Smooth as Butter: Building Beautiful Animations in Flutter

Make your Flutter UI feel alive with captivating animations that engage users and elevate your app experience.

Why Animations Matter

Motion isn't just eye candy, it's a fundamental aspect of modern interfaces that provides:

- Meaningful feedback on user interactions
- Contextual cues for spatial relationships
- Delight that keeps users coming back

> "Great apps feel alive because of great animations."

Leading apps like Material UI, iOS, and TikTok use animations to create distinctive, intuitive experiences.

Types of Animations in Flutter

1	2	3
Implicit Automatically animates property changes with minimal code • AnimatedContainer • AnimatedOpacity • AnimatedPositioned	**Explicit** Offers complete control via animation controllers • AnimationController • Tween • AnimatedBuilder	**Hero** Creates shared element transitions between screens • Hero widget with matching tags • Smooth navigation experiences

Implicit Animations

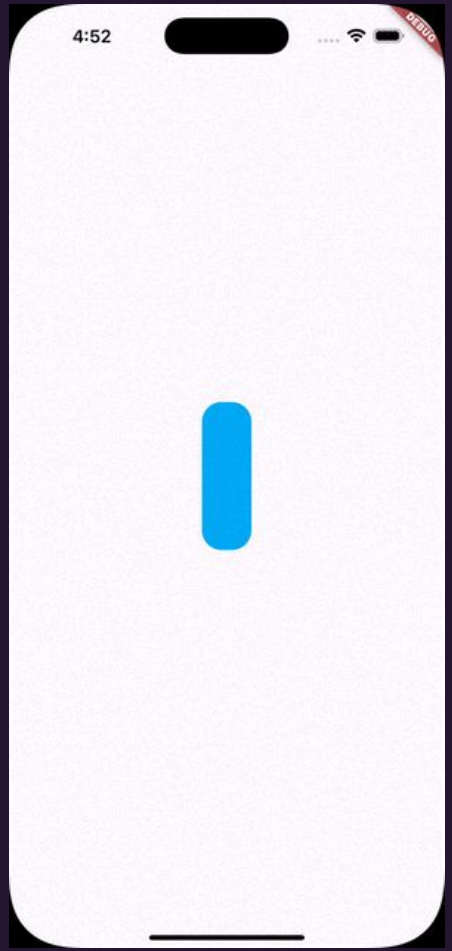
When to use:

- Simple property transitions
- Size, color, opacity changes
- Position adjustments

Perfect for state-based UI changes where you need smooth transitions with minimal code.

animated_builder.dart

```dart
bool valueChanger = true;
...
return AnimatedContainer(
  duration: Duration(milliseconds: 500),
  height: valueChanger ? 50 : 150,
  width: valueChanger ? 150 : 50,
  decoration: BoxDecoration(
    color: valueChanger ? Colors.amberAccent : Colors.lightBlue,
    borderRadius: BorderRadius.circular(valueChanger ? 40 : 20),
  ),
);
...
```



Explicit Animations

AnimationController

Manages animation timing, direction, and playback state

Requires TickerProviderStateMixin in your StatefulWidget

Tween

Defines start and end values for the animation

Can animate any value type (double, color, offset, etc.)

AnimatedBuilder

Efficiently rebuilds only animated portions of your UI

Improves performance for complex animations

animated_builder.dart

```dart
final animationController = AnimationController(
  vsync: this,
  duration: Duration(milliseconds: 400),
);
final animation = Tween<double>(begin:0, end:1).animate(animationController);
```

AnimatedBuilder + Custom Widgets

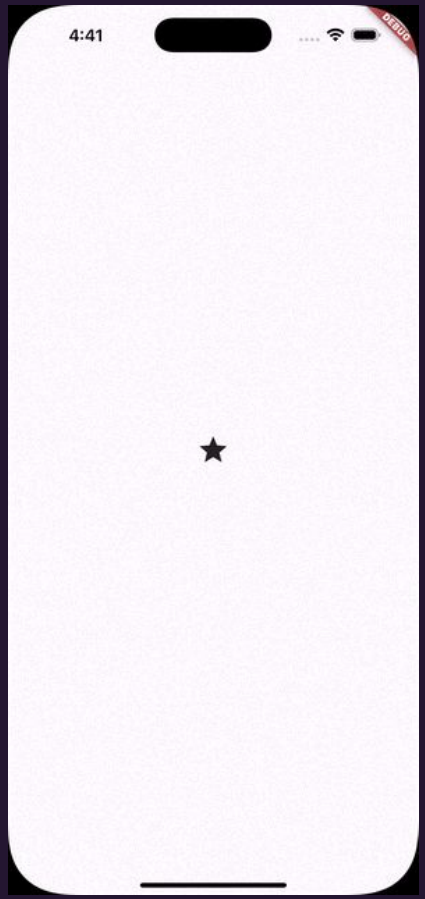
AnimatedBuilder creates efficient animations by:

- Rebuilding only what's needed during animation
- Separating static from animated content
- Enabling powerful combinations with Transform

Performance tip: Pass static widgets as the "child" parameter to prevent unnecessary rebuilds.

animated_builder.dart

```dart
...
return AnimatedBuilder(
  animation: animation,
  builder: (context, child) => Transform.scale(
    scale: animation.value,
    child: child
  ),
  child: Icon(
    Icons.star,
    size: 150,
  ),
);
...
```



Hero Animations

Wrap Source Widget

Add a Hero widget with a unique tag to the widget you want to animate.

Match on Destination

Use the same tag on the destination screen to create connection.

Navigate

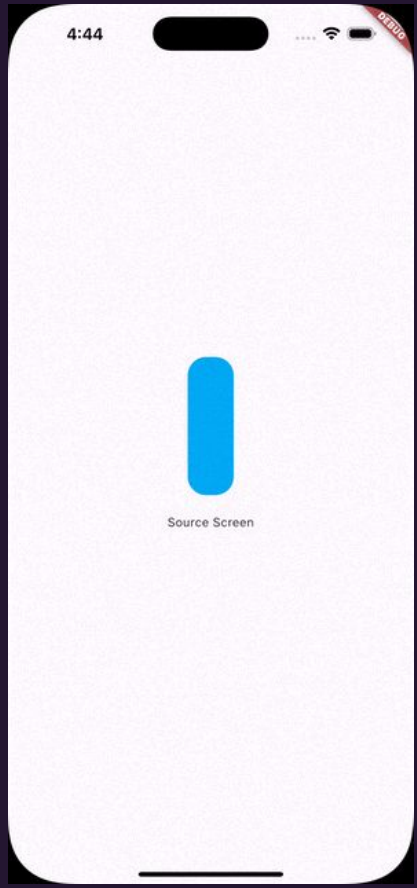
Flutter automatically handles the transition animation between screens.

source_screen.dart | destination_screen.dart

```dart
...
return Hero(
  tag: 'heroTag',
  child: SourceChildWidget(),
);
...
```

source_screen.dart | destination_screen.dart

```dart
...
return Hero(
  tag: 'heroTag',
  child: DestinationWidget(),
);
...
```



Animation Best Practices

Keep Durations Short

Aim for 200 – 400 ms for most animations

Longer animations can feel sluggish and interrupt flow

Use Built-in Curves

Curves.easeOut for natural movement

Curves.bounceIn for playful effects

Curves.easeInOutCubic for smooth transitions

Optimize Performance

Run heavy logic outside animation callbacks

Use const widgets when possible

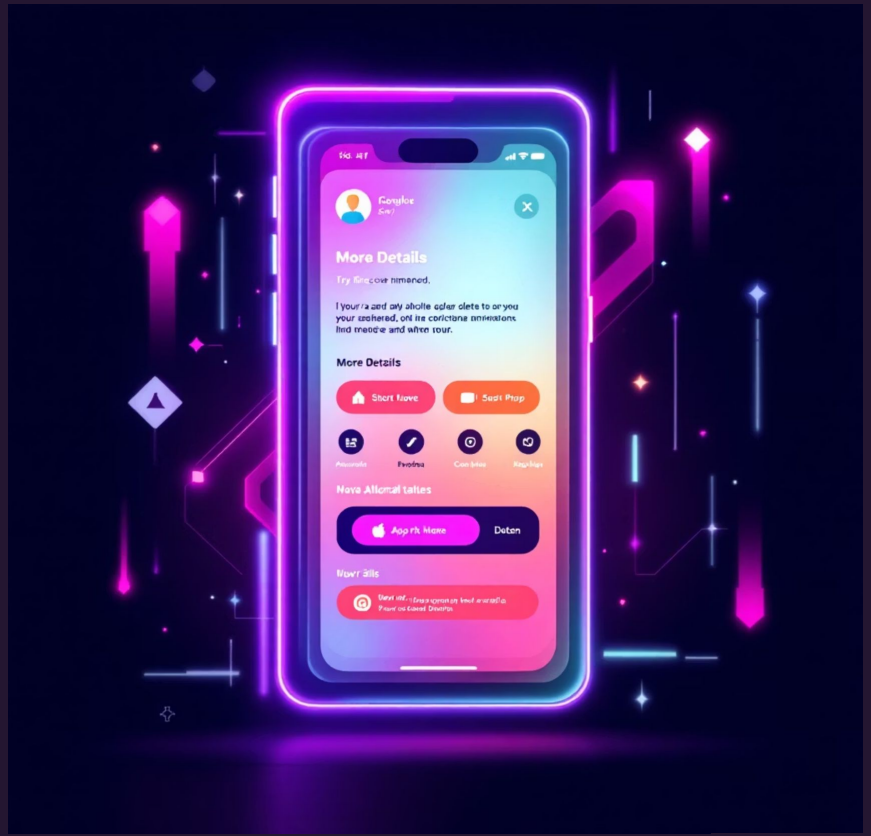
Test on low-end devices

Real UI Demo - Expanding Card

Creating a responsive expanding card combines multiple animation techniques:

- AnimatedContainer for size changes
- AnimatedOpacity for fading in details
- AnimatedCrossFade for swapping content

This pattern works great for news feeds, product listings, and settings panels where space is limited.



The beauty is in the simplicity, users get a rich, interactive experience with minimal code overhead.

Tools & Resources

Flutter Animate

Powerful package that simplifies complex animations with a chainable API

pub.dev/packages/flutter_animate

Rive

Create and implement advanced interactive animations

rive.app

Flutter Docs

Comprehensive guides and tutorials for all animation types

flutter.dev/docs/development/ui/animations

For a bonus challenge: Try integrating Teachable Machine for gesture or sound-triggered animations!

Key Animation Takeaways

Let's cement what we've learned about crafting engaging animations:

1

Implicit Animations = Easy Wins

System-provided animations require minimal code but deliver immediate polish. Perfect for transitions, color changes, and simple property animations.

2

Explicit Animations = Full Control

When you need precise timing, complex sequences, or custom behaviors, explicit animations give you frame-by-frame control.

3

Hero Animations = Smooth Navigation

Connect screens with shared elements that transform between states, creating visual continuity that guides users through your app.

"Delight comes from the details." Start small and animate with intention.

Let's Animate Better Apps

Thank You!

Questions?

I'd love to hear from you about your animation challenges and successes.