

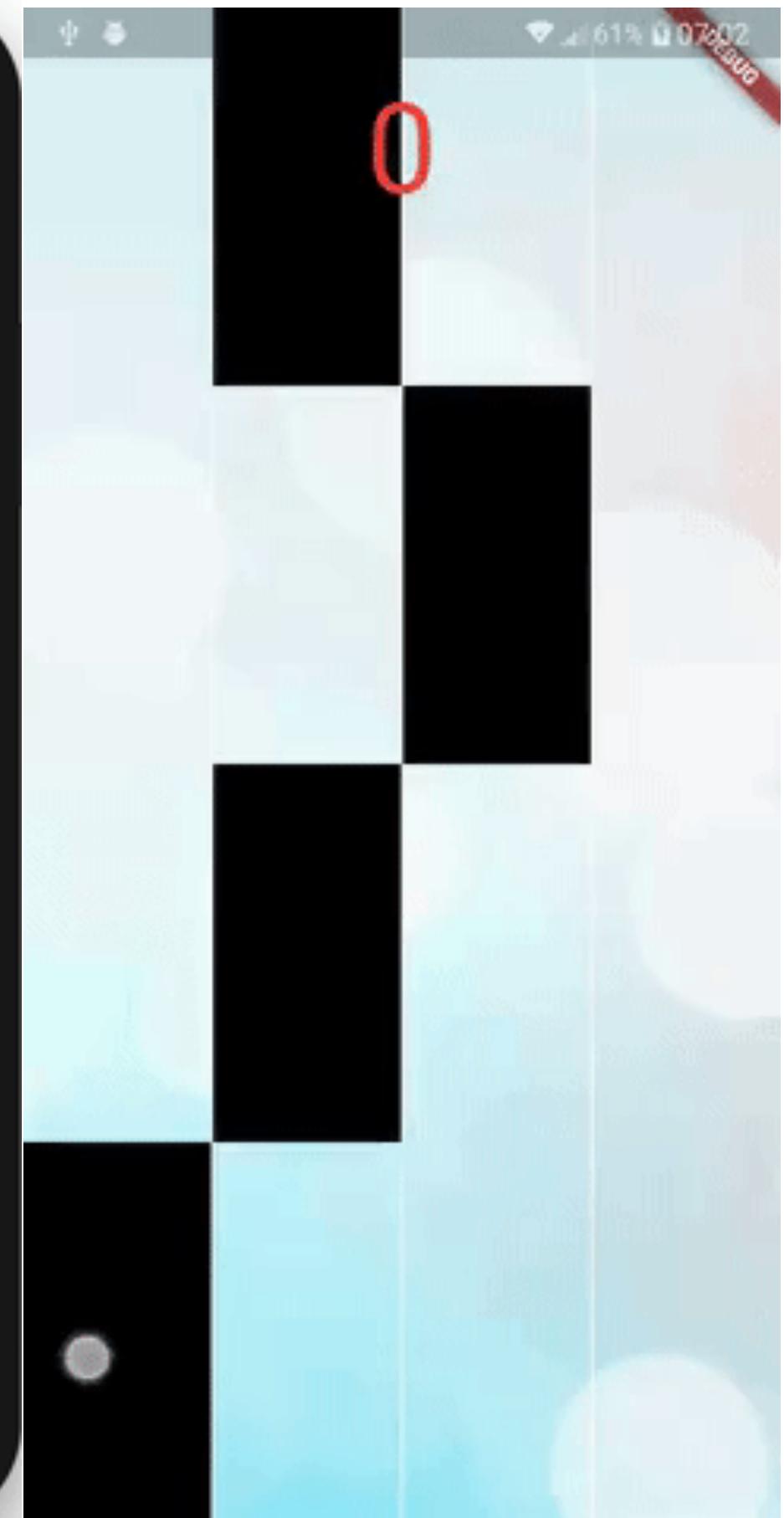
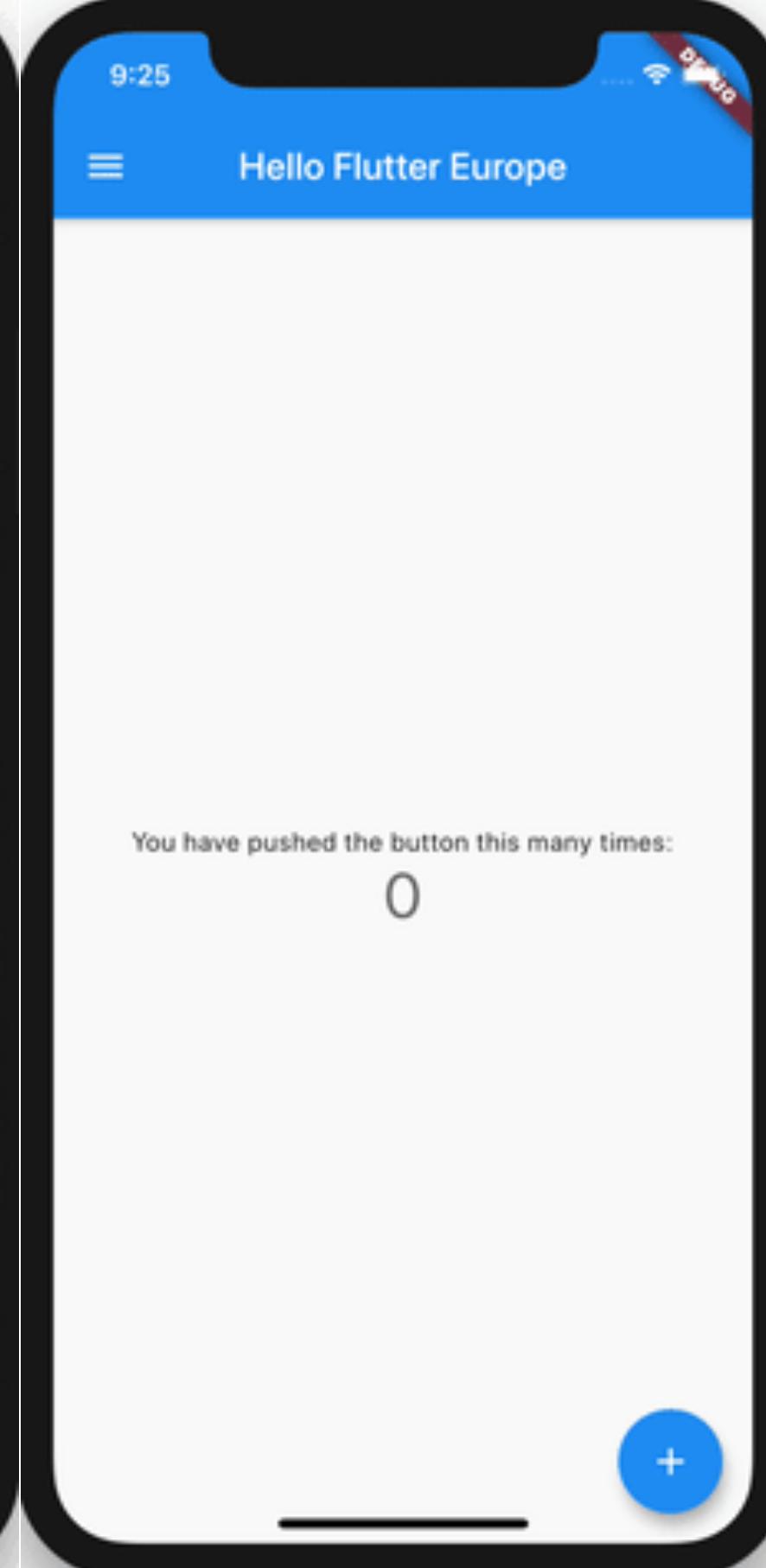
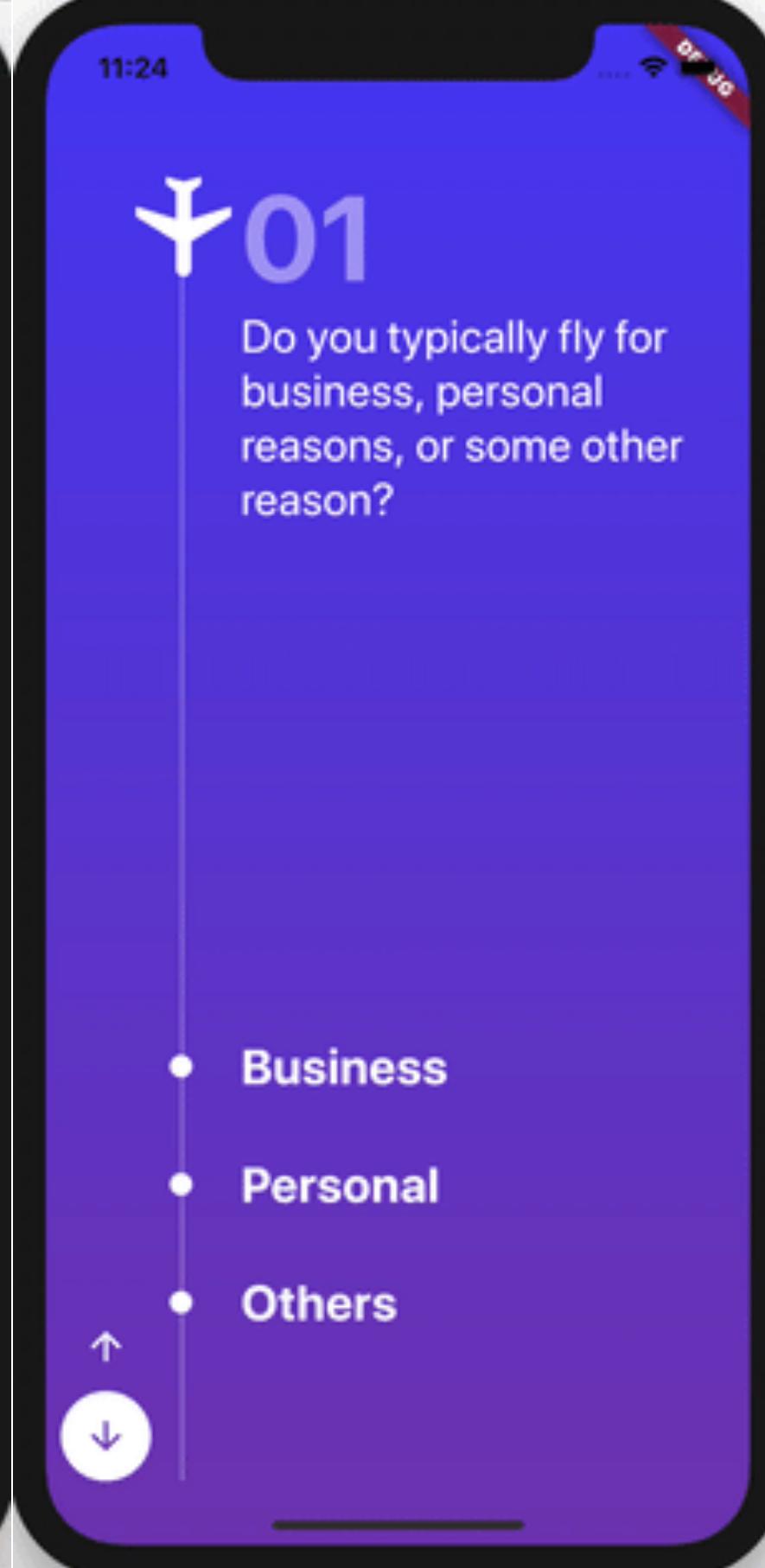
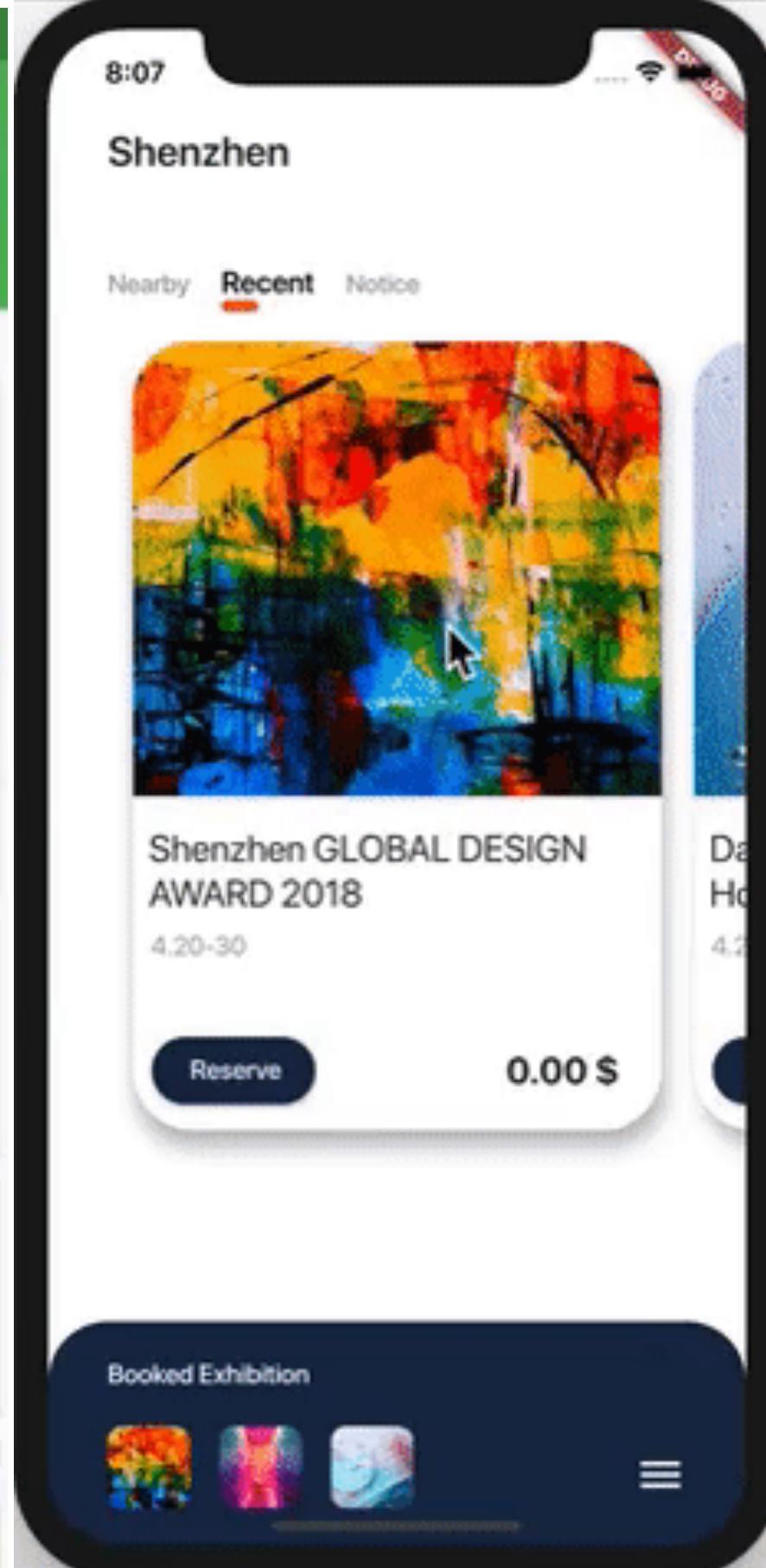
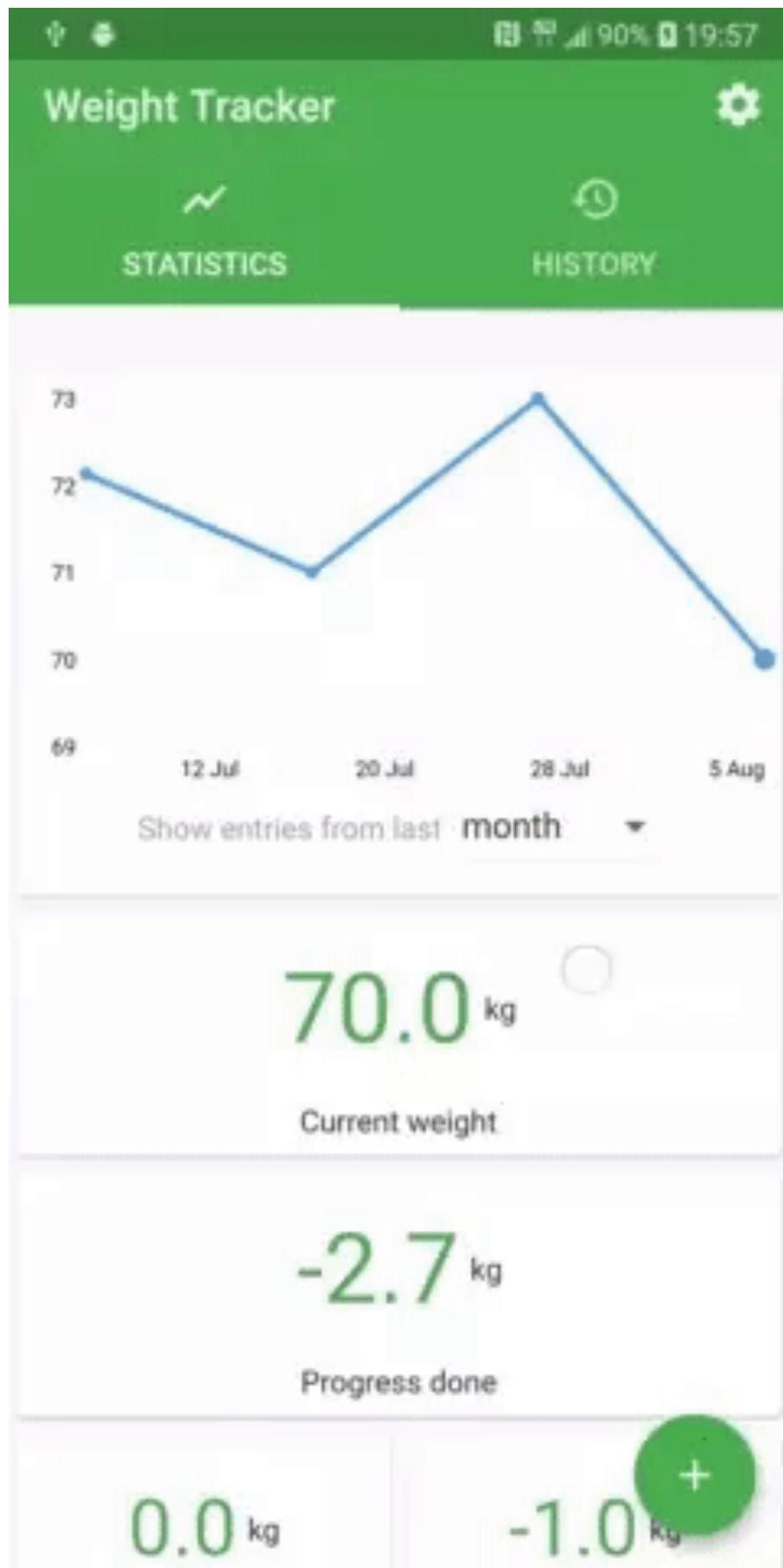
# Animations in Flutter

Marcin Szałek  
@marcin\_szalek

Warsaw  
January 23, 2023

# Who am I?

- mgr inż. Marcin Szałek
- Flutter developer since Alpha (May 2017)
- Co-organizer and speaker at Flutter Europe
- Currently Mobile Lead at Cheddar
- Recently developer at FIFA World Cup Qatar 2022 

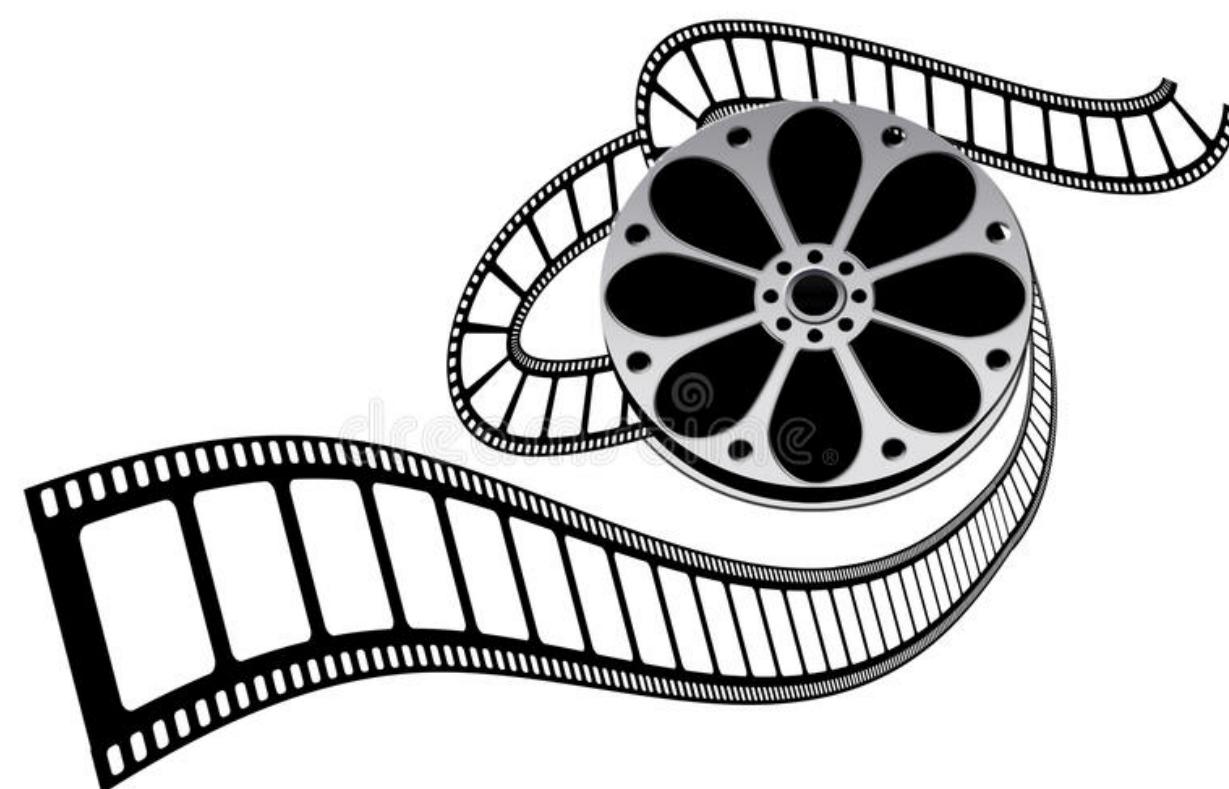


# What are animations?

It's the motion on the screen

# And what is motion?

Many still images shown in quick successions



# Why animations?

## Functional

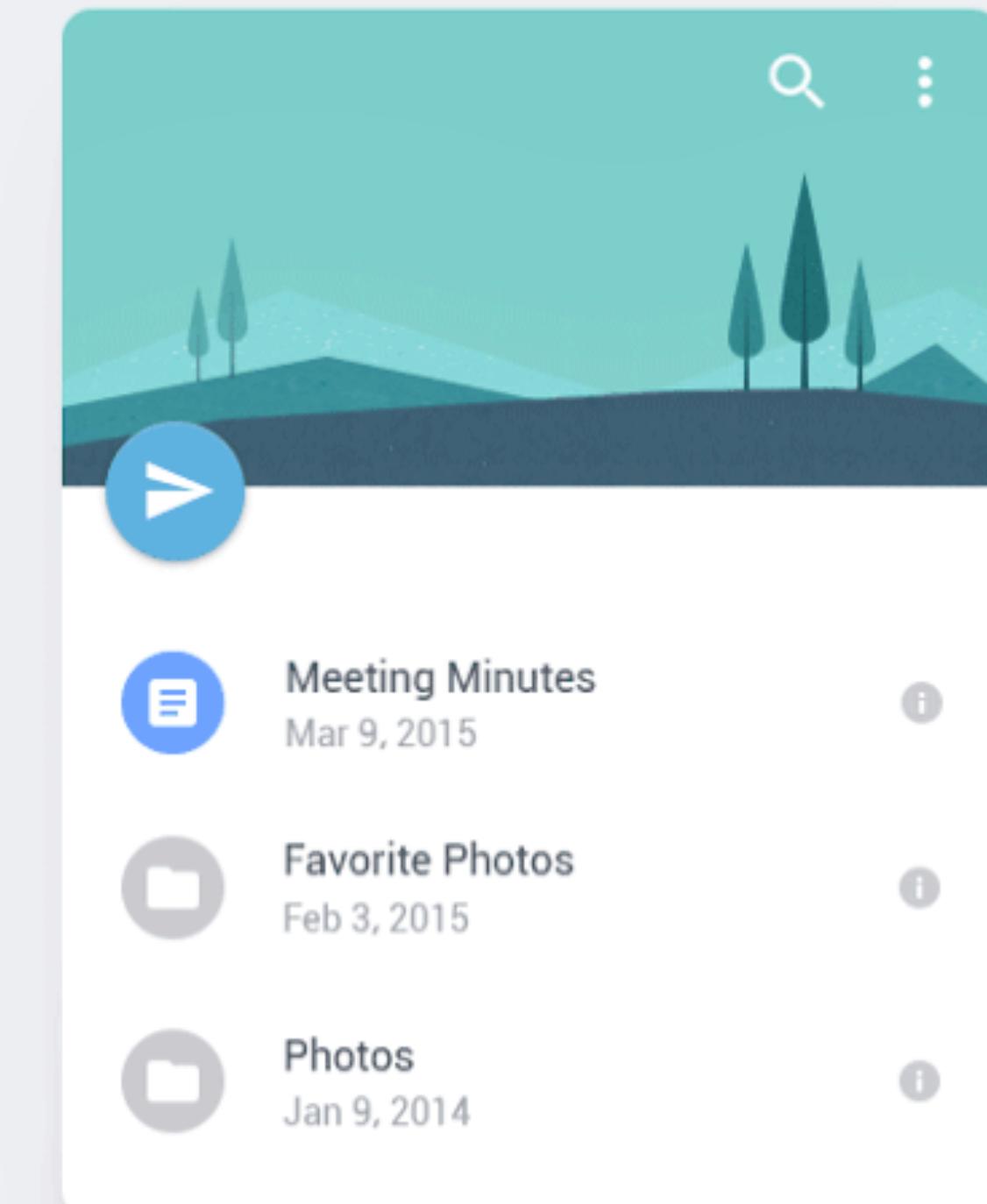
- Meaningful transitions
- Provide visual feedback
- Show system status
- Help user get started



# Why animations?

## Delightful

- To demonstrate polish
- To bring personality and humanity to the app
- To entertain users



# Implicit vs Explicit Animations

# Implicit animations

AnimatedContainer

AnimatedAlign

AnimatedPositioned

AnimatedPositionedDirectional

AnimatedOpacity

AnimatedDefaultTextStyle

AnimatedPhysicalModel

AnimatedPadding

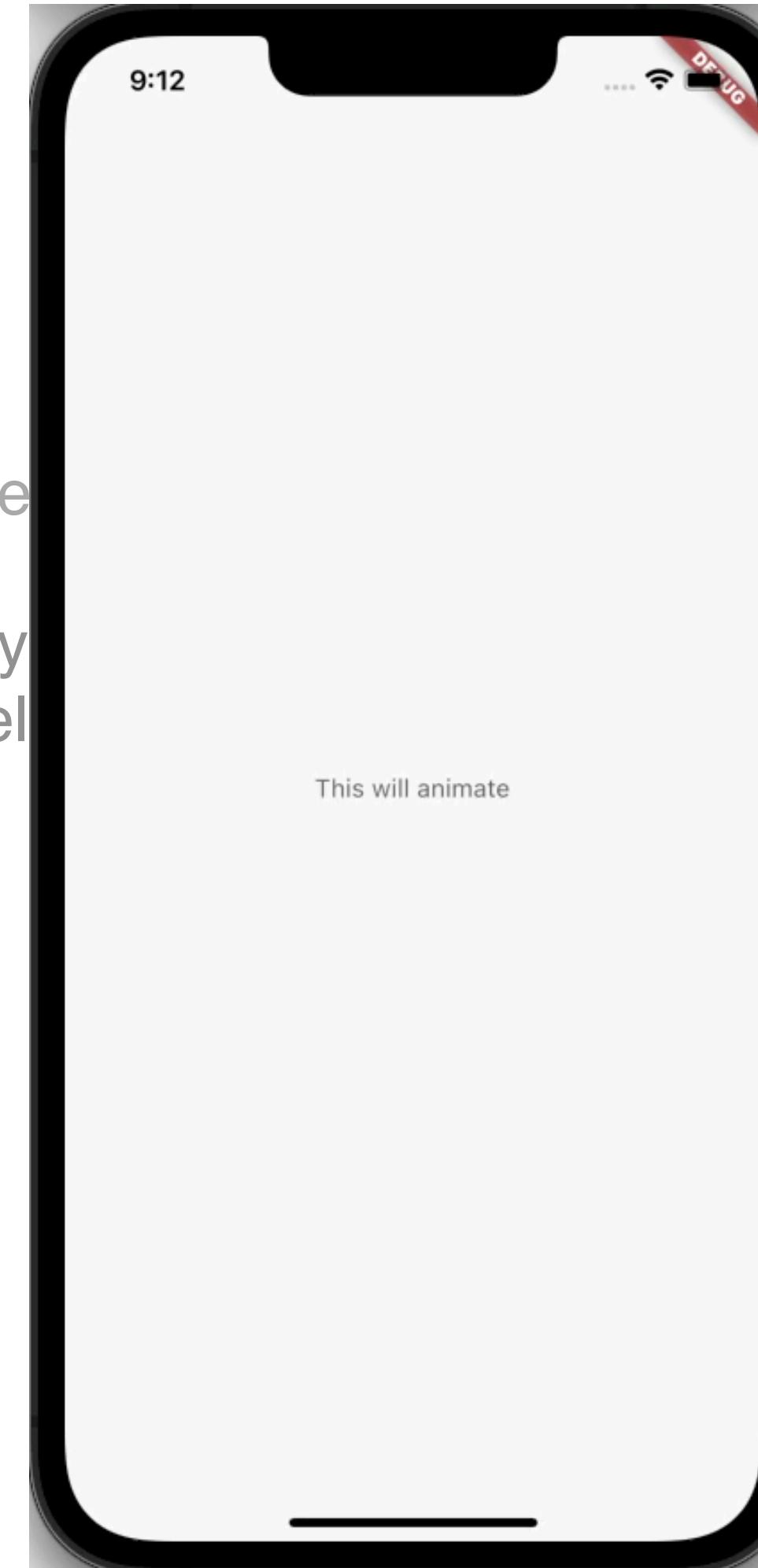
AnimatedTheme

AnimatedCrossFade

AnimatedSwitcher

# Implicit animations

AnimatedContainer  
AnimatedAlign  
AnimatedPositioned  
AnimatedPositionedDirection  
**AnimatedOpacity**  
AnimatedDefaultTextStyle  
AnimatedPhysicalModel  
AnimatedPadding  
AnimatedTheme  
AnimatedCrossFade  
AnimatedSwitcher



```
class _PageState extends State<Page> {  
    double _opacity = 1.0;  
  
    @override  
    Widget build(BuildContext context) {  
        return AnimatedOpacity(  
            opacity: _opacity,  
            duration: const Duration(milliseconds: 300),  
            child: const Text('This will animate'),  
        );  
    }  
}
```

# Implicit animations

AnimatedContainer

AnimatedAlign

AnimatedPositioned

AnimatedPositionedDirectional

AnimatedOpacity

AnimatedDefaultTextStyle

AnimatedPhysicalModel

**AnimatedPadding**

AnimatedTheme

AnimatedCrossFade

AnimatedSwitcher

```
class _PageState extends State<Page> {
  EdgeInsetsGeometry _padding = const EdgeInsets.all(8);

  @override
  Widget build(BuildContext context) {
    return AnimatedPadding(
      padding: _padding,
      duration: const Duration(milliseconds: 300),
      child: const Text('This will animate'),
    );
  }
}
```

# Implicit animations

## **AnimatedContainer**

AnimatedAlign  
AnimatedPositioned  
AnimatedPositionedDirectional  
AnimatedOpacity  
AnimatedDefaultTextStyle  
AnimatedPhysicalModel  
AnimatedPadding  
AnimatedTheme  
AnimatedCrossFade  
AnimatedSwitcher

```
class _PageState extends State<Page> {
    bool _isActive = true;

    @override
    Widget build(BuildContext context) {
        return AnimatedContainer(
            height: _isActive ? 300 : 200,
            color: _isActive ? Colors.green : Colors.yellow,
            duration: Duration(milliseconds: 300),
            child: const Text('This will animate'),
        );
    }
}
```

# Implicit animations

AnimatedContainer

AnimatedAlign

AnimatedPositioned

AnimatedPositionedDirectional

AnimatedOpacity

AnimatedDefaultTextStyle

AnimatedPhysicalModel

AnimatedPadding

AnimatedTheme

**AnimatedCrossFade**

**AnimatedSwitcher**

```
class _PageState extends State<Page> {
    bool _isActive = true;

    @override
    Widget build(BuildContext context) {
        return AnimatedSwitcher(
            duration: Duration(milliseconds: 300),
            child: _isActive
                ? const Text('Active')
                : const CircularProgressIndicator(),
        );
    }
}
```

# Explicit animations

# AnimationController



# AnimationController

```
class _MyPageState extends State<MyPage> with SingleTickerProviderStateMixin {
    late final AnimationController _animationController;

    @override
    void initState() {
        super.initState();
        _animationController = AnimationController(vsync: this);
    }

    @override
    void dispose() {
        _animationController.dispose();
        super.dispose();
    }

    @override
    Widget build(BuildContext context) {
        return ...;
    }
}
```

# AnimationController

# AnimationController

## Bad, ugly and good

# AnimationController

## Bad, ugly and good

```
class _MyPageState extends State<MyPage> with SingleTickerProviderStateMixin {
    late final AnimationController _animationController;

    @override
    void initState() {
        super.initState();
        _animationController = AnimationController(
            vsync: this,
            duration: const Duration(seconds: 2),
        );
        _animationController.repeat();
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Center(
                child: Transform.scale(
                    scale: _animationController.value,
                    child: Container(
                        color: Colors.red,
                        height: 300,
                        width: 300,
                    ),
                ),
            ),
        );
    }
}
```



# AnimationController

## Bad, ugly and good

```
class _MyPageState extends State<MyPage> with SingleTickerProviderStateMixin {
    late final AnimationController _animationController;

    @override
    void initState() {
        super.initState();
        _animationController = AnimationController(
            vsync: this,
            duration: const Duration(seconds: 2),
        );
        _animationController.addListener(() => setState(() {}));
        _animationController.repeat();
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Center(
                child: Transform.scale(
                    scale: _animationController.value,
                    child: Container(
                        color: Colors.red,
                        height: 300,
                        width: 300,
                    ),
                ),
            ),
        );
    }
}
```



# AnimationController

```
class _MyPageState extends State<MyPage> with SingleTickerProviderStateMixin {
    late final AnimationController _animationController;

    @override
    void initState() {
        super.initState();
        _animationController = AnimationController(
            vsync: this,
            duration: const Duration(seconds: 2),
        );
        _animationController.repeat();
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Center(
                child: AnimatedBuilder(
                    animation: _animationController,
                    builder: (context, child) => Transform.scale(
                        scale: _animationController.value,
                        child: child,
                    ),
                    child: Container(
                        color: Colors.red,
                        height: 300,
                        width: 300,
                    ),
                ),
            ),
        );
    }
}
```



# AnimationController

```
class _MyPageState extends State<MyPage> with SingleTickerProviderStateMixin {
    late final AnimationController _animationController;

    @override
    void initState() {
        super.initState();
        _animationController = AnimationController(
            vsync: this,
            duration: const Duration(seconds: 2),
        );
        _animationController.repeat();
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Center(
                child: AnimatedBuilder(
                    animation: _animationController,
                    builder: (context, child) => Transform.scale(
                        scale: _animationController.value,
                        child: child,
                    ),
                    child: Container(
                        color: Colors.red,
                        height: 300,
                        width: 300,
                    ),
                ),
            ),
        );
    }
}
```



# AnimationController

```
class _MyPageState extends State<MyPage> with SingleTickerProviderStateMixin {
    late final AnimationController _animationController;
    late final Animation<double> _animation;

    @override
    void initState() {
        super.initState();
        _animationController = AnimationController(
            vsync: this,
            duration: const Duration(seconds: 2),
        );
        _animationController.repeat();
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Center(
                child: AnimatedBuilder(
                    animation: _animationController,
                    builder: (context, child) => Transform.scale(
                        scale: _animationController.value,
                        child: child,
                    ),
                    child: Container(
                        color: Colors.red,
                        height: 300,
                        width: 300,
                    ),
                ),
            ),
        );
    }
}
```

3:55



```
class _MyPageState extends State<MyPage> with SingleTickerProviderStateMixin {
    late final AnimationController _animationController;
    late final Animation<double> _animation;

    @override
    void initState() {
        super.initState();
        _animationController = AnimationController(
            vsync: this,
            duration: const Duration(seconds: 2),
        );
        _animation = CurvedAnimation(
            parent: _animationController,
            curve: Curves.elasticOut,
        );
        _animationController.repeat();
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Center(
                child: AnimatedBuilder(
                    animation: _animationController,
                    builder: (context, child) => Transform.scale(
                        scale: _animationController.value,
                        child: child,
                    ),
                    child: Container(
                        color: Colors.red,
                        height: 300,
                        width: 300,
                    ),
                ),
            ),
        );
    }
}
```

3:55



```
class _MyPageState extends State<MyPage> with SingleTickerProviderStateMixin {
    late final AnimationController _animationController;
    late final Animation<double> _animation;

    @override
    void initState() {
        super.initState();
        _animationController = AnimationController(
            vsync: this,
            duration: const Duration(seconds: 2),
        );
        _animation = CurvedAnimation(
            parent: _animationController,
            curve: Curves.elasticOut,
        );
        _animationController.repeat();
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Center(
                child: AnimatedBuilder(
                    animation: _animationController,
                    builder: (context, child) => Transform.scale(
                        scale: _animation.value,
                        child: child,
                    ),
                    child: Container(
                        color: Colors.red,
                        height: 300,
                        width: 300,
                    ),
                    ),
                    ),
                    ),
                    );
    }
}
```

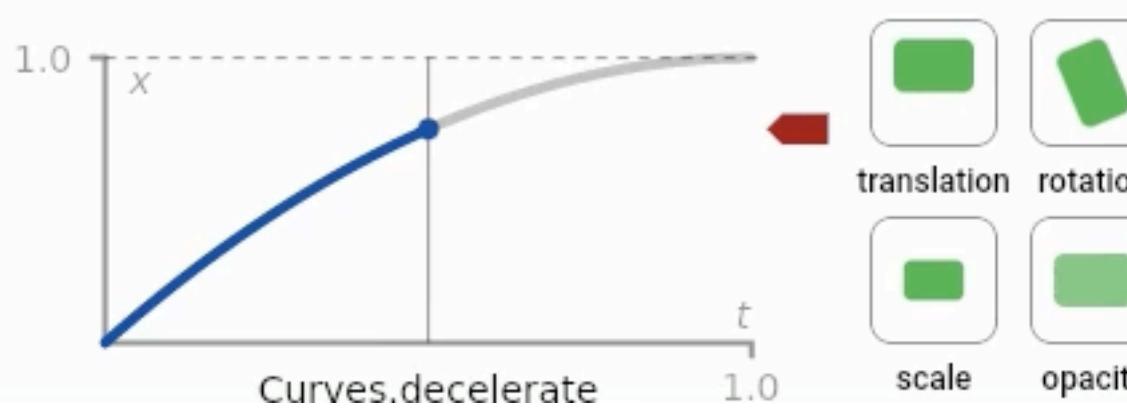
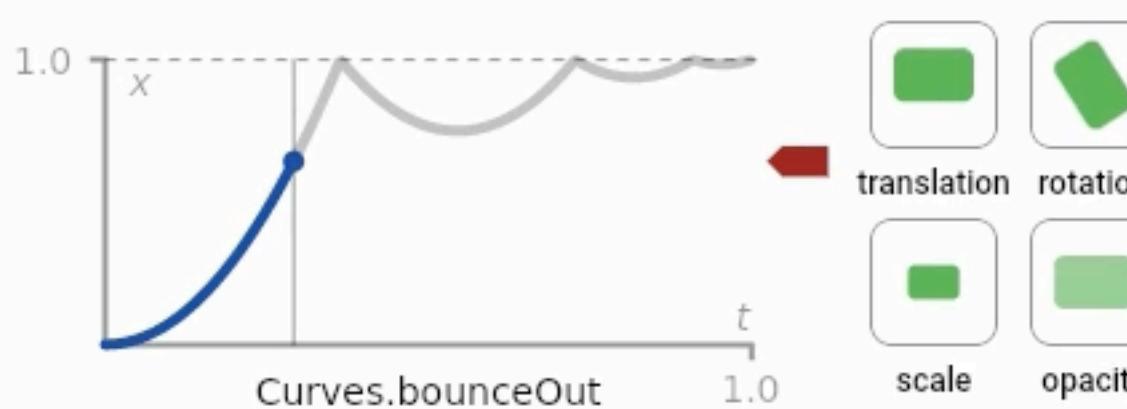
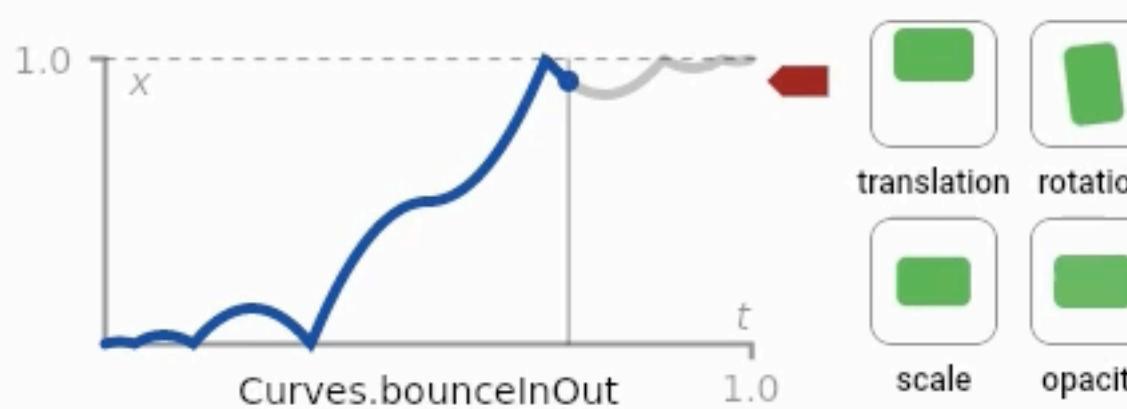
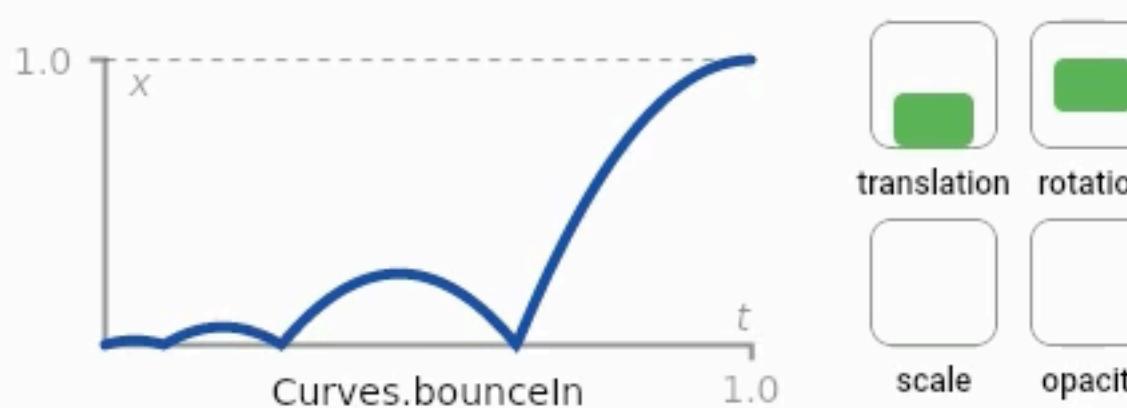
3:55



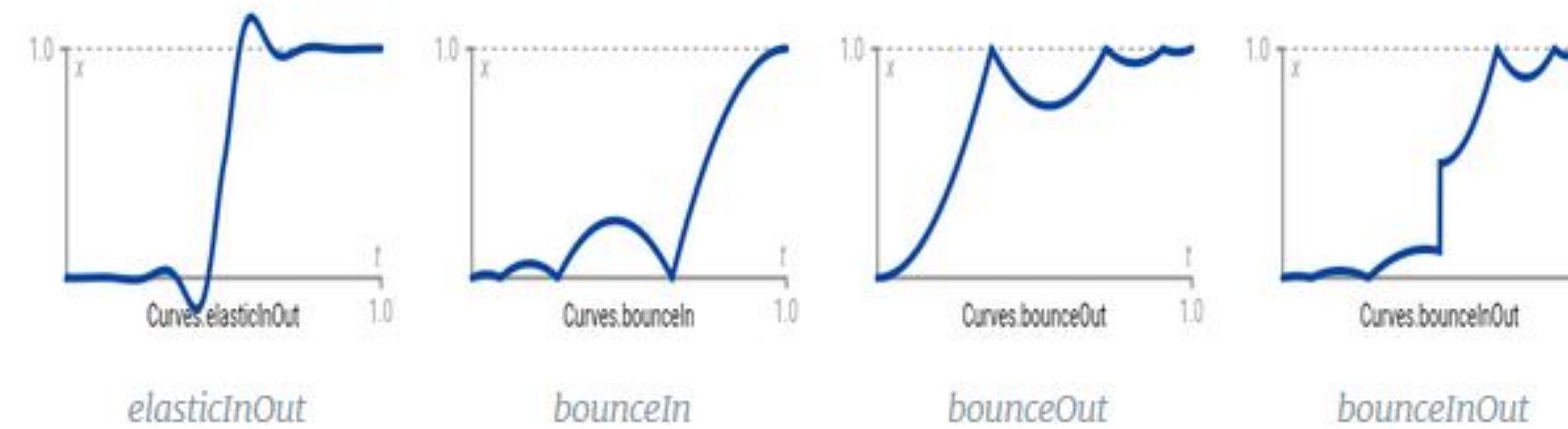
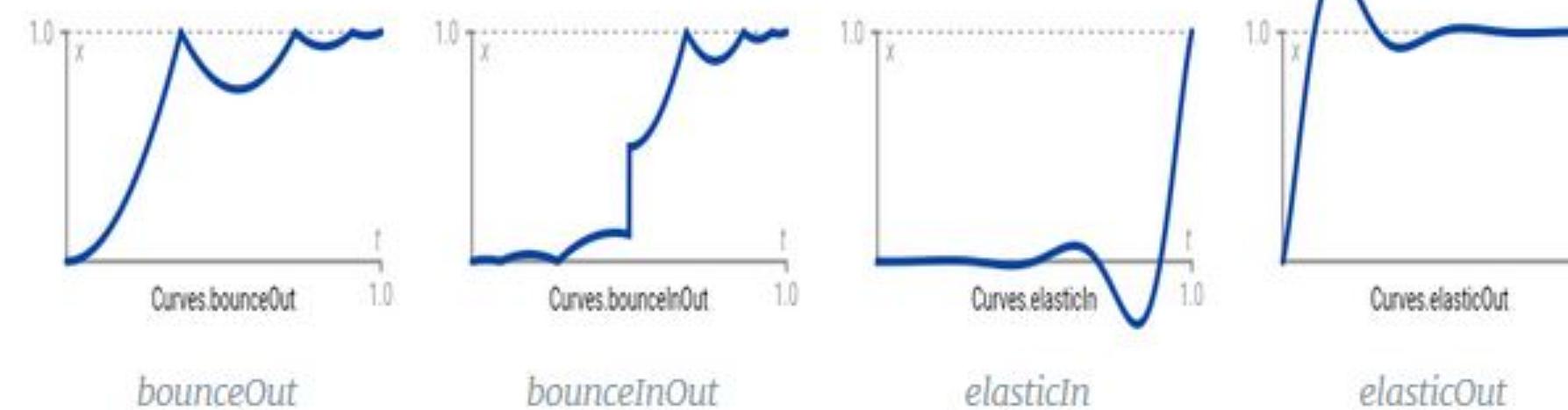
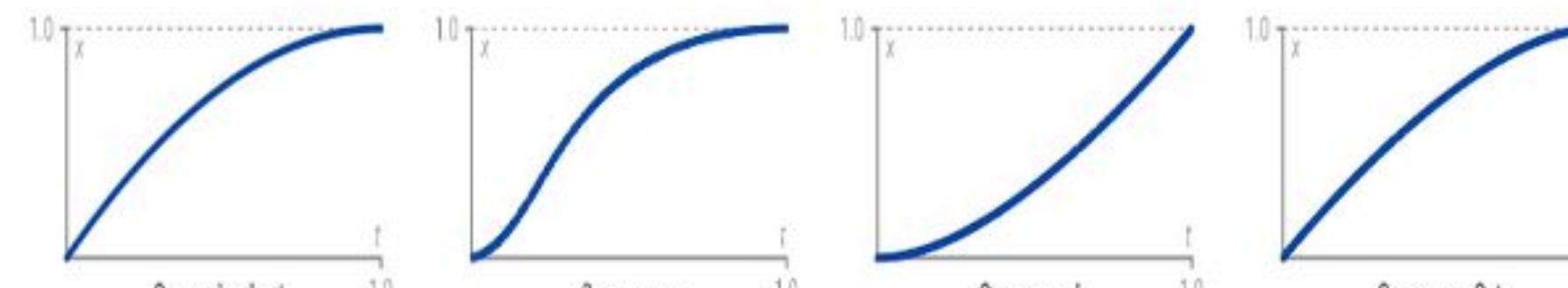
## Curves class

Null safety

A collection of common animation curves.



# Animation curves



# What can be animated?



# Tweens



```
@override  
void initState() {  
    super.initState();  
    _animationController = AnimationController(  
        vsync: this,  
        duration: const Duration(seconds: 2),  
    );  
    _animation = Tween(begin: 0.5, end: 2.0).animate(_animationController);  
    _animationController.repeat();  
}
```



# Tweens



```
@override  
void initState() {  
    super.initState();  
    _animationController = AnimationController(  
        vsync: this,  
        duration: const Duration(seconds: 5),  
    );  
    _animation = IntTween(begin: 0, end: 100).animate(CurvedAnimation(  
        parent: _animationController,  
        curve: Curves.easeOutExpo,  
    ));  
    _animationController.repeat();  
}
```

0

# Tweens



```
@override  
void initState() {  
    super.initState();  
    _animationController = AnimationController(  
        vsync: this,  
        duration: const Duration(seconds: 5),  
    );  
    _animation = IntTween(begin: 0, end: 100).animate(CurvedAnimation(  
        parent: _animationController,  
        curve: Curves.easeOutExpo,  
    ));  
    _animationController.repeat();  
}
```

0

```
@override  
Widget build(BuildContext context) {  
    return Scaffold(  
        body: Center(  
            child: AnimatedBuilder(  
                animation: _animationController,  
                builder: (context, child) => Text(  
                    _animation.value.toString(),  
                    style: const TextStyle(fontSize: 50),  
                ),  
            ),  
        ),  
    );  
}
```



# Tweens



```
_animation = ColorTween(  
  begin: Colors.yellow,  
  end: Colors.orange,  
).animate(_animationController);
```

```
_animation = SizeTween(  
  begin: const Size(200, 50),  
  end: const Size(50, 200),  
).animate(_animationController);
```

```
_animation = BorderRadiusTween(  
  begin: BorderRadius.circular(0),  
  end: BorderRadius.circular(80),  
).animate(_animationController);
```



# Tweens

1:27

```
class _MyPageState extends State<MyPage> with SingleTickerProviderStateMixin {
  late final AnimationController _animationController;
  late final Animation _borderAnimation;
  late final Animation _sizeAnimation;
  late final Animation _colorAnimation;

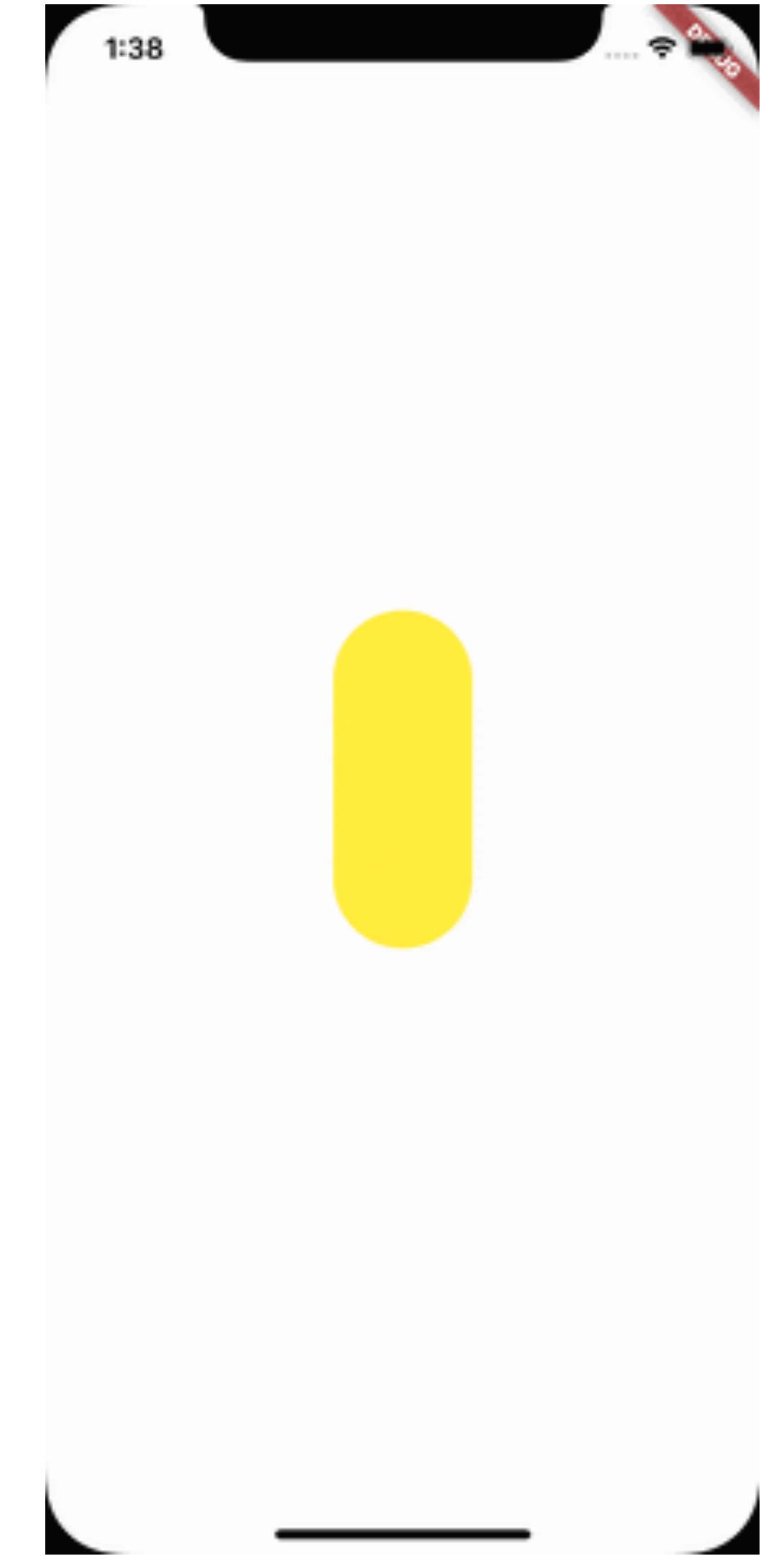
  @override
  void initState() {
    super.initState();
    _animationController = AnimationController(
      vsync: this,
      duration: const Duration(seconds: 1),
    );
    _borderAnimation = BorderRadiusTween(
      begin: BorderRadius.circular(0),
      end: BorderRadius.circular(80),
    ).animate(_animationController);
    _sizeAnimation = SizeTween(
      begin: Size(200, 50),
      end: Size(50, 200),
    ).animate(_animationController);
    _colorAnimation = ColorTween(
      begin: Colors.yellow,
      end: Colors.red,
    ).animate(_animationController);
    _animationController.repeat(reverse: true);
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: AnimatedBuilder(
          animation: _animationController,
          builder: (context, child) => Container(
            decoration: BoxDecoration(
              borderRadius: _borderAnimation.value,
              color: _colorAnimation.value,
            ),
            width: _sizeAnimation.value.width,
            height: _sizeAnimation.value.height,
          ),
        ),
      ),
    );
  }
}
```



# Staggered Animations

```
_borderAnimation = BorderRadiusTween(  
    begin: BorderRadius.circular(0),  
    end: BorderRadius.circular(80),  
).animate(CurvedAnimation(  
    parent: _animationController,  
    curve: const Interval(0, 0.4),  
));  
  
_sizeAnimation = SizeTween(  
    begin: Size(200, 50),  
    end: Size(50, 200),  
).animate(CurvedAnimation(  
    parent: _animationController,  
    curve: const Interval(0.3, 0.7),  
));  
  
_colorAnimation = ColorTween(  
    begin: Colors.yellow,  
    end: Colors.red,  
).animate(CurvedAnimation(  
    parent: _animationController,  
    curve: const Interval(0.7, 1.0),  
));
```



# TweenAnimationBuilder

```
class _PageState extends State<Page> {
  @override
  Widget build(BuildContext context) {
    return TweenAnimationBuilder(
      tween: IntTween(begin: 0, end: 100),
      duration: const Duration(seconds: 1),
      builder: (context, value, child) => Text(value.toString()),
    );
  }
}
```

# TweenAnimationBuilder

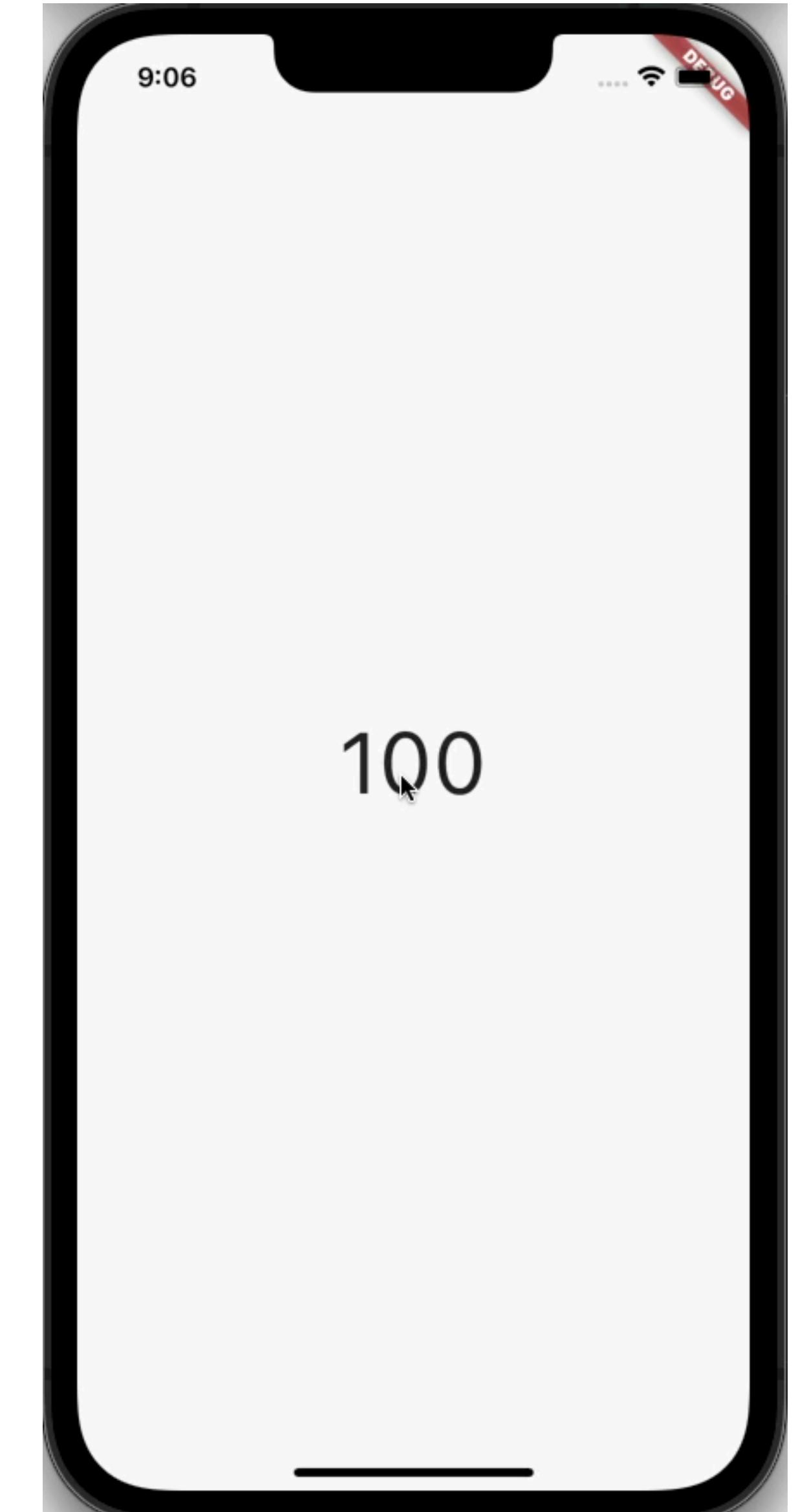
```
class _PageState extends State<Page> {
    int _counter = 100;

    @override
    Widget build(BuildContext context) {
        return GestureDetector(
            onTap: () => setState(() => _counter += 100),
            child: TweenAnimationBuilder(
                tween: IntTween(begin: 0, end: _counter),
                duration: const Duration(seconds: 1),
                builder: (context, value, child) => Text(value.toString()),
            ),
        );
    }
}
```

# TweenAnimationBuilder

```
class _PageState extends State<Page> {
    int _counter = 100;

    @override
    Widget build(BuildContext context) {
        return GestureDetector(
            onTap: () => setState(() => _counter += 100),
            child: TweenAnimationBuilder(
                tween: IntTween(begin: 0, end: _counter),
                duration: const Duration(seconds: 1),
                builder: (context, value, child) => Text(value.toString()),
            ),
        );
    }
}
```



# Can we make it easier?

- SlideTransition
- ScaleTransition
- RotationTransition
- SizeTransition
- FadeTransition
- RelativeRectTween
- PositionedTransition
- RelativePositionedTransition
- DecoratedBoxTransition
- AlignTransition
- DefaultTextStyleTransition

# Can we make it easier?

SlideTransition  
ScaleTransition  
**RotationTransition**  
SizeTransition  
FadeTransition  
RelativeRectTween  
PositionedTransition  
RelativePositionedTransition  
DecoratedBoxTransition  
AlignTransition  
DefaultTextStyleTransition



```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: RotationTransition(
        turns: _animationController,
        child: Container(
          color: Colors.red,
          width: 300,
          height: 300,
        ),
      ),
    );
}
```

**Some other tricks...**

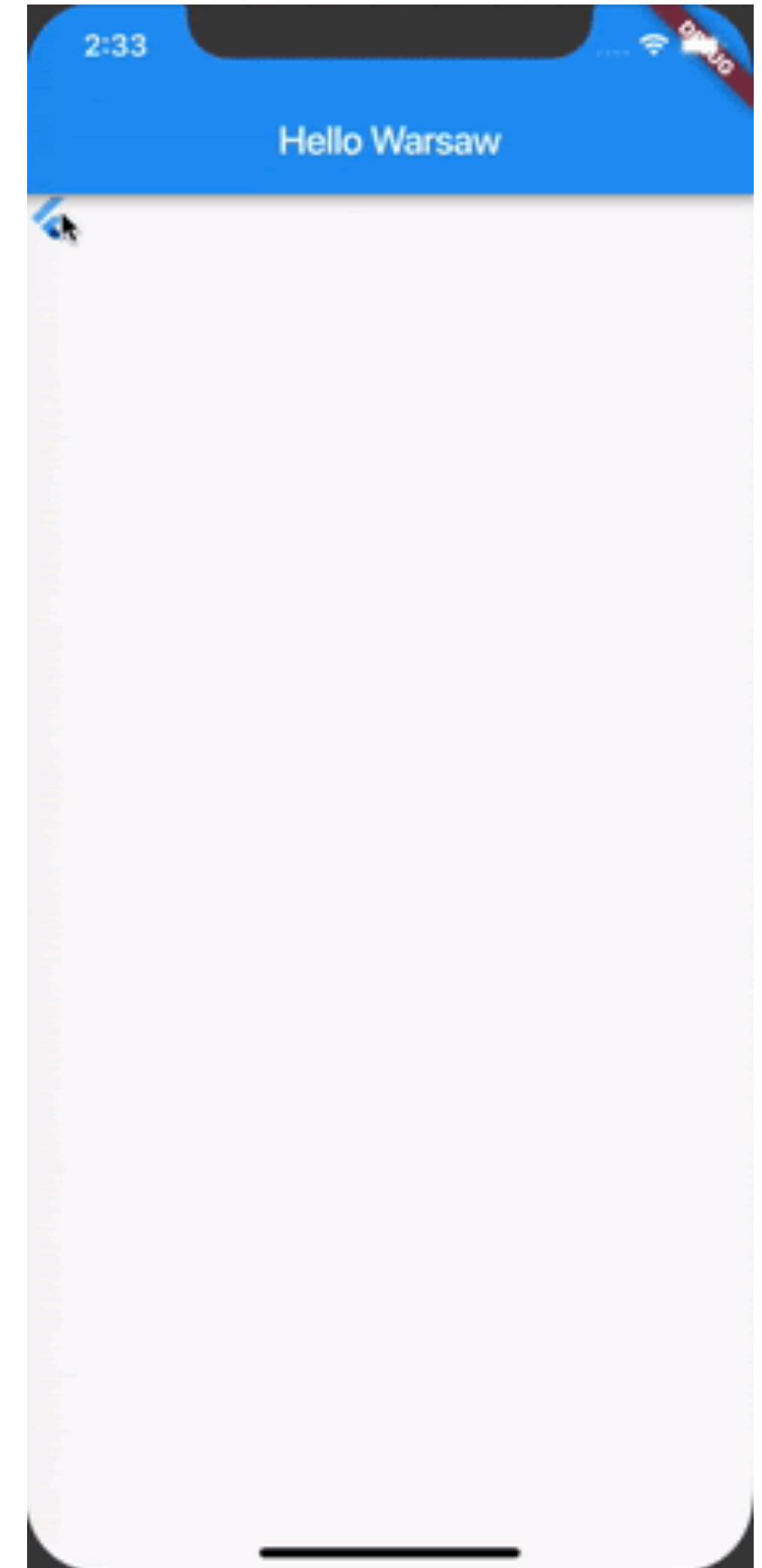
# Heroes

```
class _PageState extends State<Page> {
    void _goToPage2() => Navigator.of(context).push(
        MaterialPageRoute(builder: (_) => Page2()),
    );
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(title: const Text('Hello Warsaw')),
        body: Hero(
            tag: 'FlutterLogo',
            child: GestureDetector(
                onTap: _goToPage2,
                child: const FlutterLogo(),
            ),
        ),
    );
}
}

class Page2 extends StatelessWidget {
    const Page2({Key? key}) : super(key: key);

    @Override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: const Text('Page 2'),
            ),
            body: const Center(
                child: Hero(
                    tag: 'FlutterLogo',
                    child: FlutterLogo(size: 200),
                ),
            ),
        );
    }
}
```



# Animations package

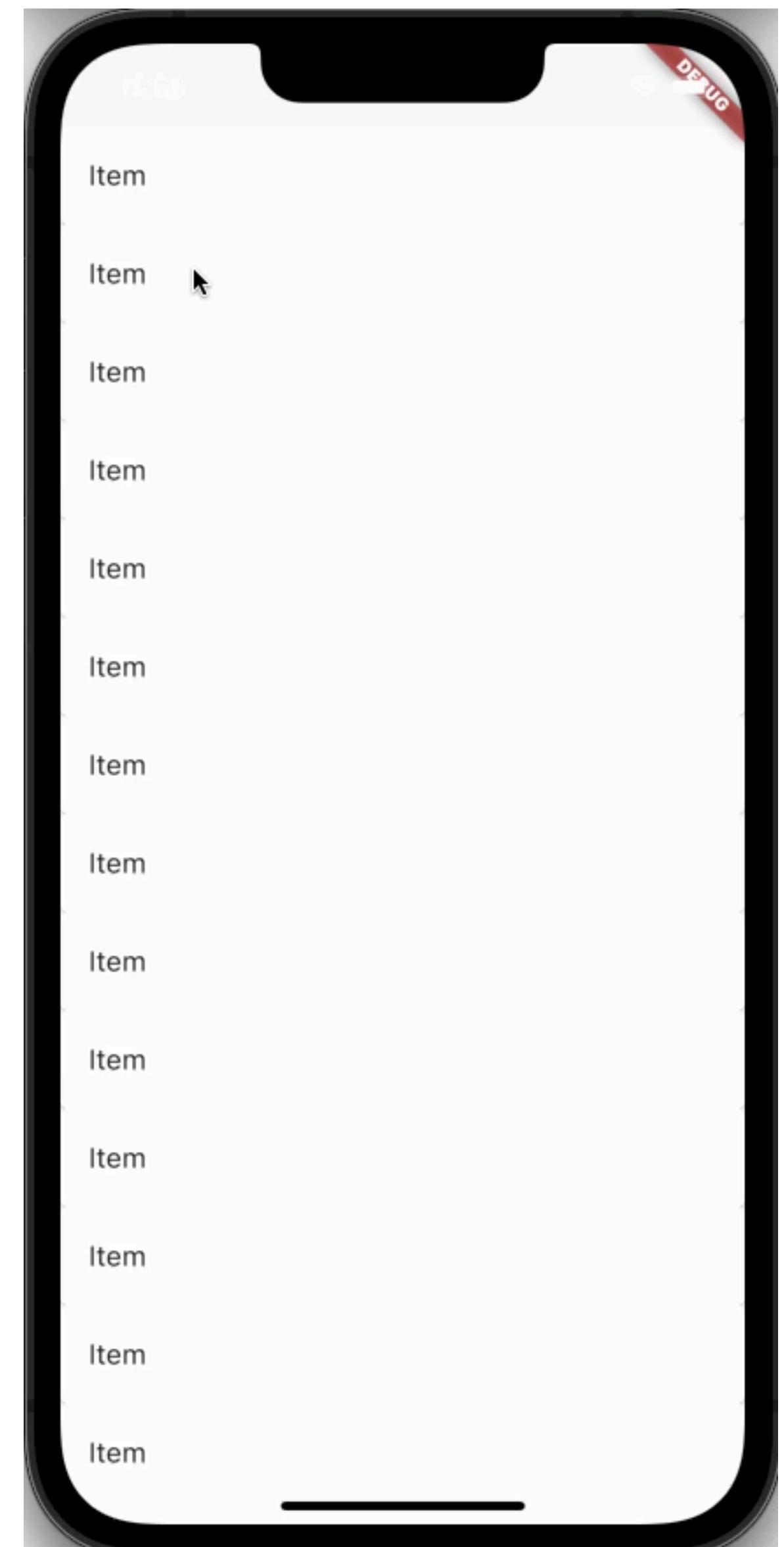
The image displays a collage of nine screenshots from mobile applications, each illustrating a different animation or interaction design. The screenshots are arranged in three rows: the top row contains five screens, the middle row contains two screens, and the bottom row contains two screens.

- Screenshot 1:** A music library interface showing album covers for "Ology" by Galant and "Mothership" by Dance Gavin Dance. It includes a search bar at the top and navigation buttons at the bottom.
- Screenshot 2:** An inbox screen with a purple header. It lists several notifications, such as "Brunch this weekend?" and "Summer BBQ". A plus sign icon is visible at the bottom right.
- Screenshot 3:** A calendar view for May. It shows events like "Design Review" on Friday, "Dinner with Rob" on Saturday, and "Giants game" on Sunday. A small modal window is overlaid on the calendar.
- Screenshot 4:** A list of "Summer Recipes". The first item is "Crunchy croissants". Each recipe card includes a small thumbnail image, a title, a number (e.g., 01, 02), and a brief description.
- Screenshot 5:** A user profile screen for "Hi David Park". It features a profile picture, a sign-in button, and fields for "Email or phone number" and "Forgot email?". Buttons for "Create account" and "NEXT" are at the bottom.
- Screenshot 6:** A recipe card for "Crunchy croissant". It shows a step-by-step guide with numbered circles (1-6) and descriptive text. Step 1 shows hands mixing flour. The text describes adding remaining flour, milk, melted butter, and salt to the mixture.
- Screenshot 7:** A second recipe card for "Crunchy croissants". It lists steps 2 through 6. Step 2 shows hands kneading dough. The text describes adding the mixture to a bowl and covering it.
- Screenshot 8:** A sidebar or search results page for "Summer Recipes". It lists the same five recipes as Screenshot 4, each with a small thumbnail and a number (01-05).
- Screenshot 9:** A third recipe card for "Crunchy croissants". It shows steps 3 through 6. Step 3 shows hands adding ingredients to a bowl. The text describes kneading the mixture.

# Animations package

```
class _OpenContainer extends StatelessWidget {
  const _OpenContainer({
    Key? key,
  }) : super(key: key);

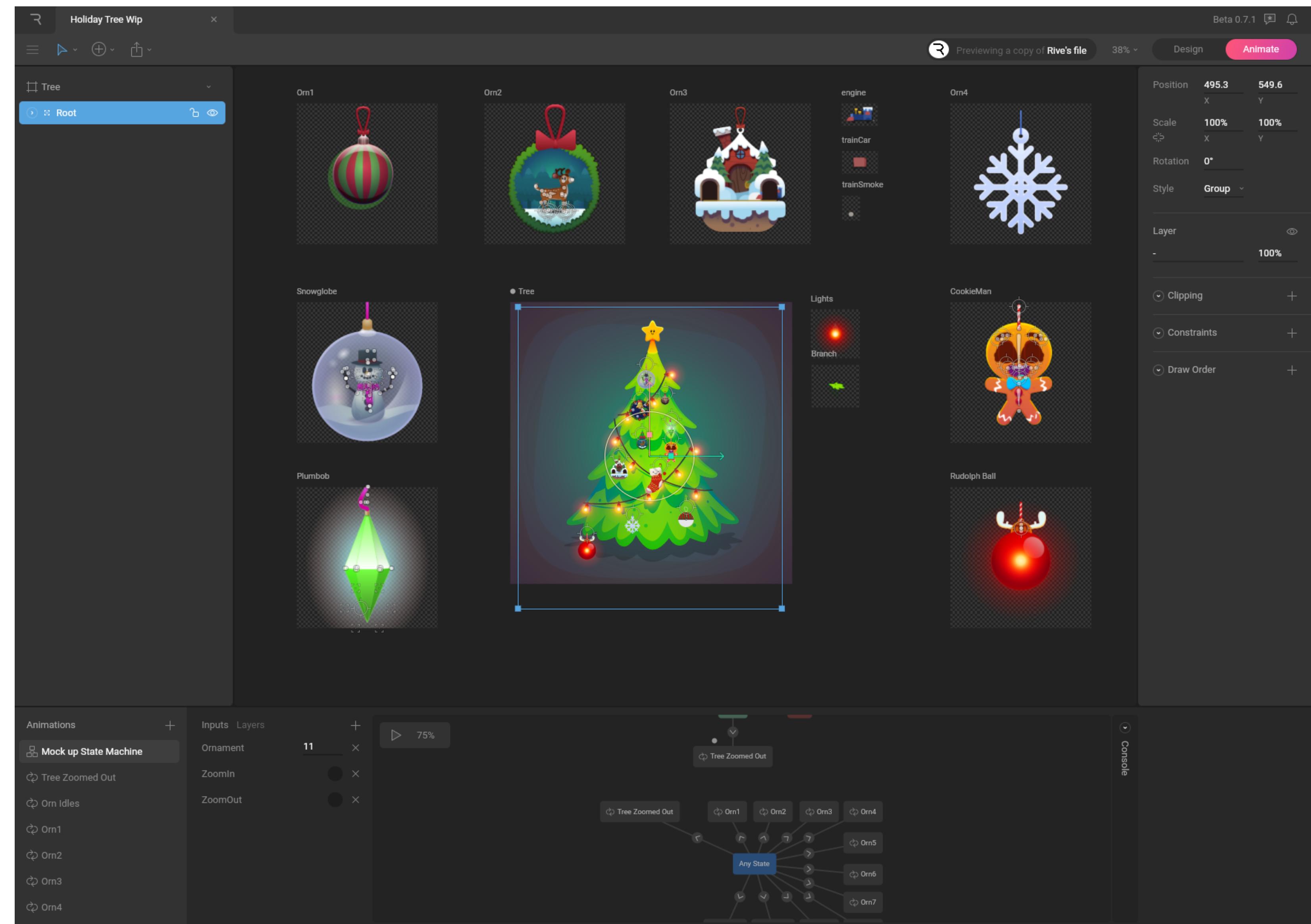
  @override
  Widget build(BuildContext context) {
    return OpenContainer(
      transitionDuration: Duration(milliseconds: 1000),
      closedBuilder: (_, __) => const Center(
        child: ListTile(title: Text('Item')),
      ),
      openBuilder: (_, __) => Scaffold(
        appBar: AppBar(title: const Text('Page 2')),
        body: const Center(child: Text('page2')),
      ),
    );
  }
}
```



# Rive



# Rive



# Next steps

The screenshot shows a YouTube video player. At the top, a yellow bar contains the text "Explicit Animations require control of time...". Below it, the video title is "Animations in Flutter Done Right" by "Emily Fortuna & Andrew Fitz Gibbon". The video content shows a man speaking. On the left, there is code for the `AnimatedIcon` constructor:

```
icon: AnimatedIcon(  
    progress: _animationController,  
    icon: AnimatedIcons.play_pause,  
)  
  
Constructors  
  
AnimatedIcon({Key key, @required AnimatedIconData icon, @required Animation<double> progress,  
    Color color, double size, String semanticLabel, TextDirection textDirection})  
Creates an AnimatedIcon. [...]  
const
```

A purple arrow points from the text "Explicit Animations require control of time..." to the `progress` parameter in the constructor code. The video player interface includes a play button, volume controls, and a timestamp of 26:45 / 42:05.

[youtube.com/watch?v=wnARLByOtKA](https://youtube.com/watch?v=wnARLByOtKA)

The screenshot shows a YouTube video player. At the top, a yellow bar contains the text "Explicit Animations require control of time...". Below it, the video title is "Implementing complex UI with Flutter" by "Marcin Szałek". The video content shows a man speaking. On the left, there is Dart code for a state class:

```
class _PageState extends State<Page> {  
    List< GlobalKey< _ItemFaderState>> keys;  
  
    @override  
    void initState() {  
        super.initState();  
        keys = List.generate(  
            5,  
            (_) => GlobalKey< _ItemFaderState>(),  
        );  
        OnInit();  
    }  
  
    void OnInit() async {  
        for ( GlobalKey< _ItemFaderState> key in keys) {  
            await Future.delayed(Duration(milliseconds: 40));  
            key.currentState.show();  
        }  
    }  
  
    void onTap() async {  
        for ( GlobalKey< _ItemFaderState> key in keys) {  
            await Future.delayed(Duration(milliseconds: 40));  
            key.currentState.hide();  
        }  
        widget.onOptionSelected();  
    }  
    ...  
}
```

To the right of the code, there is a screenshot of a mobile application showing a purple screen with a list of items: "Business", "Personal", and "Others". A small inset image shows a person speaking at a conference. The video player interface includes a play button, volume controls, and a timestamp of 22:35 / 44:25.

[youtube.com/watch?v=FCyoHcICqc8](https://youtube.com/watch?v=FCyoHcICqc8)

# Next steps

Flutter

Multi-Platform ▾ Development ▾ Ecosystem ▾ Showcase Docs ▾ Get started

Free access to best-selling book Flutter Apprentice, for a limited time only! Learn more...

Get started

Samples & tutorials

Development

- User interface
  - Introduction to widgets
  - Building layouts
  - Adding interactivity
  - Assets and images
  - Navigation & routing
- Animations
  - Introduction
  - Overview
  - Tutorial
  - Implicit animations
  - Hero animations
  - Staggered animations
- Advanced UI
  - Widget catalog
  - Data & backend
  - Accessibility & internationalization
  - Platform integration
  - Packages & plugins
  - Add Flutter to existing app
  - Tools & features
  - Migration notes
- Testing & debugging
- Performance & optimization
- Deployment
- Resources
- Reference
- Who is Dash?
- Widget index

## Introduction to animations

Development > UI > Animations

Well-designed animations make a UI feel more intuitive, contribute to the slick look and feel of a polished app, and improve the user experience. Flutter's animation support makes it easy to implement a variety of animation types. Many widgets, especially [Material widgets](#), come with the standard motion effects defined in their design spec, but it's also possible to customize these effects.

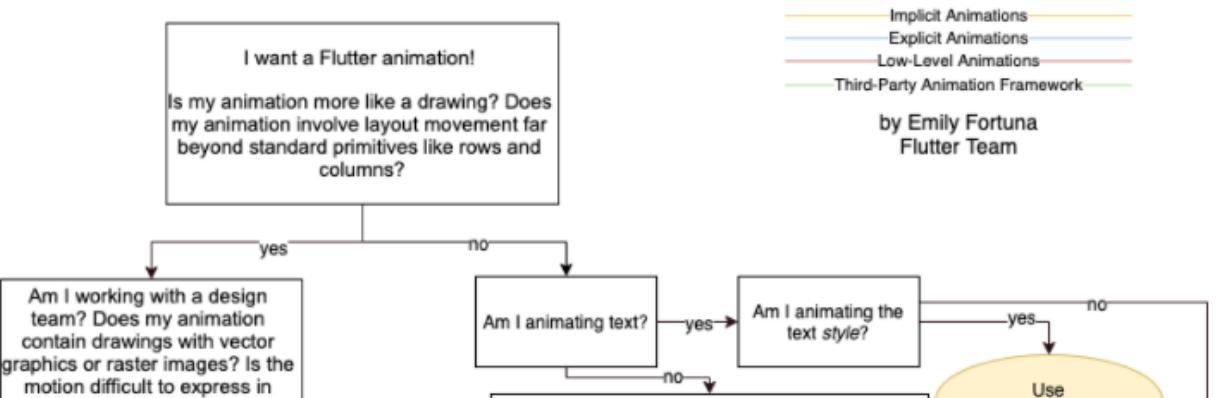
## Choosing an approach

There are different approaches you can take when creating animations in Flutter. Which approach is right for you? To help you decide, check out the video, [How to choose which Flutter Animation Widget is right for you?](#) (Also published as a [companion article](#).)



(To dive deeper into the decision process, watch the [Animations in Flutter done right](#) video, presented at Flutter Europe.)

As shown in the video, the following decision tree helps you decide what approach to use when implementing a Flutter animation:



The decision tree starts with "I want a Flutter animation! Is my animation more like a drawing? Does my animation involve layout movement far beyond standard primitives like rows and columns?"  
If yes: Am I working with a design team? Does my animation contain drawings with vector graphics or raster images? Is the motion difficult to express in  
If no: Am I animating text?  
If yes: Am I animating the text style?  
If yes: Use  
If no: Use

Contents

- Choosing an approach
- Codelabs, tutorials, and articles
- Animation types
  - Tween animation
  - Physics-based animation
- Pre-canned animations
- Common animation patterns
  - Animated list or grid
  - Shared element transition
  - Staggered animation
- Other resources

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

# Thank you :)

Marcin Szałek

@marcin\_szalek

marcin@fidev.io

# Questions?

Marcin Szałek

@marcin\_szalek

marcin@fidev.io