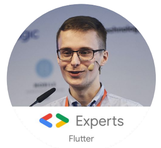
Experts

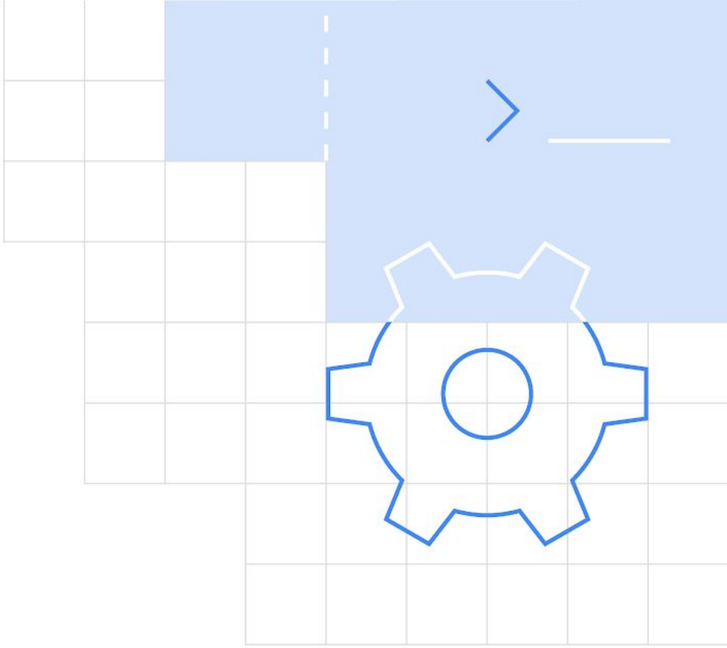
Improve your animations skills in Flutter

Animations made easier



Dominik Roszkowski
@OrestesGaolin
Principal Engineer at Very Good Ventures

Google Developers

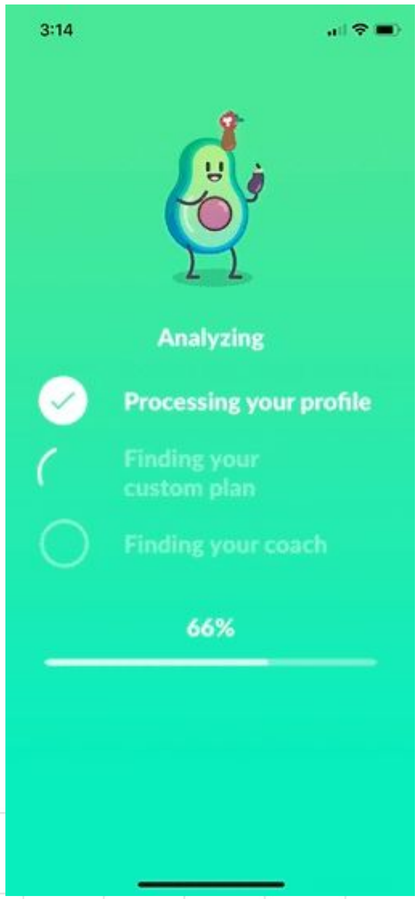


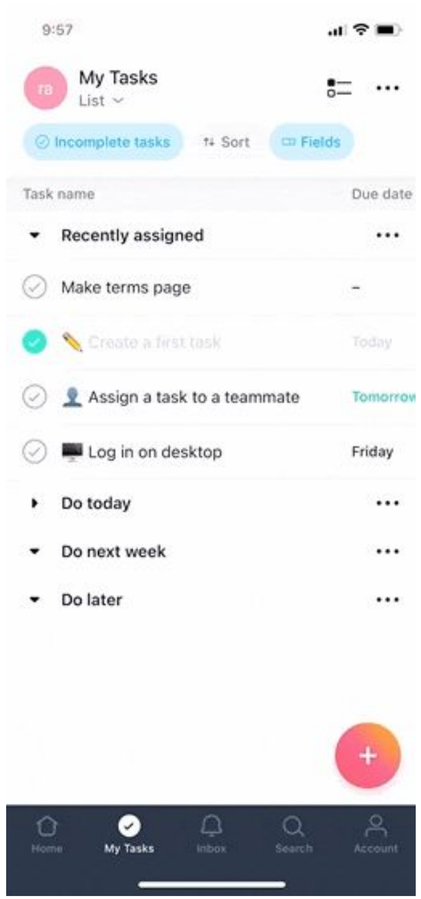
verygood.ventures/careers

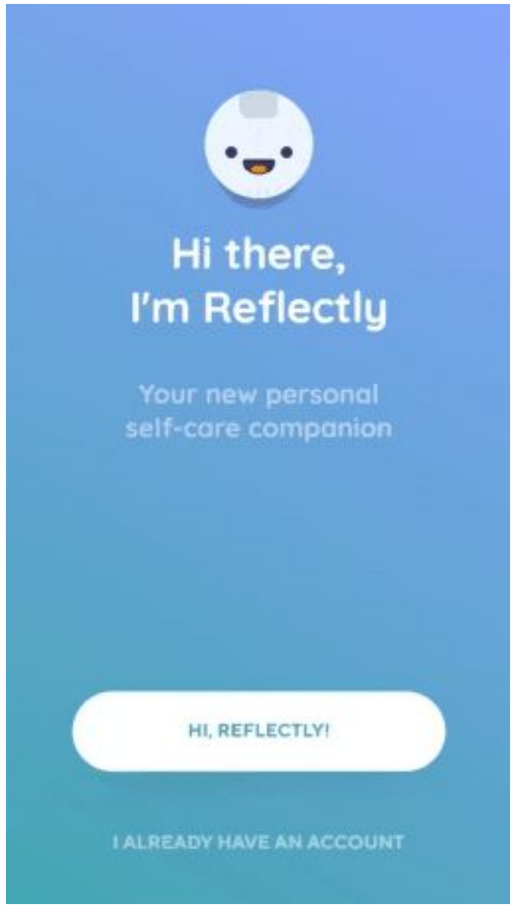
We’re hiring :)

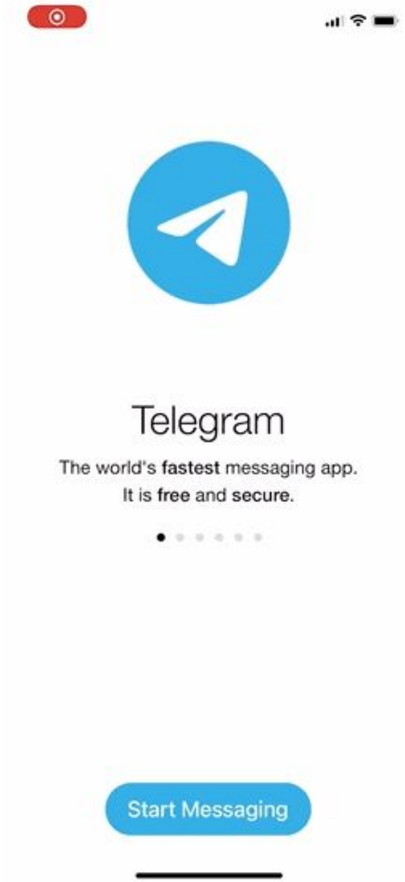
Experts

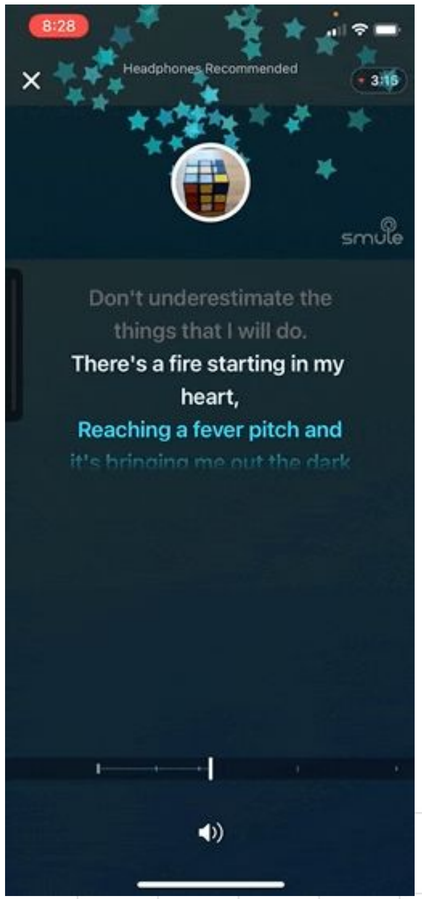
Animations are everywhere

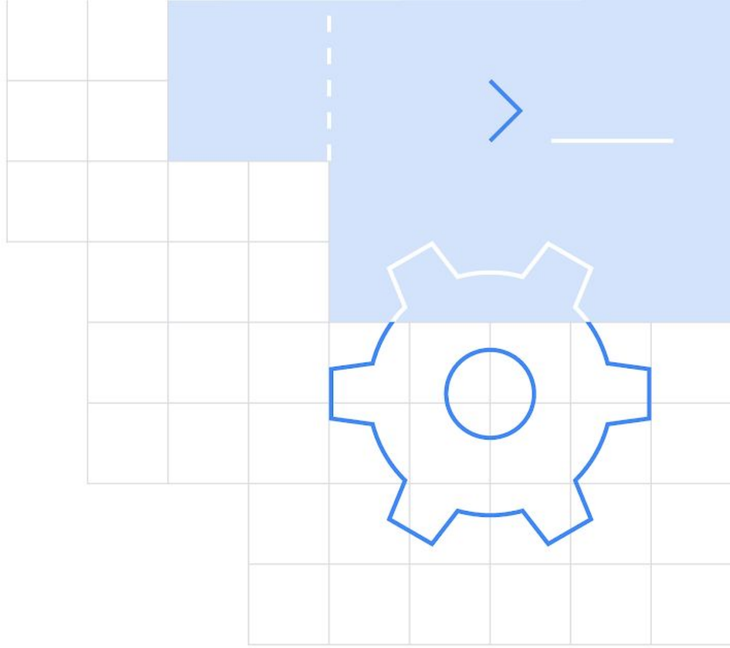












Implicit vs Explicit Animations

Animation basics

Experts

Implicit animations

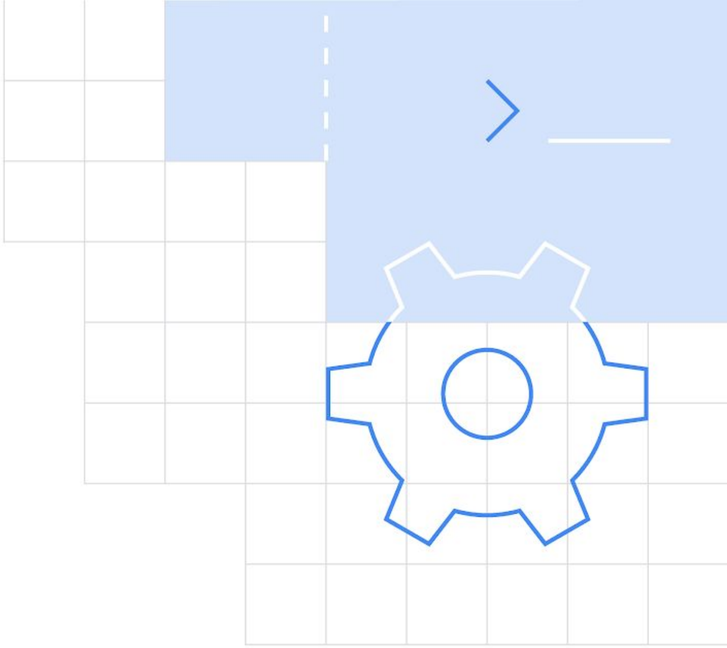
Animating properties without thinking about it

In general widgets starting with *Animated*:

- AnimatedPositioned/Align
- AnimatedOpacity
- AnimatedContainer
- AnimatedPadding
- AnimatedCrossFade
- AnimatedSwitcher
- AnimatedPhysicalModel

...

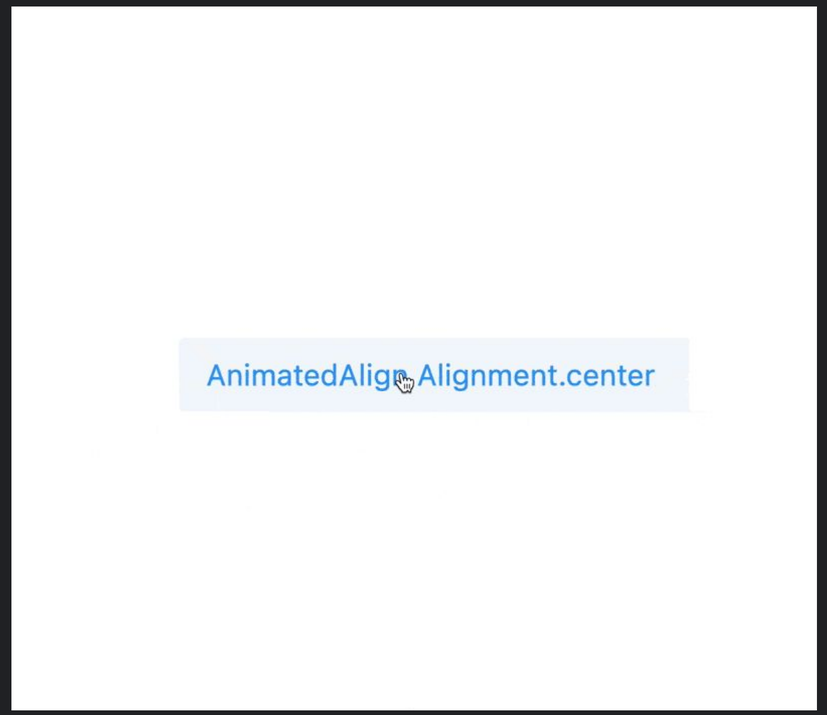
Demo App



roszkowski.dev/animations

Experts

```dart
AnimatedAlign(
  child: TextButton(
    onPressed: () {
      setState(() {
        alignment = Alignment(random1, random2);
      });
    },
    child: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Text('AnimatedAlign $alignment'),
    ),
  ),
  duration: kThemeAnimationDuration,
  alignment: alignment,
),
```



Custom ImplicitlyAnimatedWidget

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

class SmoothLoadingIndicator extends ImplicitlyAnimatedWidget {
 SmoothLoadingIndicator({
   required this.progress,
   Duration duration = const Duration(milliseconds: 100),
   Curve curve = Curves.linear,
   this.color = Colors.blueAccent,
   this.backgroundColor = Colors.blue,
 }) : super(duration: duration, curve: curve);

 final double progress;
 final Color color;
 final Color backgroundColor;

 @override
 ImplicitlyAnimatedWidgetState<ImplicitlyAnimatedWidget> createState() =>
     _SmoothLoadingIndicatorState();
}
```

implicit_loader.dart

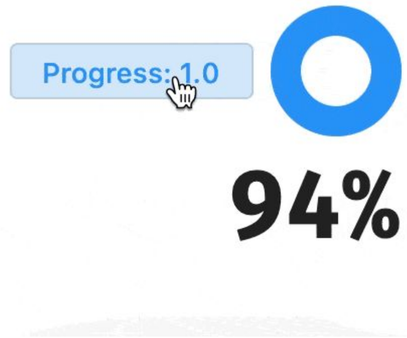
Custom ImplicitlyAnimatedWidget

```dart
class _SmoothLoadingIndicatorState extends AnimatedWidgetBaseState<SmoothLoadingIndicator> {
 Tween<double> _progress;

 @override
 Widget build(BuildContext context) {
   return CircularProgressIndicator(
     backgroundColor: widget.backgroundColor,
     color: widget.color,
     value: _progress.evaluate(animation), // use the provided animation on Tween
   );
 }

 @override
 void forEachTween(TweenVisitor<dynamic> visitor) {
   _progress = visitor(
       _progress, // current value - null initially
       (widget.progress).clamp(0.01, 1.0), // target value
       (dynamic value) => Tween<double>(begin: value as double), // Tween to animate the current value
    ) as Tween<double>;
 }
}
```


implicit_loader.dart

Custom ImplicitlyAnimatedWidget

- will animate when the value changes on rebuild
- you can use any type of value (int, string, custom class)
- useful when none of the **Animated*** widgets is enough but you don't want to mess with **AnimationController**



implicit_loader.dart

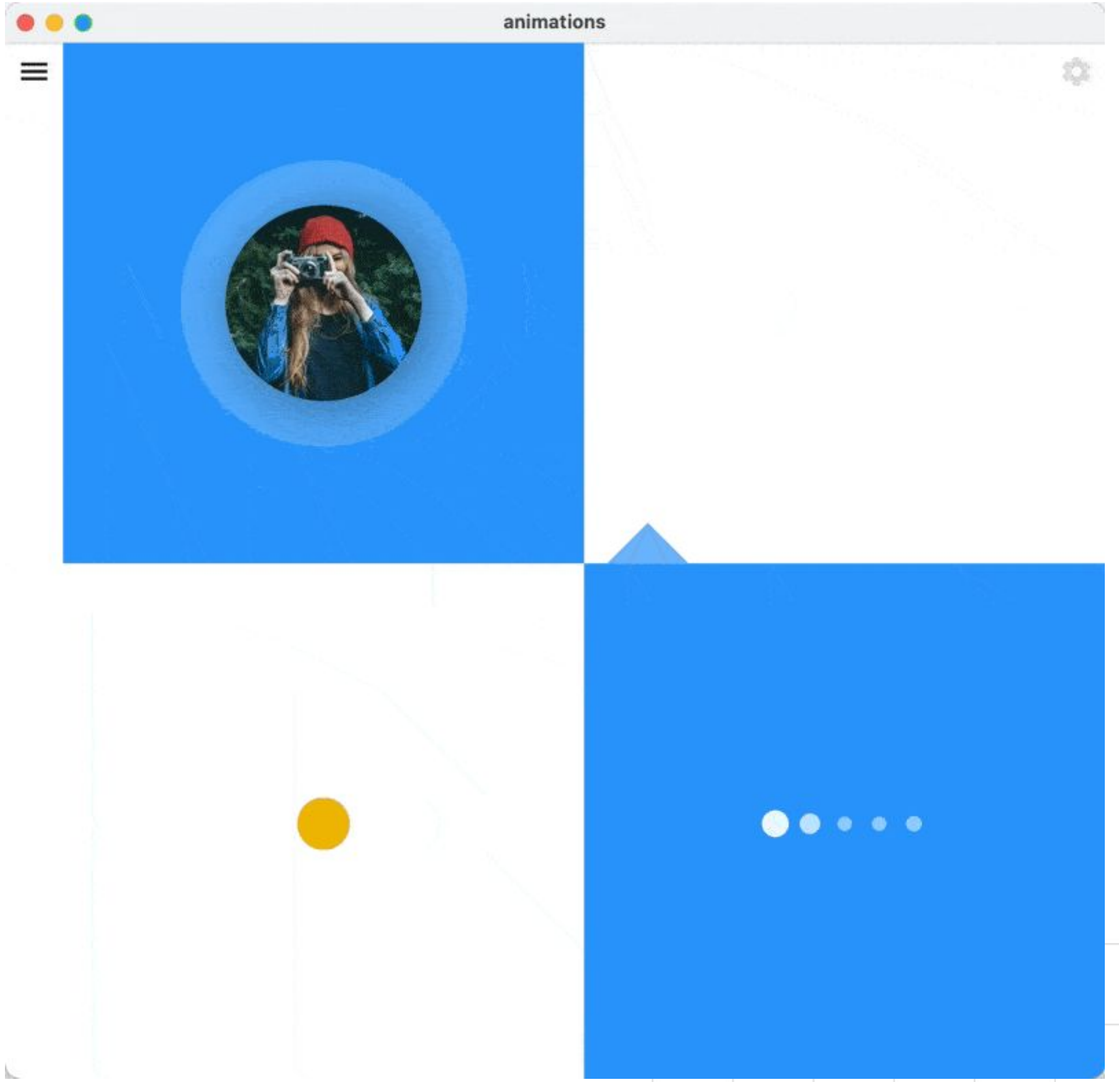
Explicit animations

Driving animations on your own

Using AnimationController or any other “engine” to drive the animation.

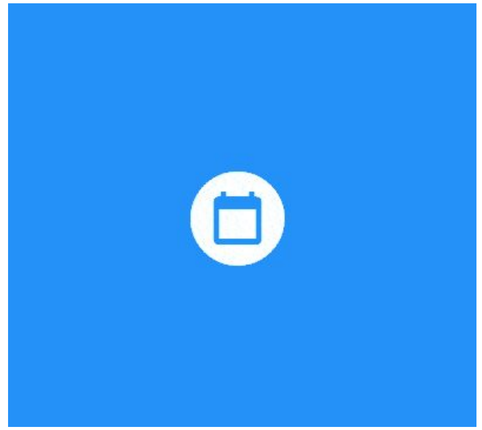
You can also use widgets like:

- ScaleTransition
- SizeTransition
- FadeTransition
- AnimatedWidget



roszkowski.dev/animations

Using AnimationController



```
class AnimatedRadiatingIcon extends StatefulWidget {
 @override
 _AnimatedRadiatingIconState createState() => _AnimatedRadiatingIconState();
}

class _AnimatedRadiatingIconState extends State<AnimatedRadiatingIcon>
   with TickerProviderStateMixin {
 AnimationController animationController;

 @override
 void initState() {
   super.initState();
   animationController = AnimationController(
     vsync: this,
     duration: Duration(seconds: 1),
   )
     ..forward()
     ..repeat(reverse: true);
 }

 @override
 void dispose() {
   animationController.dispose();
   super.dispose();
 }
```

```
@override
 Widget build(BuildContext context) {
   return AnimatedBuilder(
     animation: animationController,
     builder: (context, child) {
       return DecoratedBox(
         decoration: ShapeDecoration(
           color: Colors.white.withOpacity(0.5),
           shape: CircleBorder(),
         ),
         child: Padding(
           padding: EdgeInsets.all(8.0 * animationController.value),
           child: child, //it's the padding that is changing with time
         ),
       );
     },
     child: DecoratedBox(
       decoration: ShapeDecoration(
         color: Colors.white,
         shape: CircleBorder(),
       ),
       child: IconButton(
         onPressed: () {},
         color: Colors.blue,
         icon: Icon(Icons.calendar_today, size: 24),
       ),
     ),
   );
 }
}
```

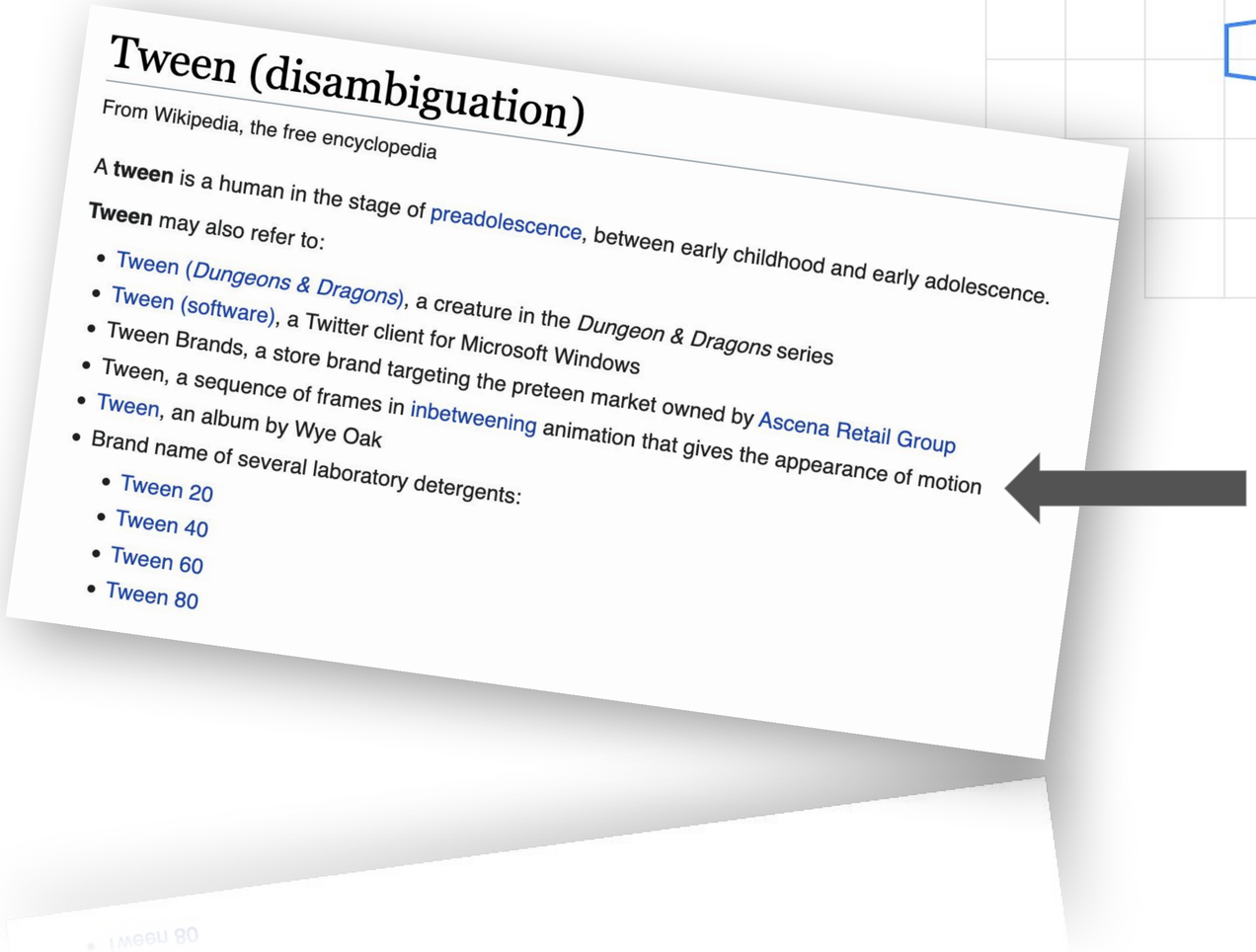
AnimatedBuilder

Triggers builder only when it’s needed

You can pass child property if subtree doesn’t need to be rebuilt

```
AnimatedBuilder(
  animation: animationController,
  builder: (context, child) {
    return Container(
      child: Padding(
        padding: EdgeInsets.all(8.0 * animationController.value),
        child: child,
      ),
    );
  },
  child: myChild,
),
```

Tweens



Tween (disambiguation)

From Wikipedia, the free encyclopedia

A **tween** is a human in the stage of preadolescence, between early childhood and early adolescence.

Tween may also refer to:

- Tween (*Dungeons & Dragons*), a creature in the *Dungeon & Dragons* series
- Tween (software), a Twitter client for Microsoft Windows
- Tween Brands, a store brand targeting the preteen market owned by Ascena Retail Group
- Tween, a sequence of frames in inbetweening animation that gives the appearance of motion
- Tween, an album by Wye Oak
- Brand name of several laboratory detergents:
  - Tween 20
  - Tween 40
  - Tween 60
  - Tween 80

Experts

Tweens

Mapping and interpolating AnimationController value

```dart
_animation = _controller.drive(
  Tween<Offset>(
    begin: const Offset(100.0, 50.0),
    end: const Offset(200.0, 300.0),
  ),
);
```

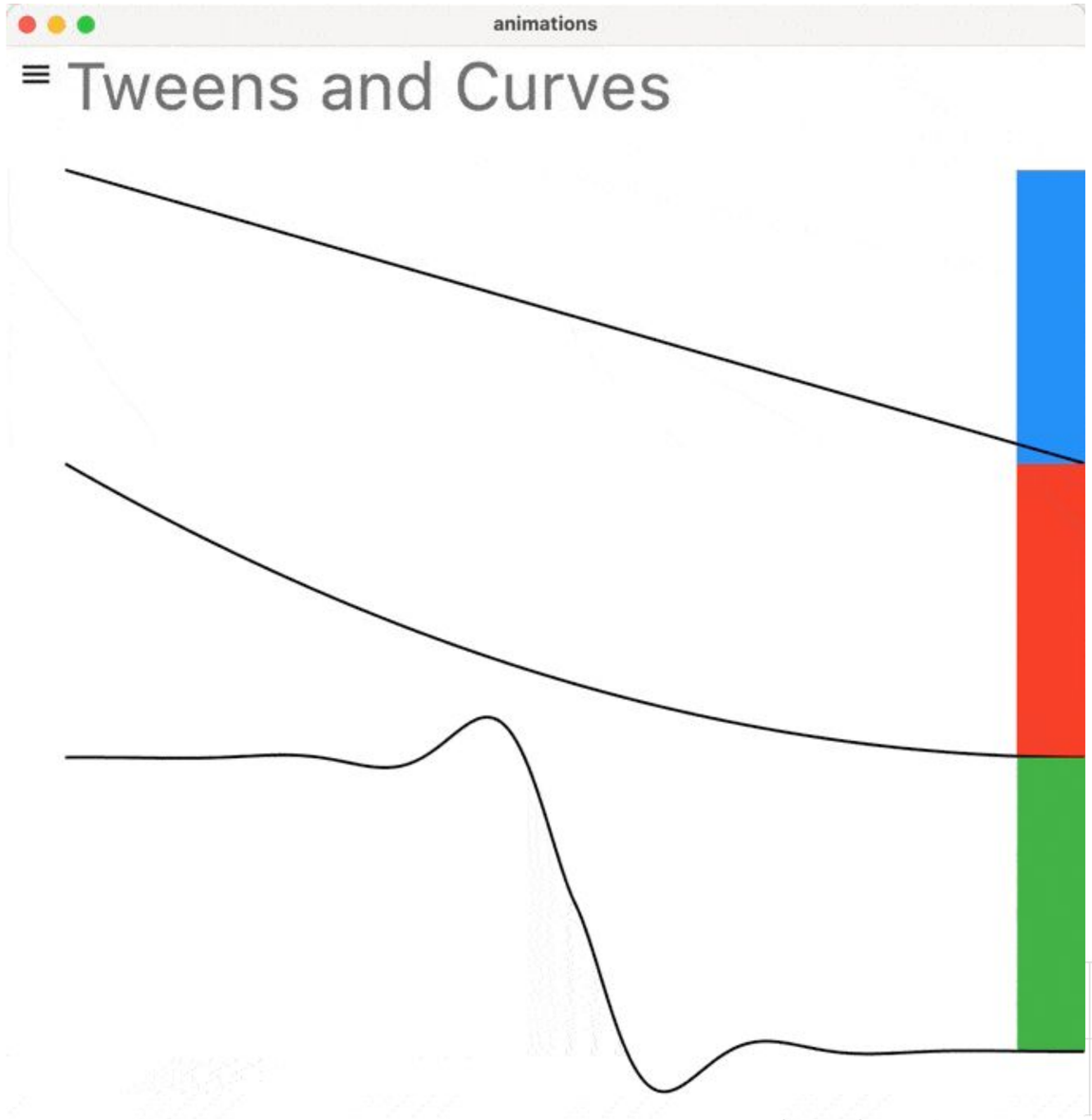
```dart
_animation = Tween<Offset>(
  begin: const Offset(100.0, 50.0),
  end: const Offset(200.0, 300.0),
).animate(_controller);
```

Both are equivalent and can be used
e.g. in AnimatedBuilder!

Tweens ctd.

Using custom curves with CurvedAnimation

```
Animation<Offset> getCurvedTween(
    AnimationController _controller, Curve curve) {
  return Tween<Offset>(
    begin: const Offset(-2.0, 0.0),
    end: const Offset(2.0, 0.0),
  ).animate(
    CurvedAnimation(
      parent: _controller,
      curve: curve,
    ),
  );
}
```



TweenAnimationBuilder

Tweens made easier

Easy interpolation between 2 values

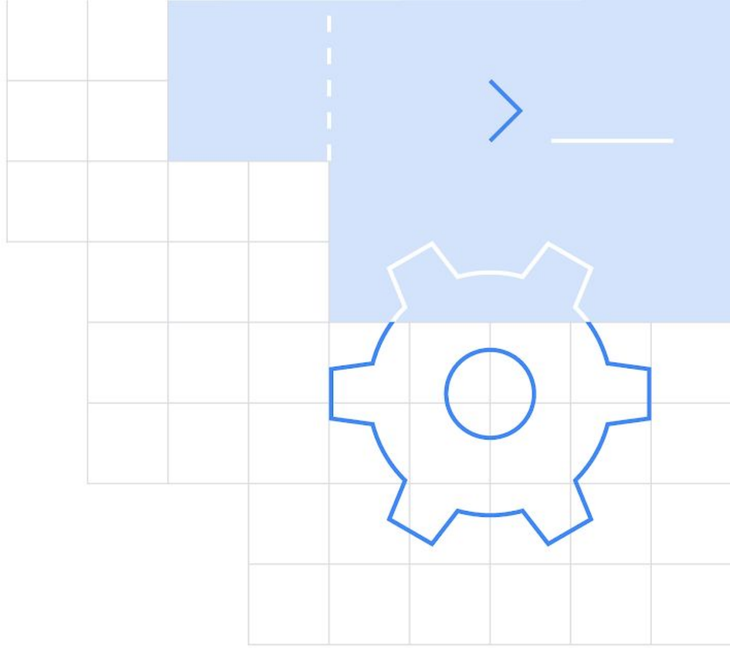
Tween can be anything:

- number
- Offset
- string...

Tween can be mutated!

```
class TweenAnimationDemo extends StatelessWidget {
 const TweenAnimationDemo({Key key, this.scale}) : super(key: key);

 final double scale;

 @override
 Widget build(BuildContext context) {
   return Center(
     child: TweenAnimationBuilder(
       duration: Duration(seconds: 2),
       tween: Tween<double>(begin: 0.0, end: scale ?? 1.0),
       curve: Curves.easeInOut,
       builder: (context, value, child) {
         return Transform.scale(
           scale: value,
           child: child,
         );
       },
       child: Text('I\'m child'),
     ),
   );
 }
}
```

Experts



More concepts

Staggered animation

TickerProvider and Ticks

Spring Simulation

Experts

Staggered animations

Using single AnimationController for multiple animation steps

Orchestrate animations with chained Tweens

Improve readability by extracting common “animation classes”

```dart
class HomePageAnimatedBuilder extends StatelessWidget {
 const HomePageAnimatedBuilder({
   Key key,
   this.builder,
   this.animation,
   this.child,
 }) : super(key: key);

 final MyTransitionBuilder builder;
 final Listenable animation;
 final Widget child;

 @override
 Widget build(BuildContext context) {
   return AnimatedBuilder(
     animation: animation,
     builder: (context, child) {
       return builder(context, child, HomePageEnterAnimation(animation));
     },
     child: child,
   );
 }
}
```

Delegate the “ordinary” AnimationController to your custom helper widget

```dart
typedef MyTransitionBuilder = Widget Function(
 BuildContext context,
 Widget child,
 HomePageEnterAnimation animation,
);
```

```dart
class HomePageEnterAnimation {
 HomePageEnterAnimation(this.controller)
     : headerOpacity = Tween<double>(begin: 0, end: 1.0).animate(
         CurvedAnimation(
           parent: controller,
           curve: Interval(0, 0.2, curve: Curves.easeIn),
         ),
       ),
       row1Offset =
           Tween<Offset>(begin: Offset(0, 5), end: Offset.zero).animate(
         CurvedAnimation(
           parent: controller,
           curve: Interval(0.1, 0.25, curve: Curves.easeOut),
         ),
       ),
       row2Offset =
           Tween<Offset>(begin: Offset(0, 5), end: Offset.zero).animate(
         CurvedAnimation(
           parent: controller,
           curve: Interval(0.15, 0.30, curve: Curves.easeOut),
         ),
       ),
       );

 final AnimationController controller;
 final Animation<double> headerOpacity;
 final Animation<Offset> row1Offset;
 final Animation<Offset> row2Offset;
}
```

Extract Tweens logic into the separate animation class

Each Tween needs to be used with Interval within 0.0 and 1.0

```dart
class PageLayout extends StatelessWidget {
 final AnimationController controller;

 @override
 Widget build(BuildContext context) {
   return HomePageAnimatedBuilder(
     animation: controller,
     builder: (context, child, animation) {
       return ListView(
         children: [
           Gap(16),
           Opacity(
             opacity: animation.headerOpacity.value,
             child: _Header(),
           ),
           SlideTransition(
             position: animation.row1Offset,
             child: _Row1(),
           ),
           SlideTransition(
             position: animation.row2Offset,
             child: _Row2(),
           ),
         ],
       );
     },
   );
 }
```

Use meaningful names in your layout code

Motion design timeline

https://uxmisfit.com/2017/04/23/motion-design-specs-how-to-present-animations-and-interactions-to-developers/

Low level animation control

```
class BoidSimulation extends ChangeNotifier {
 BoidSimulation(this.vsync) {
   _ticker = vsync.createTicker(_onEachTick)..start();
 }
 final TickerProvider vsync;
 late Ticker _ticker;

 void _onEachTick(Duration deltaTime) {
      ...
      notifyListeners();
 }
...
```

ChangeNotifier can be used in AnimatedBuilder

Full source code at github.com/orestesgaolin/animations_samples

TickerProvider and Ticker

TickerProvider

Any class that notifies about new frame triggered e.g. `TickerProviderStateMixin`.

Ticker

Calls its callback on every new frame.

Simple physics simulations in Flutter

SpringSimulation (physics.dart)

GravitySimulation

FrictionSimulation

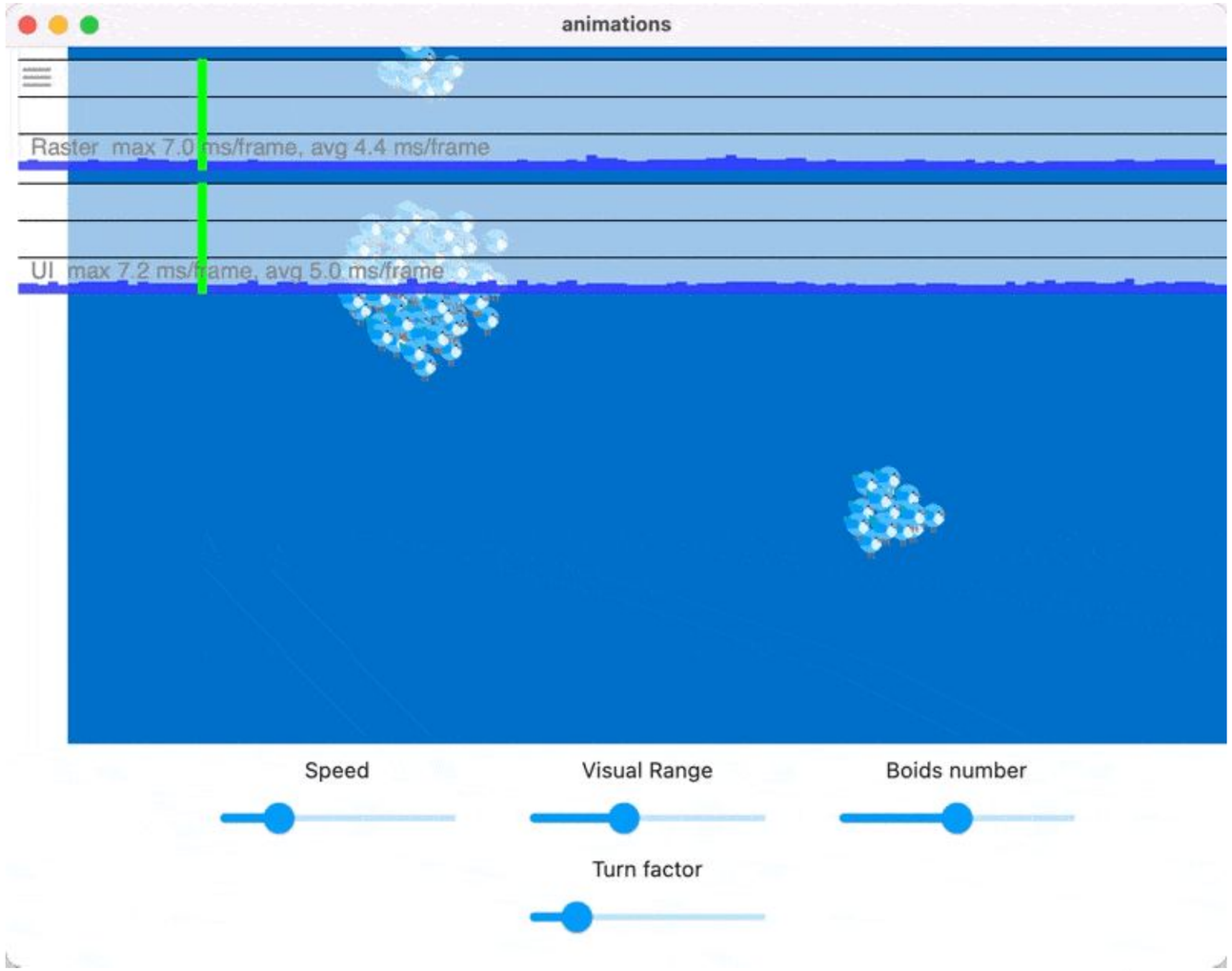
ScrollSpringSimulation...


```
simulation = SpringSimulation(
  SpringDescription(
    mass: 2,
    stiffness: 100,
    damping: 1,
  ),
  0.0,
  600.0,
  10,
);

controller = AnimationController(
  vsync: this,
  upperBound: 1500,
);

controller.animateWith(simulation);
```

Experts

Performance measurements

- Performance Overlay
- Profiling Tools
- FPS widget



Let’s Recap

- We have a lot of **Implicitly** animated widgets at our disposal
- Don’t hesitate to use **AnimationController**
- You can use **TweenAnimationBuilder** for nice transitions
- You can create your own controller using **TickerProvider**
- Simulate the physics with **SpringSimulation, GravitySimulation** or create your own

I want to learn more!

Code and slides: github.com/orestesgaolin/animations_samples

Web demo: roszkowski.dev/animations

Talks

The little things: Becoming the mythical designer-developer

Animations (Package of the Week)

Flutter Europe: Animations in Flutter done right

Packages

simple_animations

animations

animator

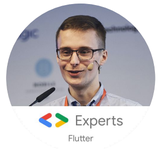
flutter_staggered_animations

sprung

funvas

Experts

Thank You!



Dominik Roszkowski
@OrestesGaolin
roszkowski.dev

Companion app code and slides:
github.com/orestesgaolin/animations_samples