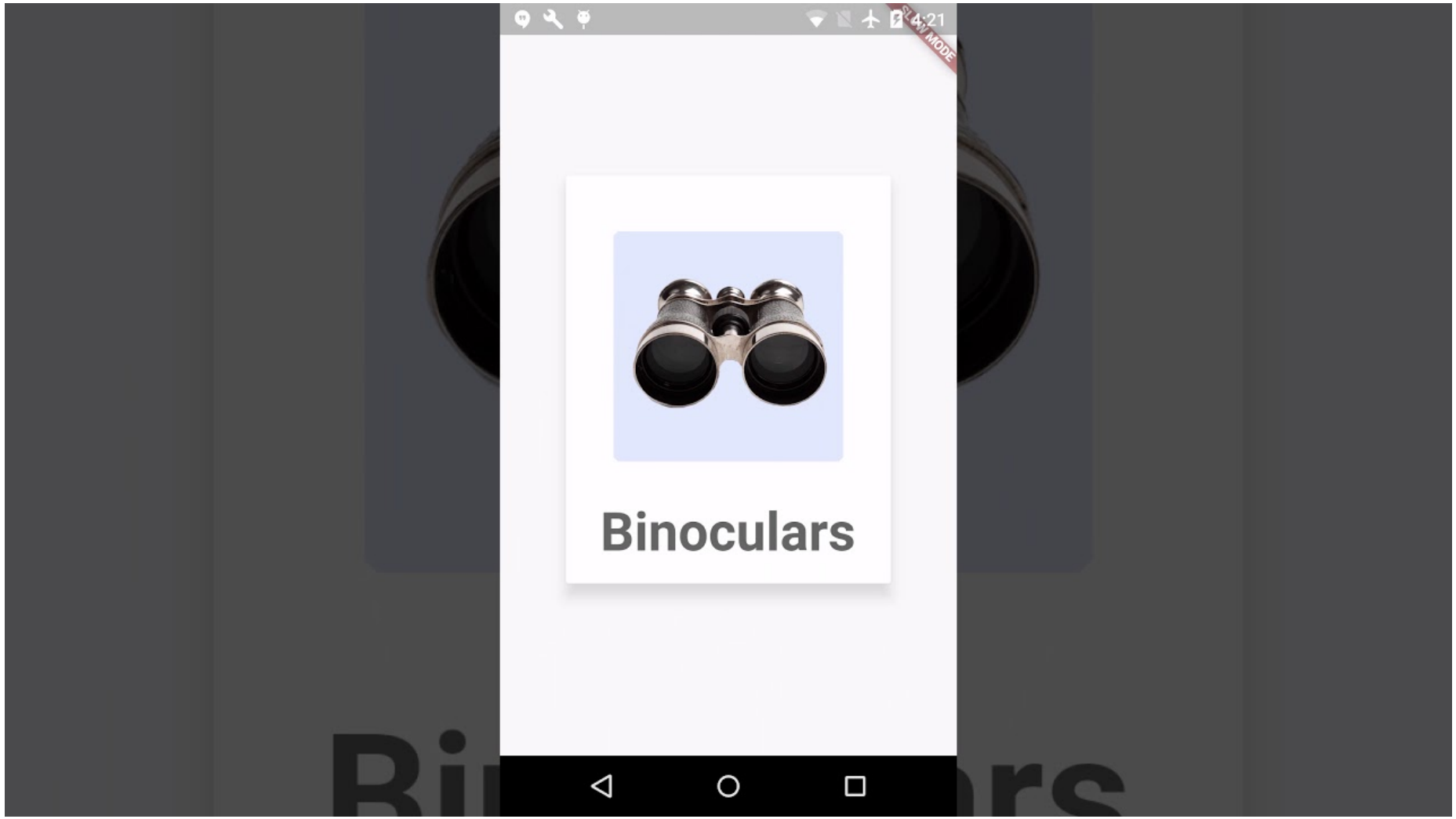
R2S
Resource Software Solution

Animations

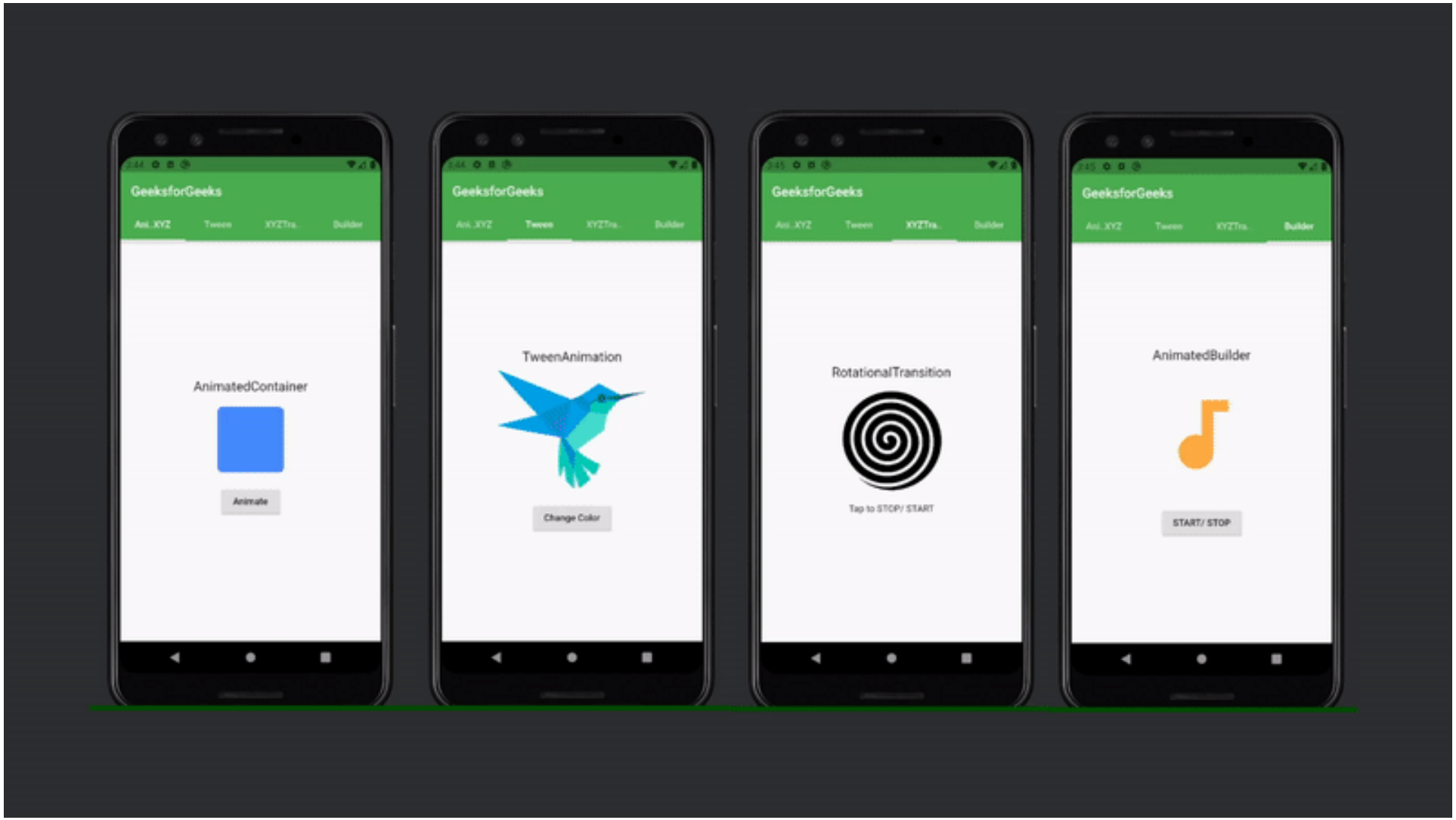
Introduction to animations

- Well-designed animations make a **UI feel more intuitive**



Animation types

- Tween
- Physics-based



Animation types

- **Tween** animation: short for *in-betweening*
  - In a tween animation, the **beginning and ending** points are defined, as well as a timeline, and a curve that defines the timing and speed of the transition

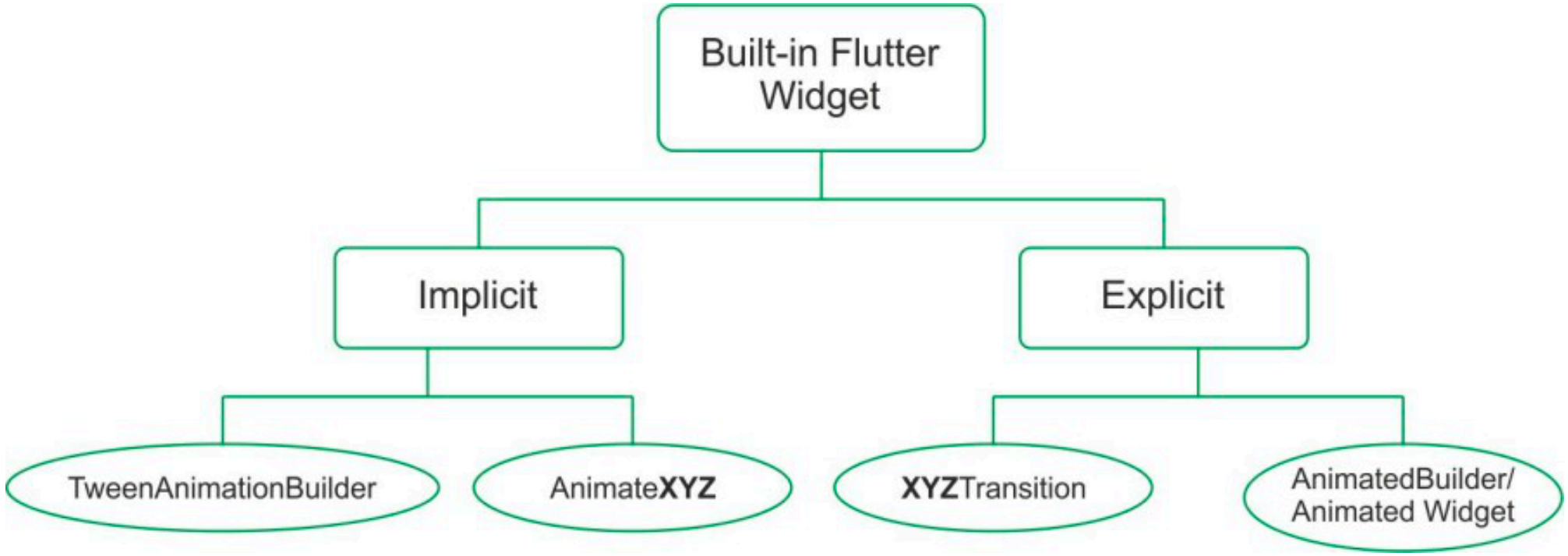
```
late final Animation<Offset> _offsetAnimation = Tween<Offset>(
  begin: Offset.zero,
  // Offset(double dx, double dy)
  // The first argument sets dx, the horizontal component,
  // and the second sets dy, the vertical component.
  end: const Offset(0, 1.5),
).animate(CurvedAnimation(  // Tween
    parent: _controller,
    curve: Curves.elasticIn,
    reverseCurve: Curves.easeOutCirc)
);
```

Animation types

- **Physics-based** animation: motion is modeled to resemble **real-world** behavior.
- Example:
  - When you toss a ball, for example, where and when it lands depends on how fast it was tossed and how far it was from the ground.
  - Similarly, dropping a ball attached to a spring falls (and bounces) differently than dropping a ball attached to a string.

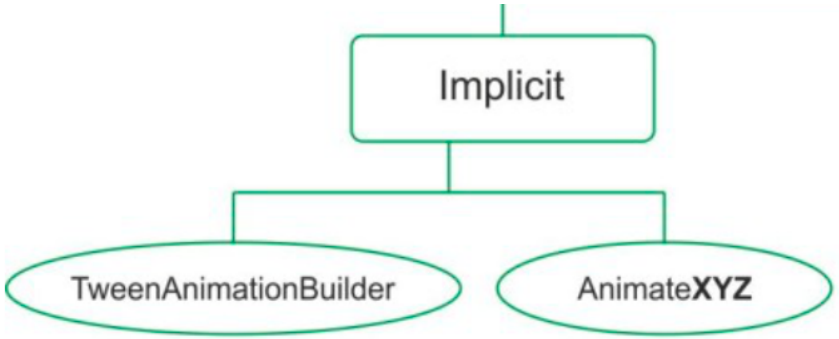
Types of techniques for animation

- Flutter provides two types of techniques for animation.
  1. **Implicit** animation
  2. **Explicit** animation



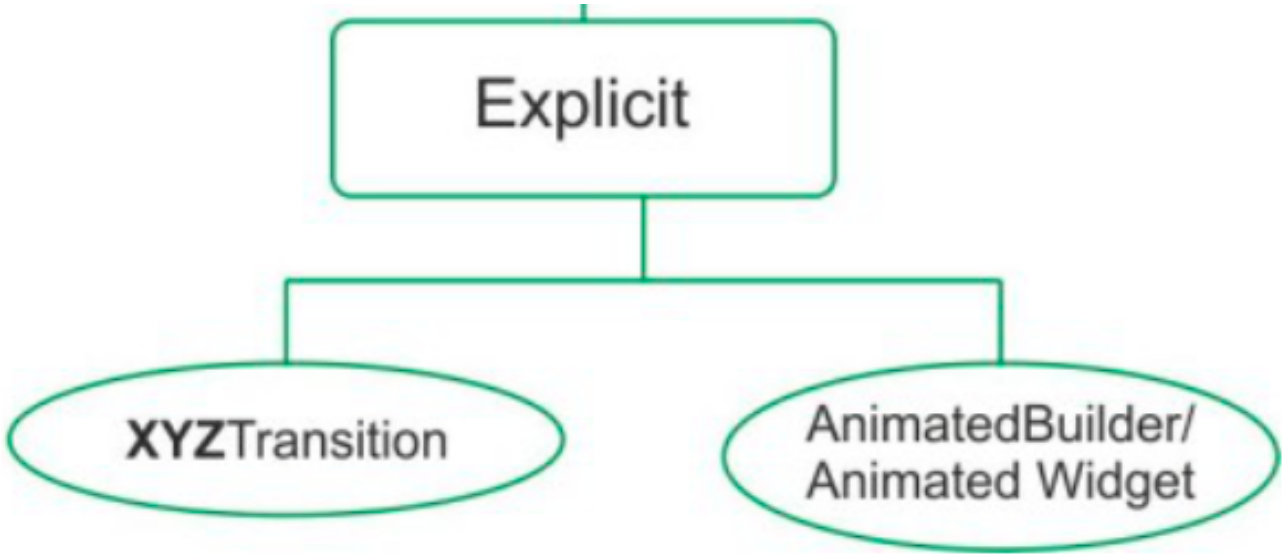
Types of techniques for animation

- **Implicit Animation:**
  - **AnimatedXYZ**: XYZ is a specific **widget** available to be animated.
  - Existing XYZ widgets:
    - Align → **AnimatedAlign**
    - Container → **AnimatedContainer**
    - DefaultTextStyle → **AnimatedDefaultTextStyle**
    - Padding → **AnimatedPadding**
    - Positioned → **AnimatedPositioned**
    - Opacity → **AnimatedOpacity**



Types of techniques for animation

- **Explicit Animation**:
  - **XYZTransition**: XYZ is a specific **widget** available as Transition
  - Some explicit *XYZTransition* available are:
    - SizeTransition
    - FadeTransition
    - AlignTransition
    - RotationTransition
    - DecoratedBoxTransition
    - PositionedTransition



Implicit animations

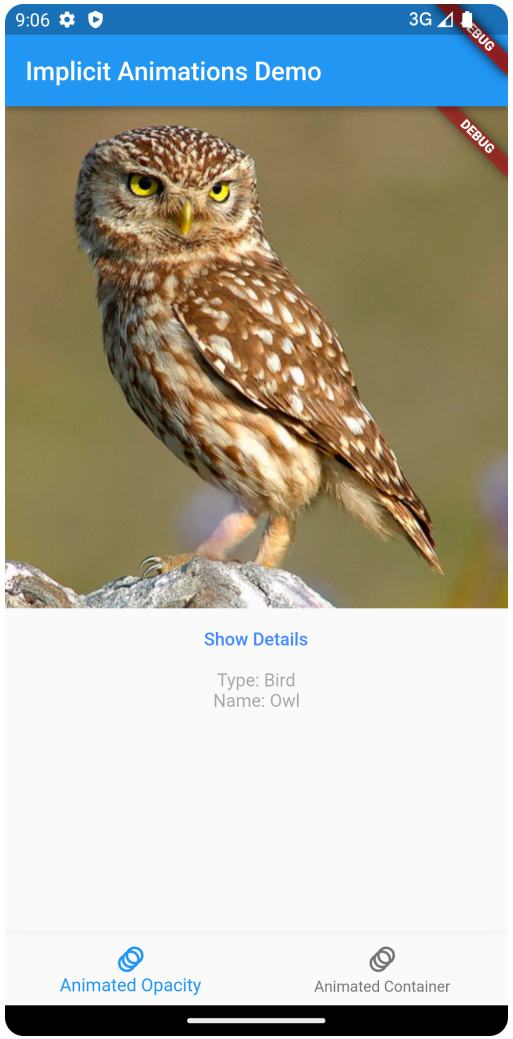
- **AnimatedOpacity**: Animated version of **Opacity** which automatically transitions the **child's opacity** over a given duration whenever the given opacity changes.

```
class _FadeInAnimationState extends State<_FadeInAnimation> {
  double opacity = 0;

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

```
TextButton(
  onPressed: () {
    setState(() {
      opacity = 1;
    });
  },
  child: const Text(...),  // Text
),  // TextButton
```

```
AnimatedOpacity(
  opacity: opacity,
  duration: const Duration(seconds: 4),
  child: Column(
    children: const [
      Text('Type: Bird'),
      Text('Name: Owl'),
    ],
  ),  // Column
)  // AnimatedOpacity
```

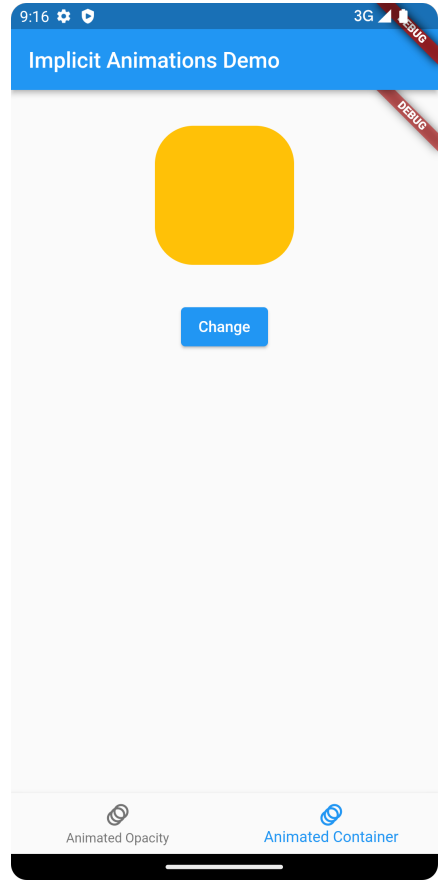


Implicit animations

- **AnimatedContainer**: It will automatically animate between the old and new **values of properties** when they **change** using the provided curve and duration

```dart
final mainColumn = Container(
  padding: const EdgeInsets.all(20),
  child: Column(
    children: [
      titleText,
      subTitle,
      ratings,
      iconList,
      Expanded(child: mainImage),
    ],
  ),  // Column
);  // Container
```

```dart
AnimatedContainer(
  width: 128,
  height: 128,
  margin: EdgeInsets.all(margin),
  decoration: BoxDecoration(
    color: Colors.amber,
    borderRadius: BorderRadius.circular(borderRadius),
  ),  // BoxDecoration
  duration: const Duration(milliseconds: 400),
),  // AnimatedContainer
```



Explicit animations

- **SlideTransition**: Animates the **position of a widget** relative to its normal position.


```dart
class _MyHomeState extends State<_MyHome> with SingleTickerProviderStateMixin {
  late final AnimationController _controller = AnimationController(
      vsync: this,
      duration: const Duration(seconds: 2),
  )..repeat(reverse: true);  // AnimationController

  late final Animation<Offset> _offsetAnimation = Tween<Offset>(
    begin: Offset.zero,
    end: const Offset(0, 1.5),
  ).animate(CurvedAnimation(parent: _controller, curve: Curves.elasticIn));  //

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: SlideTransition(
          position: _offsetAnimation,
          child: const Padding(
            padding: EdgeInsets.all(8),
            child: FlutterLogo(size: 150,),
          ),  // Padding
        ),  // SlideTransition
      ),  // Center
    );  // Scaffold
  }
}
```

Explicit animations

- **AnimationController**: 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.

```dart
late final AnimationController _controller = AnimationController(
    vsync: this,
    duration: const Duration(seconds: 2),
)..repeat(reverse: true);  // AnimationController
```

Explicit animations

- The cascade notation (. .) allows you to **make a sequence of operations on the same object** (including **function calls** and **field access**)
- This notation helps keep Dart code compact and removes the need to create **temporary variables** to store data.

```dart
late final AnimationController _controller = AnimationController(
    vsync: this,
    duration: const Duration(seconds: 2),
)..repeat(reverse: true); // AnimationController
```

Explicit animations

- The cascade notation (. .): Example

```
class Example{
  var a;
  var b;
  void bSetter(b)
  {
    this.b = b;
  }
  void printValues(){
    print(this.a);
    print(this.b);
  }
}
```

```
void main() {
    //Instantiating two Example objects
    Example eg1 = new Example();
    Example eg2 = new Example();

    //Using the .. operator for operations on Example object
    print("Example 1 results:");
    eg1
    ..a = 88
    ..bSetter(53)
    ..printValues();

    //The same operations as above but without the .. operator
    print("Example 2 results:");
    eg2.a = 88;
    eg2.bSetter(53);
    eg2.printValues();
}
```

Explicit animations

- **CurvedAnimation** defines the animation's progress as a **non-linear curve**, especially if you want **different curves** when the animation is going **forward** vs when it is going **backward**.

```
late final Animation<Offset> _offsetAnimation = Tween<Offset>(
  begin: Offset.zero,
  // Offset(double dx, double dy)
  // The first argument sets dx, the horizontal component,
  // and the second sets dy, the vertical component.
  end: const Offset(0, 1.5),
).animate(CurvedAnimation(  // Tween
    parent: _controller,
    curve: Curves.elasticIn,
    reverseCurve: Curves.easeOutCirc)
);
```

Explicit animations

- **Animation** class: An animation consists of **a value (of type T)** together with a status (completed or dismissed)
- Animations can also interpolate types other than **double**, such as Animation<**Color**> or Animation<**Size**>.

```
late final Animation<Offset> _offsetAnimation = Tween<Offset>(
  begin: Offset.zero,
  // Offset(double dx, double dy)
  // The first argument sets dx, the horizontal component,
  // and the second sets dy, the vertical component.
  end: const Offset(0, 1.5),
).animate(CurvedAnimation(  // Tween
    parent: _controller,
    curve: Curves.elasticIn,
    reverseCurve: Curves.easeOutCirc)
);
```

Explicit animations

- **Tween** class: A **linear** interpolation **between** a **beginning** and **ending** value.
- By default, the AnimationController object ranges from 0.0 to 1.0. If you need a **different range** or a **different data type**, you can use a Tween.

```
late final Animation<Offset> _offsetAnimation = Tween<Offset>(
    begin: Offset.zero,
    // Offset(double dx, double dy)
    // The first argument sets dx, the horizontal component,
    // and the second sets dy, the vertical component.
    end: const Offset(0, 1.5),
).animate(CurvedAnimation(  // Tween
      parent: _controller,
      curve: Curves.elasticIn,
      reverseCurve: Curves.easeOutCirc)
);
```

Explicit animations

- **Offset** class: Simply it is a data class to store **X and Y coordinates** and pass that class data to other classes or functions.

```
late final Animation<Offset> _offsetAnimation = Tween<Offset>(
  begin: Offset.zero,
  // Offset(double dx, double dy)
  // The first argument sets dx, the horizontal component,
  // and the second sets dy, the vertical component.
  end: const Offset(0, 1.5),
).animate(CurvedAnimation(  // Tween
    parent: _controller,
    curve: Curves.elasticIn,
    reverseCurve: Curves.easeOutCirc)
);
```

Explicit animations

- What is vsync ?
  - **vsync** keeps the **track of screen**, so that Flutter does not renders the animation when the screen is not being displayed.
  - Using **SingleTickerProviderStateMixin** or **TickerProviderStateMixin** (in case of multiple animations) to **notified** about the animation frames of flutter.

```
class _MyHomeState extends State<_MyHome> with SingleTickerProviderStateMixin {
  late final AnimationController _controller = AnimationController(
    vsync: this,
    duration: const Duration(seconds: 2),
  )..repeat(reverse: true);  // AnimationController
```