Flutter

Training Assignments

Animations

Overview

What you'll learn

- Implicit animations
- Explicit animations

Tasks

1. Implicit animations

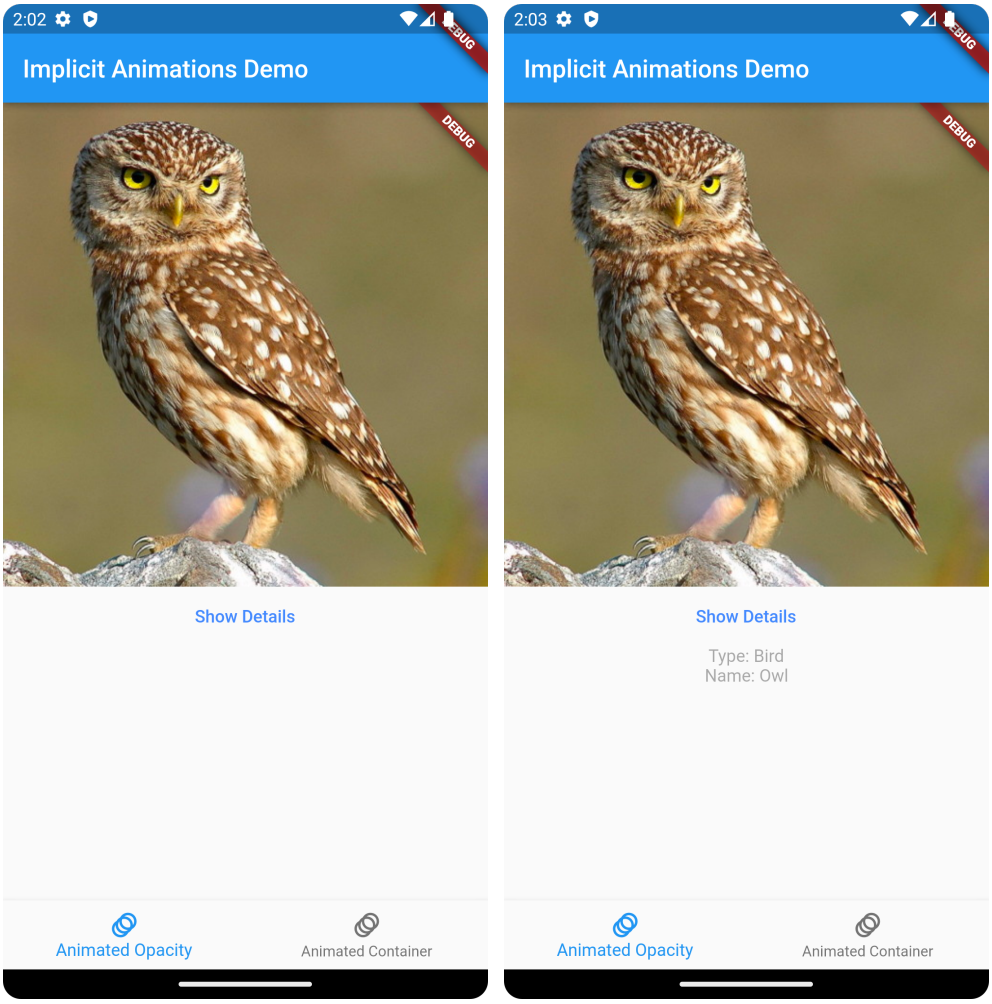
+ UI using an implicitly animated widget called AnimatedOpacity

The following lab shows how to add a fade-in effect to existing UI using an implicitly animated widget called AnimatedOpacity. It consists of a Material App home screen containing:

- A photograph of an owl.
- One **Show details** button that does nothing when clicked.
- Description text of the owl in the photograph.

```dart
class _FadeInAnimation extends StatefulWidget {
  const _FadeInAnimation({super.key});

  @override
  State<_FadeInAnimation> createState() => _FadeInAnimationState();
}

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

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        SizedBox(
          child: Image.asset('images/owl.png'),
        ),  // SizedBox
        TextButton(
          onPressed: () {
            setState(() {
              opacity = 1;
            });
          },
          child: const Text(
            'Show Details',
            style: TextStyle(color: Colors.blueAccent),
          ),  // Text
        ),  // TextButton
        AnimatedOpacity(
          opacity: opacity,
          duration: const Duration(seconds: 4),
          child: Column(
            children: const [
              Text('Type: Bird'),
              Text('Name: Owl'),
            ],
          ),  // Column
        )  // AnimatedOpacity
      ],
    );  // Column
  }
}
```

Run the app

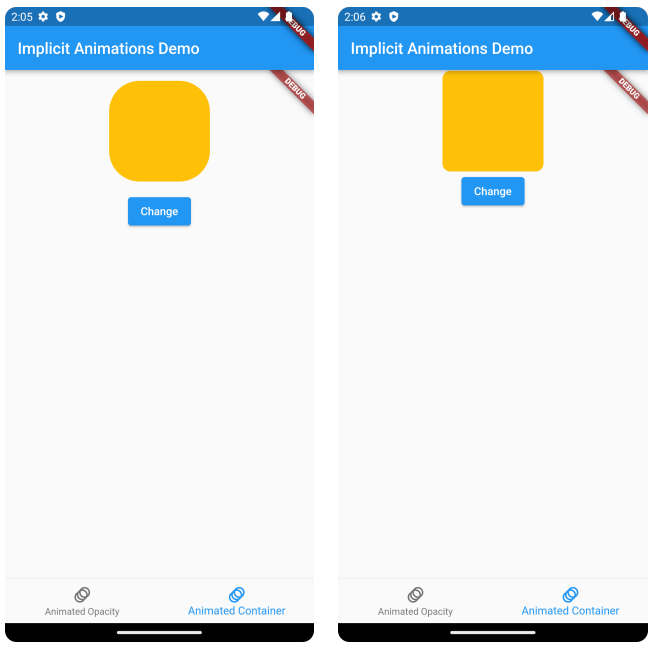

2:02
Implicit Animations Demo
DEBUG
Show Details
Animated Opacity
Animated Container
2:03
Implicit Animations Demo
DEBUG
Show Details
Type: Bird
Name: Owl
Animated Opacity
Animated Container

+ Using the AnimatedContainer widget

```dart
class AnimatedContainerDemo extends StatelessWidget {
  const AnimatedContainerDemo({super.key});

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: _MyAnimatedContainer(),
    );  // MaterialApp
  }
}

class _MyAnimatedContainer extends StatefulWidget {
  const _MyAnimatedContainer({Key? key}) : super(key: key);

  @override
  State<_MyAnimatedContainer> createState() => _MyAnimatedContainerState();
}

class _MyAnimatedContainerState extends State<_MyAnimatedContainer> {
  late double margin;
  late double borderRadius;

  @override
  void initState() {
    super.initState();
    margin = _randomMargin();
    borderRadius = _randomBorderRadius();
  }
```

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: Column(
        children: [
          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
          ElevatedButton(
            onPressed: () {
              setState(() {
                margin = _randomMargin();
                borderRadius = _randomBorderRadius();
              });
            },
            child: const Text('Change')
          )  // ElevatedButton
        ],
      ),  // Column
    ),  // Center
  );  // Scaffold
}

double _randomMargin() {
  return Random().nextDouble() * 64;
}

double _randomBorderRadius() {
  return Random().nextDouble() * 64;
}
}
```

Run the app



2. Explicit animations

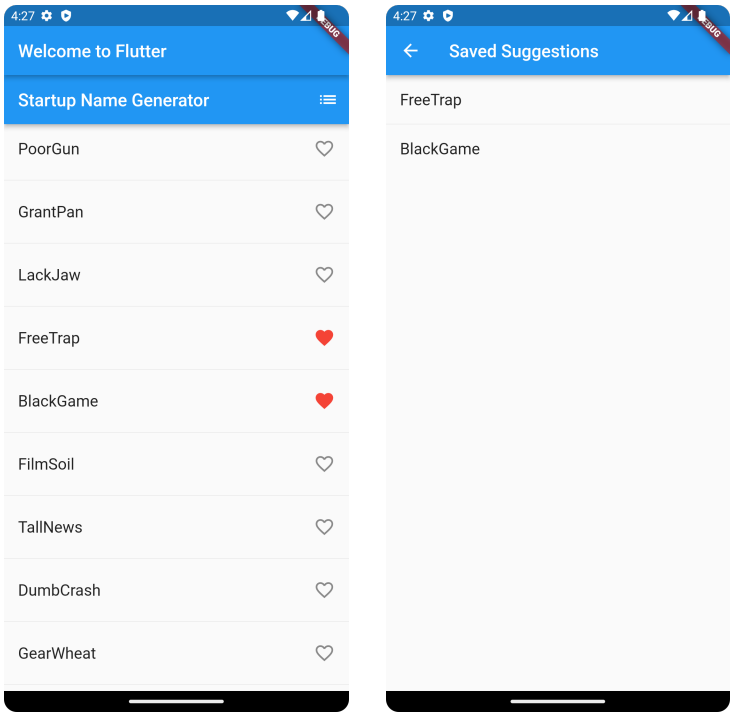
+ Code

```dart
class _MyHome extends StatefulWidget {
  const _MyHome({Key? key}) : super(key: key);

  @override
  State<_MyHome> createState() => _MyHomeState();
}

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,
    // 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)
  );

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

+ Run the app

Extra tasks

Task: Use **SlideTransition** to animate when navigating



Tips

```dart
// Navigate to a new screen
Route _createRoute() {
  return PageRouteBuilder(
    pageBuilder: (context, animation, secondaryAnimation) =>
        DetailsScreen(saved: _saved, biggerFont: _biggerFont),
    transitionsBuilder: (context, animation, secondaryAnimation, child) {
      const begin = Offset(0.0, 1.0);
      const end = Offset.zero;
      const curve = Curves.ease;

      var tween =
          Tween(begin: begin, end: end).chain(CurveTween(curve: curve));

      return SlideTransition(
        position: animation.drive(tween),
        child: child,
      ); // SlideTransition
    },
  ); // PageRouteBuilder
}
```

--THE END—