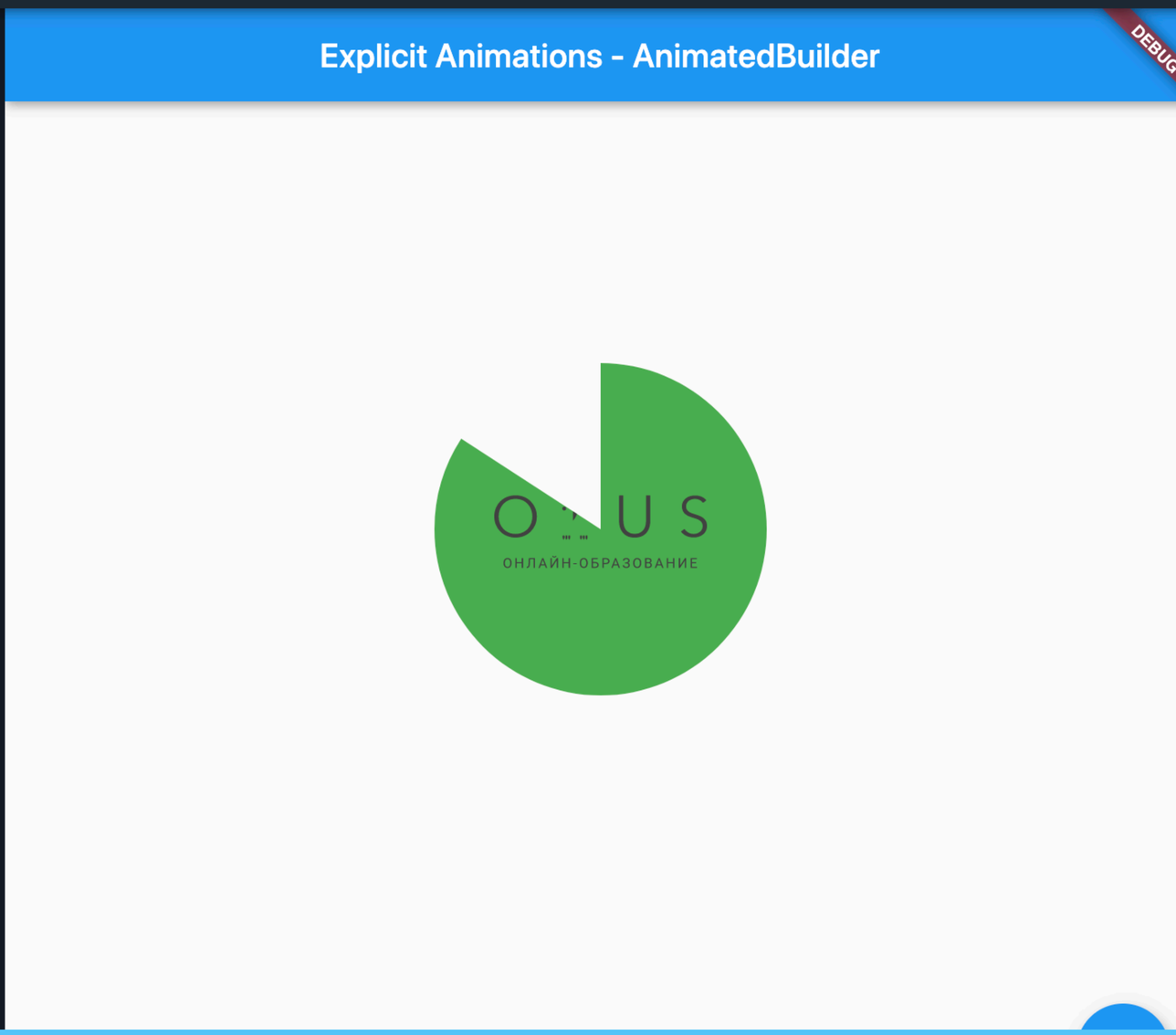
Animated Builder

http://bit.ly/AnimatedBuilder

DartPad | New Pad | Reset | Format | Install SDK | Explicit Animations - AnimatedBuilder | Samples

RUN

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

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

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

const double _logoSize = 200.0;

class OtusLogo extends StatelessWidget {
  const OtusLogo();

  @override
  Widget build(BuildContext context) {
    return Container(
      child: Image.network('https://otus.ru/static/img/logo-2.8602b.svg'),
      width: _logoSize,
      height: _logoSize,
      color: Colors.green,
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
```



Animated Widget

http://bit.ly/AnimatedWidget

```
/// A number of animated widgets ship with the framework. They are usually named
/// `FooTransition`, where `Foo` is the name of the non-animated
/// version of that widget. The subclasses of this class should not be confused
/// with subclasses of [ImplicitlyAnimatedWidget] (see above), which are usually
/// named `AnimatedFoo`. Commonly used animated widgets include:
///
///  * [AnimatedBuilder], which is useful for complex animation use cases and a
///    notable exception to the naming scheme of [AnimatedWidget] subclasses.
///  * [AlignTransition], which is an animated version of [Align].
///  * [DecoratedBoxTransition], which is an animated version of [DecoratedBox].
///  * [DefaultTextStyleTransition], which is an animated version of
///    [DefaultTextStyle].
///  * [PositionedTransition], which is an animated version of [Positioned].
///  * [RelativePositionedTransition], which is an animated version of
///    [Positioned].
///  * [RotationTransition], which animates the rotation of a widget.
///  * [ScaleTransition], which animates the scale of a widget.
///  * [SizeTransition], which animates its own size.
///  * [SlideTransition], which animates the position of a widget relative to
///    its normal position.
///  * [FadeTransition], which is an animated version of [Opacity].
///  * [AnimatedModalBarrier], which is an animated version of [ModalBarrier].
abstract class AnimatedWidget extends StatefulWidget {
```

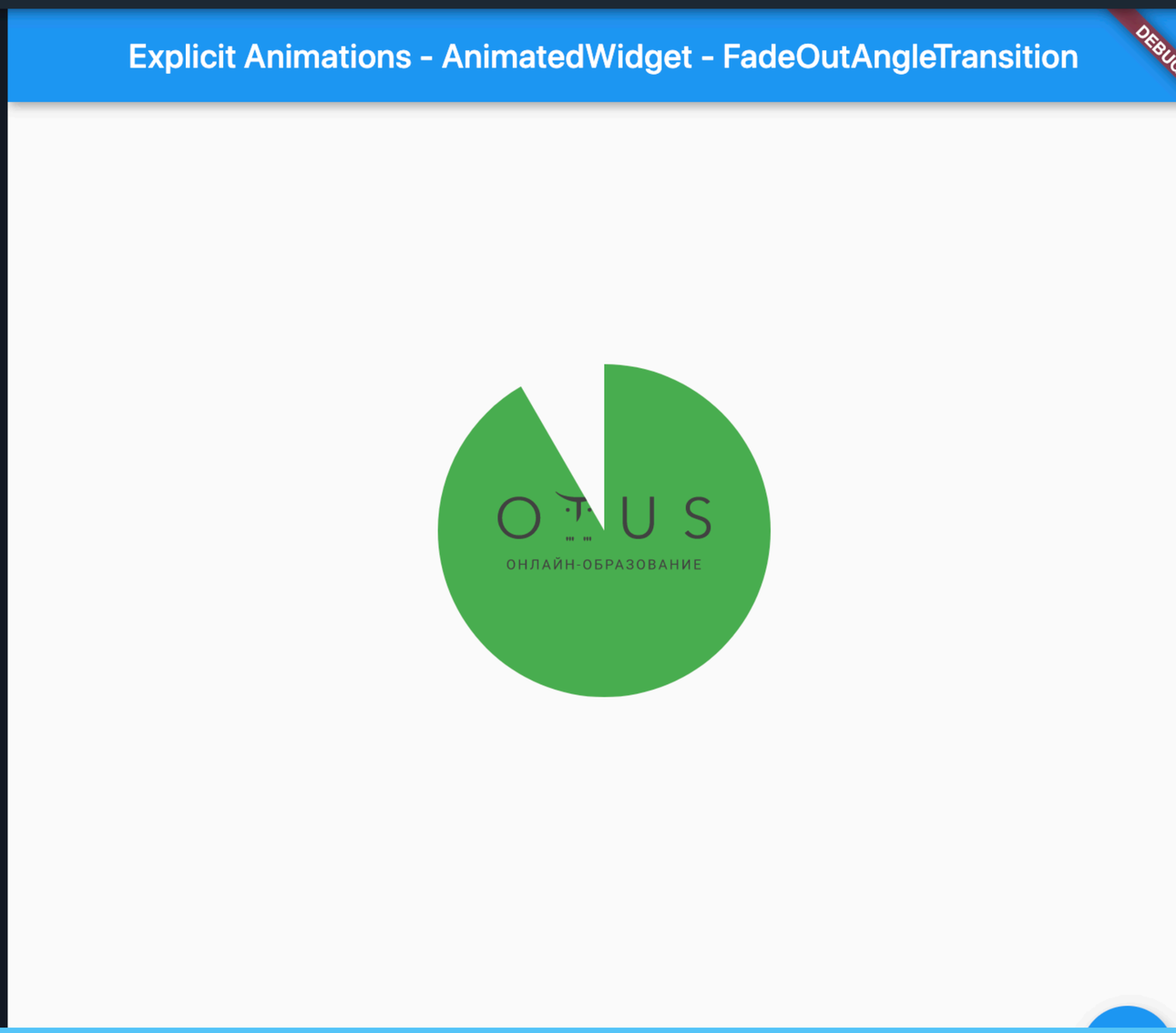
Animated Widget

http://bit.ly/AnimatedWidget

DartPad | New Pad | Reset | Format | Install SDK | Explicit Animations - AnimatedWidget - FadeOutAngleTransition | Samples

RUN

```
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Explicit Animations - AnimatedWidget -
FadeOutAngleTransition'),
      ),
      body: Center(
        child: Container(
          height: 300,
          child: Stack(
            children: <Widget>[
              FadeOutAngleTransition(
                animation: _controller,
                child: const OtusLogo(),
              ),
            ],
          ),
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          if (_controller.isAnimating) {
            _controller.stop();
          } else {
            _controller.repeat();
          }

          setState(() {});
        },
```



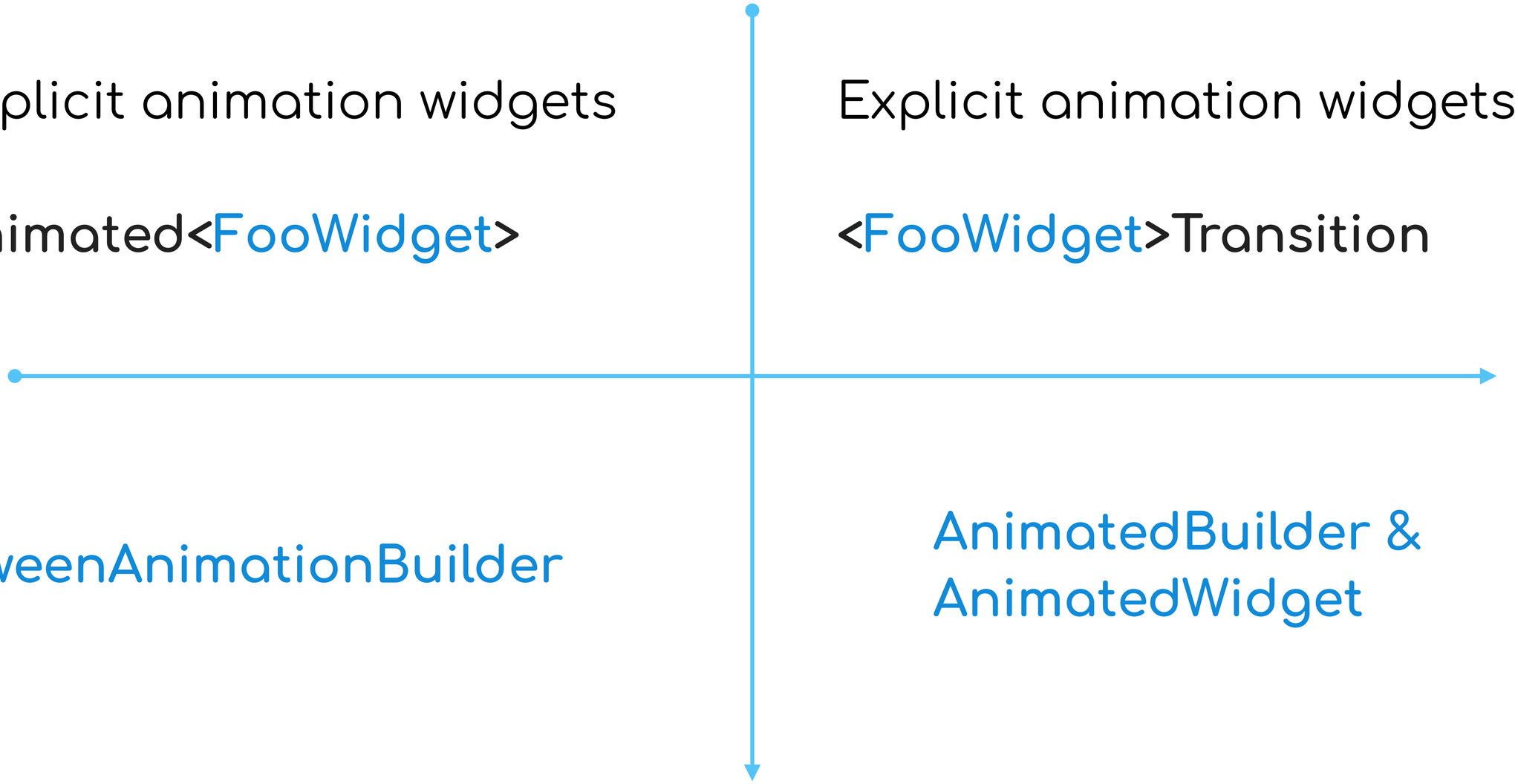
Explicit & Implicit animation widgets

Implicit animation widgets

Animated<FooWidget>

Explicit animation widgets

<FooWidget>Transition



TweenAnimationBuilder

AnimatedBuilder &
AnimatedWidget

Explicit animation widgets

3 вопроса

- Будет ли моя анимация повторяться? Например, при выполнении какого-либо условия должна ли моя анимация выполнять один и тот же эффект?
- Есть ли прерывания в моей анимации?
- Зависит ли моя анимация от выполнения другой?

ответили Да? Ваш выбор - Explicit Animation

Curves

CurvedAnimation

CurvedAnimation

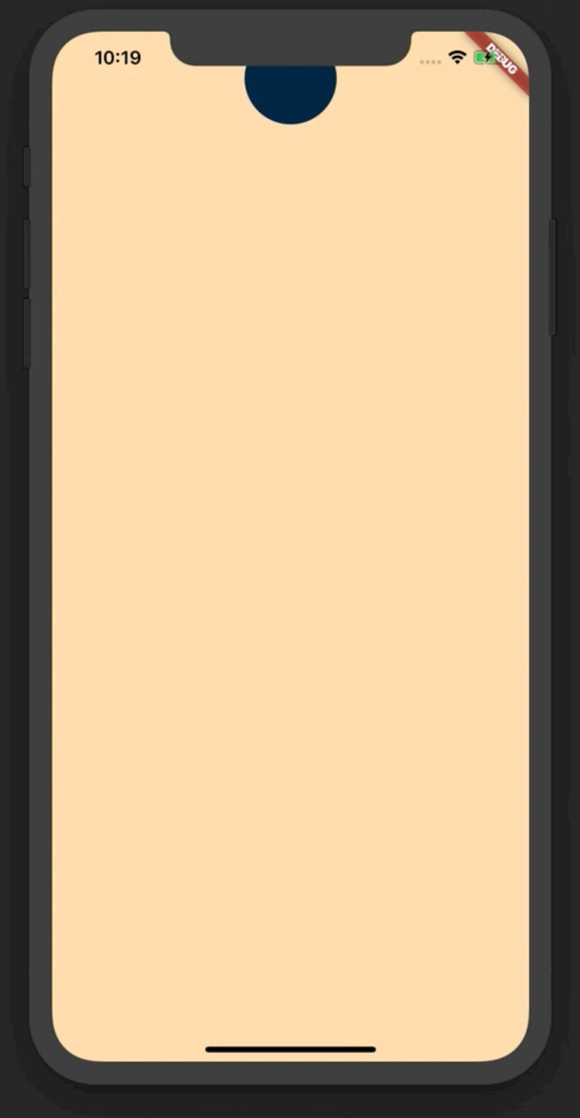
```dart
/// An animation that applies a curve to another
animation.
class CurvedAnimation extends Animation<double>

        /// The curve to use in the forward direction.
        Curve curve;


        /// The animation to which this animation applies a curve.
        @override
        final Animation<double> parent;
```

CurvedAnimation

```
animation = CurvedAnimation(
  parent: animationController,
  curve: Curves.linear,
);
```

CurvedAnimation - Настройка кривой

```
class ShapePainter extends CustomPainter {
  @override
  void paint(Canvas canvas, Size size) {
    canvas.drawCircle(
      Offset(size.width / 2, size.height / 2),
      size.width / 2 - 10,
      paint, //


animationController = AnimationController(vsync: this,
                    duration: Duration(seconds: 2));


animation = CurvedAnimation(parent: animationController,
 curve: Curves.bounceOut,
);

animationController.repeat();


Positioned( top: (size.height - 100) *  animation.value,
 child: CustomPaint(
   painter: ShapePainter(),
```



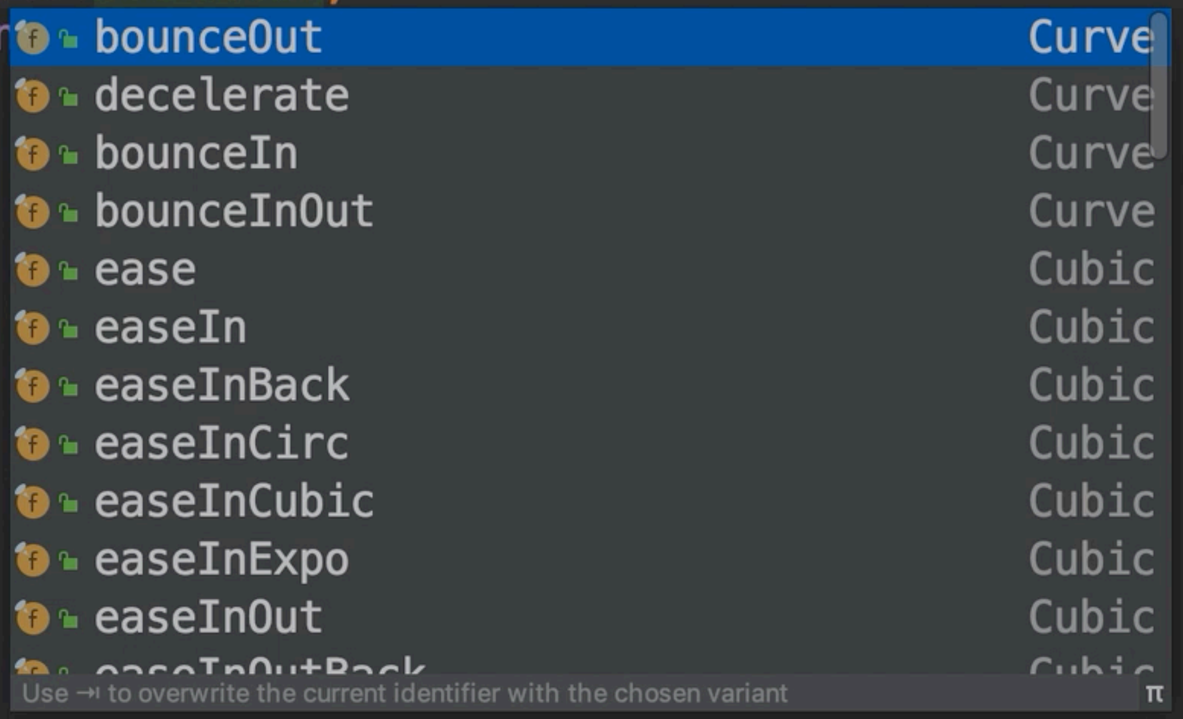
CurvedAnimation - Настройка кривой

```
animation = CurvedAnimation(
  parent: animationController,
  curve: Curves.decelerate,
);
```

```
class DecelerateCurve extends Curve {
  @override
  double transform(double t) {
    t = 1.0 - t;
    return 1.0 - t * t;
..
```

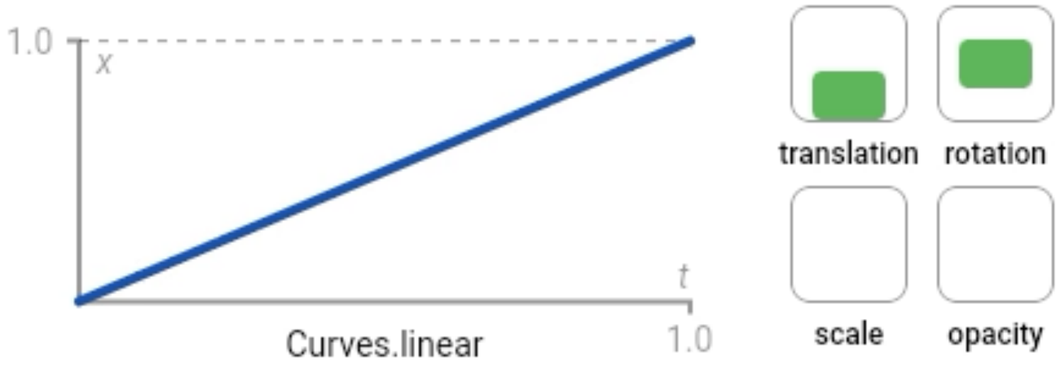
CurvedAnimation - Curves в Flutter SDK

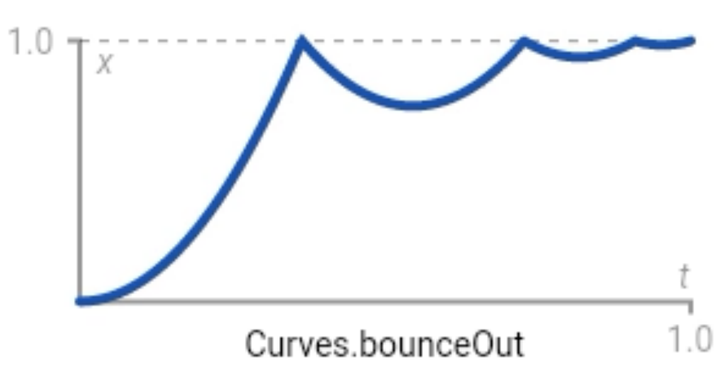
```
_animation = CurvedAnimation(
  curve: Curves.bounceOut,
  parent: _ar
);
```

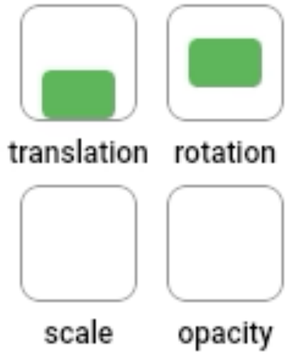


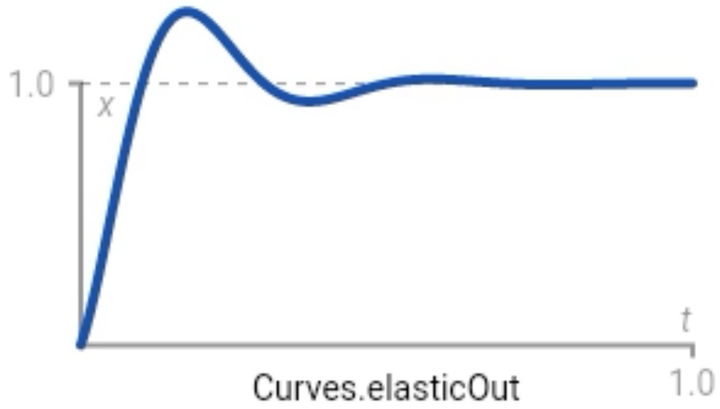
Flutter Curves

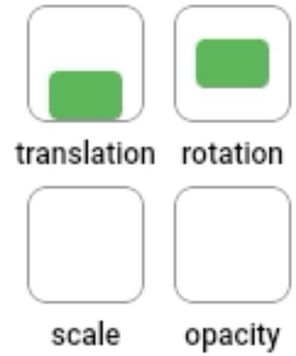
http://bit.ly/flutter-curves

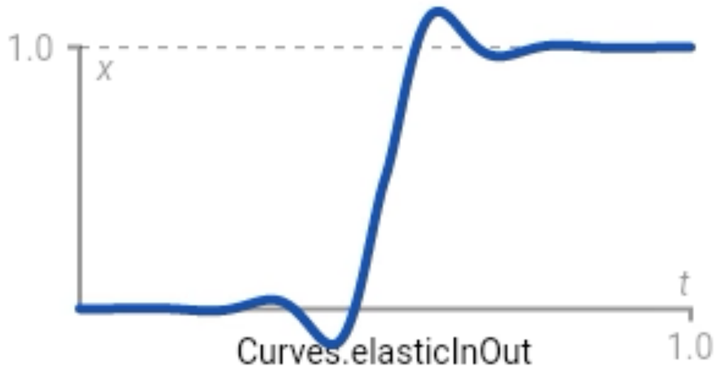


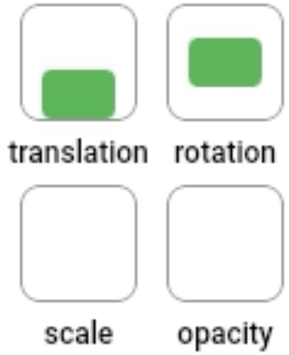












Lerp & Tween

А если я не нахожу Animated<Foo> widget?

Animated<Color? Some Behaviour?> widget

Lerp & Tween

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.

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;
```

Animatable

```dart
/// An object that can produce a value of type `T` given an [Animation<double>]
/// as input.
///
/// Typically, the values of the input animation are nominally in the range 0.0
/// to 1.0. In principle, however, any value could be provided.
///
/// The main subclass of [Animatable] is [Tween].
abstract class Animatable<T> {
```

Animatable

```
/// The current value of this object for the given [Animation].
///
/// This function is implemented by deferring to [transform]. Subclasses that
/// want to provide custom behavior should override [transform], not
/// [evaluate].
///
/// See also:
///
///  * [transform], which is similar but takes a `t` value directly instead of
///    an [Animation].
///  * [animate], which creates an [Animation] out of this object, continually
///    applying [evaluate].
T evaluate(Animation<double> animation) => transform(animation.value);
```

Tween - evaluate

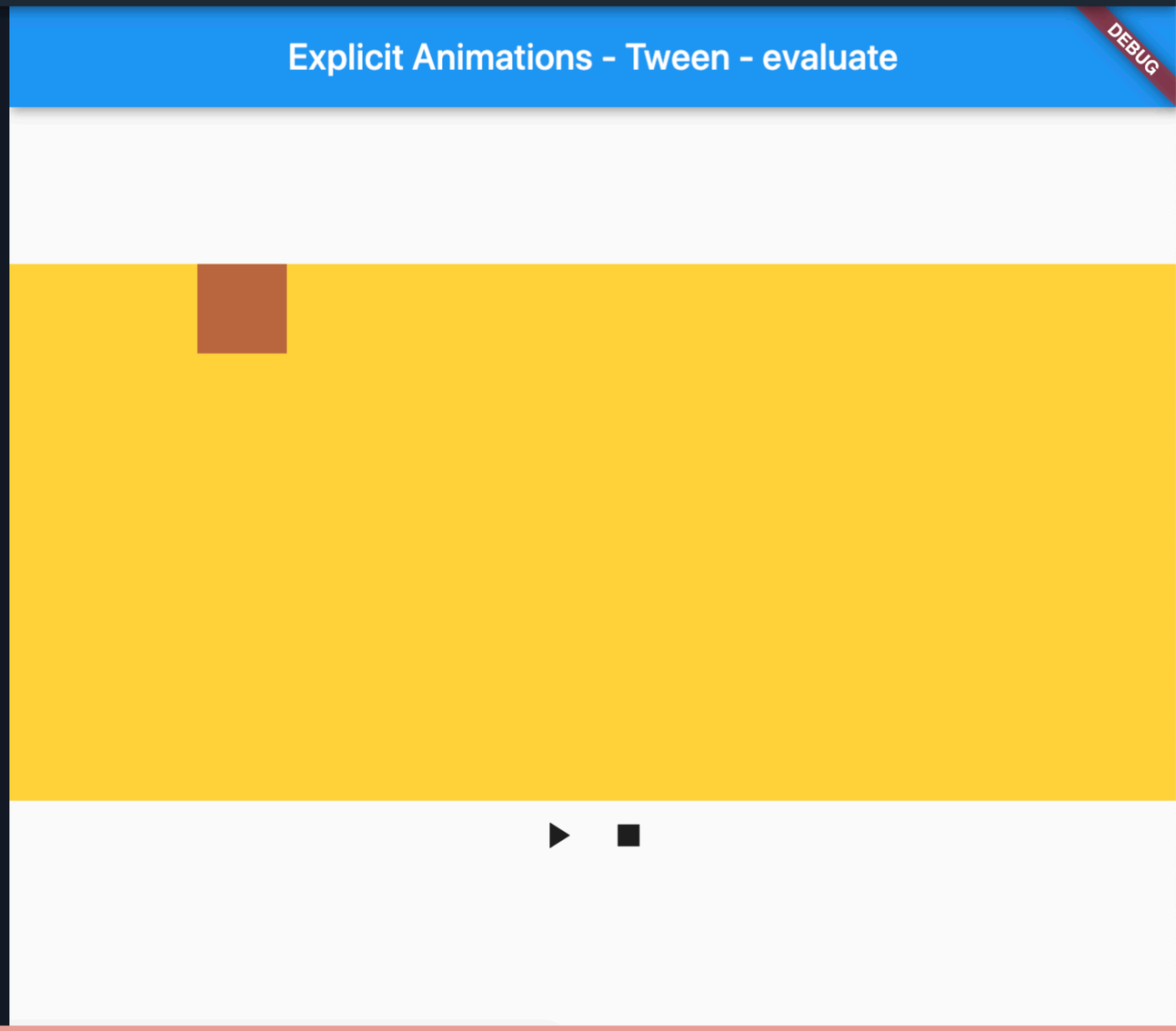
https://dartpad.dev/399df0b4da9db32cc3ee63a17077ec3c

DartPad | New Pad | Reset | Format | Install SDK | Explicit Animations - Tween - evaluate | Samples

RUN

```
        children: [
          Expanded(
            child: Container(
              height: 300,
              color: Colors.amberAccent,
              child: Stack(children: <Widget>[
                Positioned(
                  child: Container(
                    width: 50,
                    height: 50,
                    color: _colorTween.evaluate(_controller),
                  ),
                  left: _leftTween.evaluate(_controller),
                )
              ]),
            ),
          ),
        ],
      ),
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          IconButton(
            icon: Icon(Icons.play_arrow),
            onPressed: () {
              _controller.forward();
            },
          ),
          IconButton(
```

Console 5 | Documentation



Animatable

```
/// Returns a new [Animation] that is driven by the given animation but that
/// takes on values determined by this object.
///
/// Essentially this returns an [Animation] that automatically applies the
/// [evaluate] method to the parent's value.
///
/// See also:
///
///  * [AnimationController.drive], which does the same thing from the
///    opposite starting point.
Animation<T> animate(Animation<double> parent) {
  return _AnimatedEvaluation<T>(parent, this);
}

/// Returns a new [Animatable] whose value is determined by first evaluating
/// the given parent and then evaluating this object.
///
/// This allows [Tween]s to be chained before obtaining an [Animation].
Animatable<T> chain(Animatable<double> parent) {
  return _ChainedEvaluation<T>(parent, this);
}
```

Tween - animate

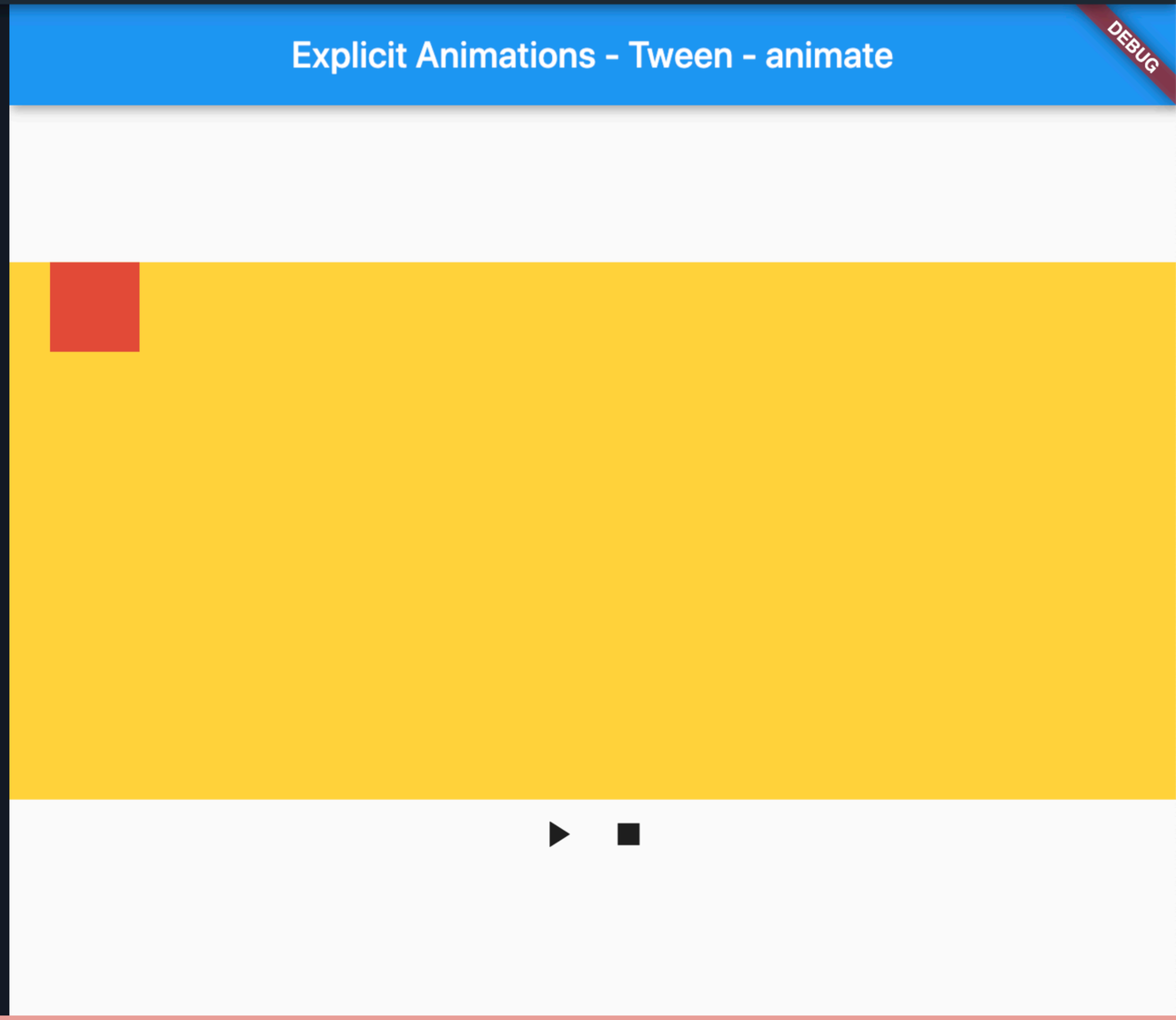
https://dartpad.dev/769bb3608cbf4dc5522d4c3819b9cdb4

DartPad | New Pad | Reset | Format | Install SDK | Explicit Animations - Tween - animate | Samples

```
    _controller = AnimationController(vsync: this, duration:
Duration(seconds: 1));
    _controller.addListener(() {
      setState(() {});
    });
    _controller.addStatusListener(_handleAnimationStatusChange);

    _curvedAnimation = CurvedAnimation(
      parent: _controller,
      curve: Curves.bounceOut,
      reverseCurve: Curves.bounceIn,
    );

    _colorAnimation = _colorTween.animate(_controller);
    _leftAnimation = _leftTween.animate(_curvedAnimation);

    super.initState();
  }

  void _handleAnimationStatusChange(AnimationStatus status) {
    print(status);
    if (status == AnimationStatus.completed) {
      _controller.reverse();
    } else if (status == AnimationStatus.dismissed) {
      _controller.forward();
    }
  }
}
```

RUN

Console 77 Documentation



Tween - custom

https://dartpad.dev/c88181c976ceccbef31e9a1f33101ced

DartPad   New Pad   Reset   Format   Install SDK

Custom Tween Implementation - Writer

Samples

RUN

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

void main() {
  runApp(MyApp());
}

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

class TypeWriterTween extends Tween<String> {
  TypeWriterTween({String begin = '', String end}) : super(begin: begin, end: end);

  @override
  String lerp(double t) {
    var currentLengthOfText = (end.length * t).round();
    return end.substring(0, currentLengthOfText);
  }
}

class AnimatedBuilderDemo extends StatefulWidget {
  @override
  _AnimatedBuilderDemoState createState() => _AnimatedBuilderDemoState();
```

DEBUG

Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lore Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum

Console   Documentation

Tween- TweenSequence

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

final controller = null;
final sizeAnimation = TweenSequence(
  [
    TweenSequenceItem(tween: Tween(begin: 0.0, end: 100.0), weight: 1),
    TweenSequenceItem(tween: Tween(begin: 100.0, end: 50.0), weight: 1),
    TweenSequenceItem(tween: Tween(begin: 50.0, end: 75.0), weight: 1),
    TweenSequenceItem(tween: Tween(begin: 75.0, end: 0.0), weight: 1),
  ],
).animate(controller);
```

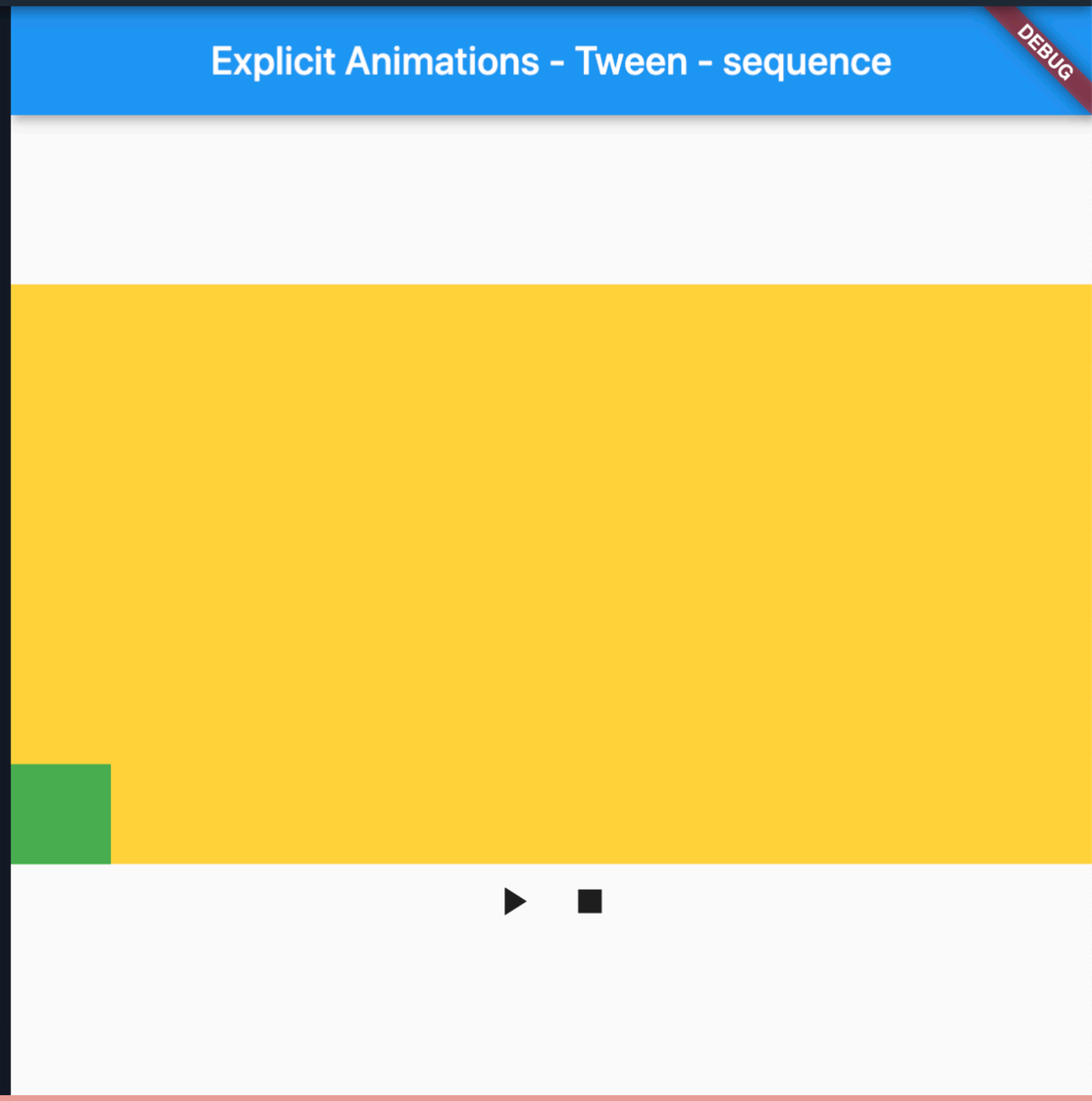
Tween- TweenSequence

http://dartpad.dev/e97a1d50856b7e7d0a5cabed7a983b7e

DartPad | New Pad | Reset | Format | Install SDK | Explicit Animations - Tween - sequence | Samples

```
    _controller.addListener(() {
      setState(() {});
    });
    _controller.addStatusListener(_handleAnimationStatusChange);

    _sizeAnimation = TweenSequence(
      [
        TweenSequenceItem(tween: Tween(begin: 50.0, end: 100.0), weight: 1),
        TweenSequenceItem(tween: Tween(begin: 100.0, end: 50.0), weight: 1),
        TweenSequenceItem(tween: Tween(begin: 50.0, end: 75.0), weight: 1),
        TweenSequenceItem(tween: Tween(begin: 75.0, end: 50.0), weight: 1),
      ],
    ).animate(_controller);

    super.initState();
  }

  void _handleAnimationStatusChange(AnimationStatus status) {
    print(status);
    if (status == AnimationStatus.completed) {
      _controller.reverse();
    } else if (status == AnimationStatus.dismissed) {
      _controller.forward();
    }
  }
}
```

RUN

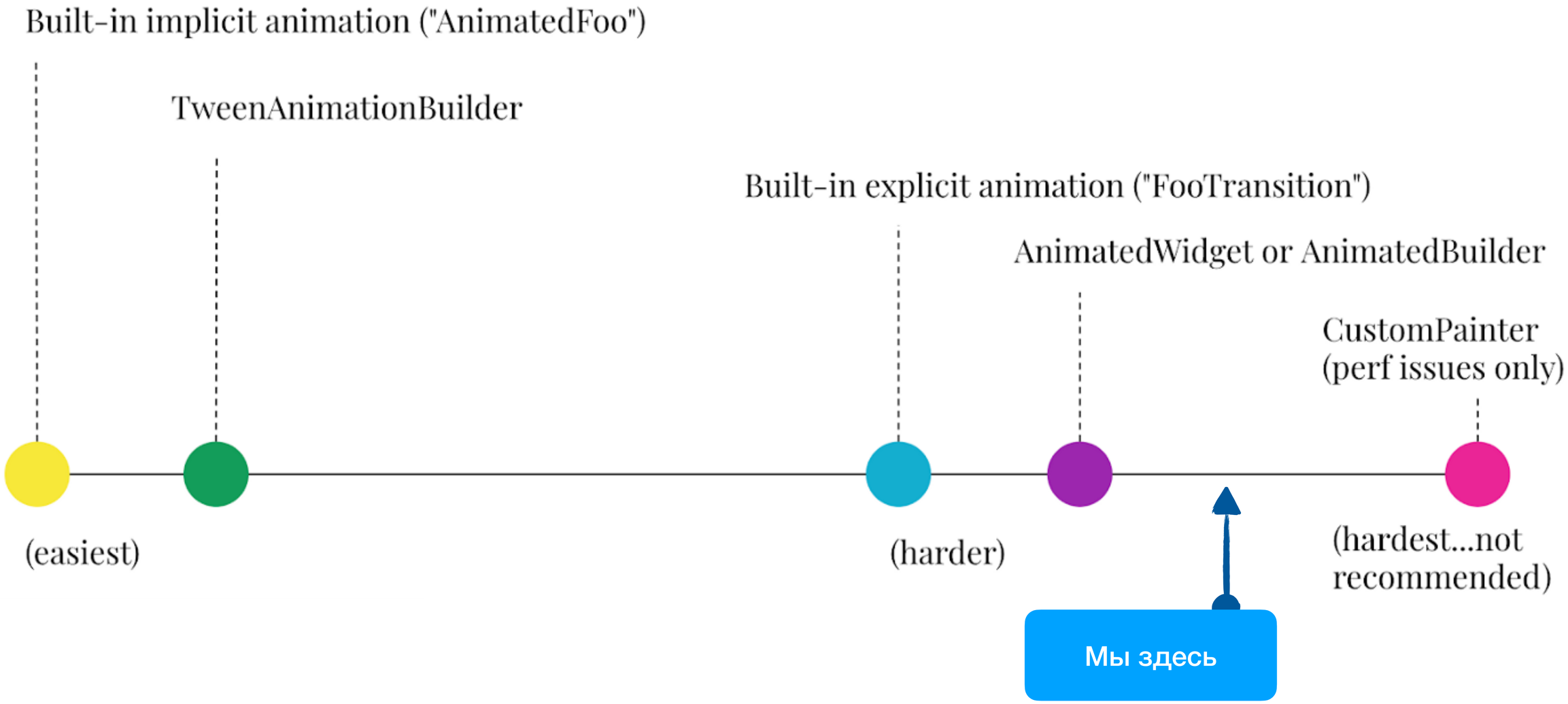


Console 79 | Documentation

Сегодня мы рассмотрели:

- Implicit animation widgets - вспомним
- Animation Controller
- Explicit animation widgets
- Curves
- Tweens

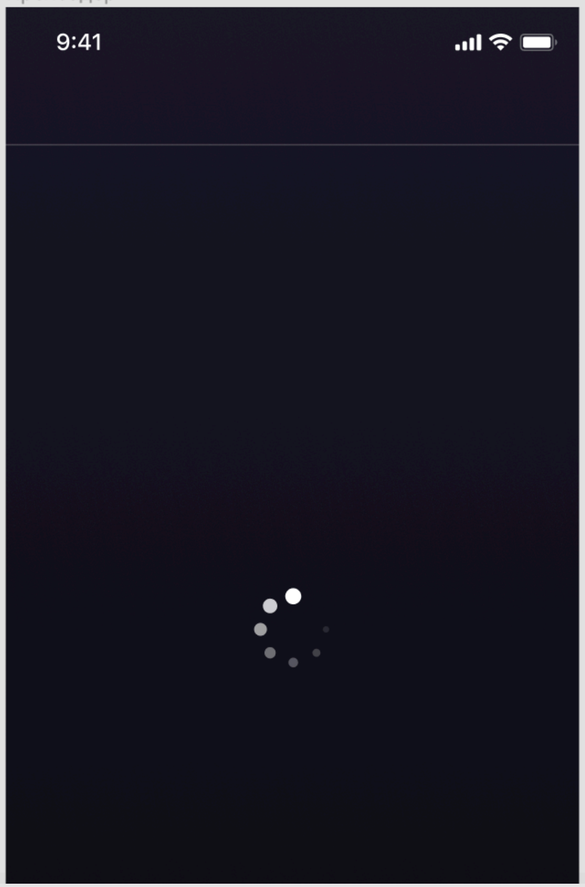
Итоги


Built-in implicit animation ("AnimatedFoo")
TweenAnimationBuilder
Built-in explicit animation ("FooTransition")
AnimatedWidget or AnimatedBuilder
CustomPainter
(perf issues only)
(easiest)
(harder)
(hardest...not
recommended)
Мы здесь

Дальше

Прелоадер

Прелоадер



Заполните, пожалуйста,
опрос о занятии по ссылке в чате

Спасибо за внимание!
Приходите на следующие вебинары

Смирнов Андрей wrike

Курс **Мобильная разработка на Flutter**