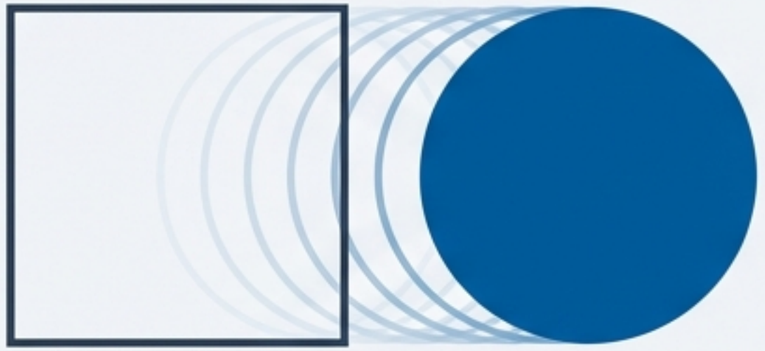
# Flutter ile Canlanan Arayüzler

## Implicit (Örtük) Animasyonlar Rehberi

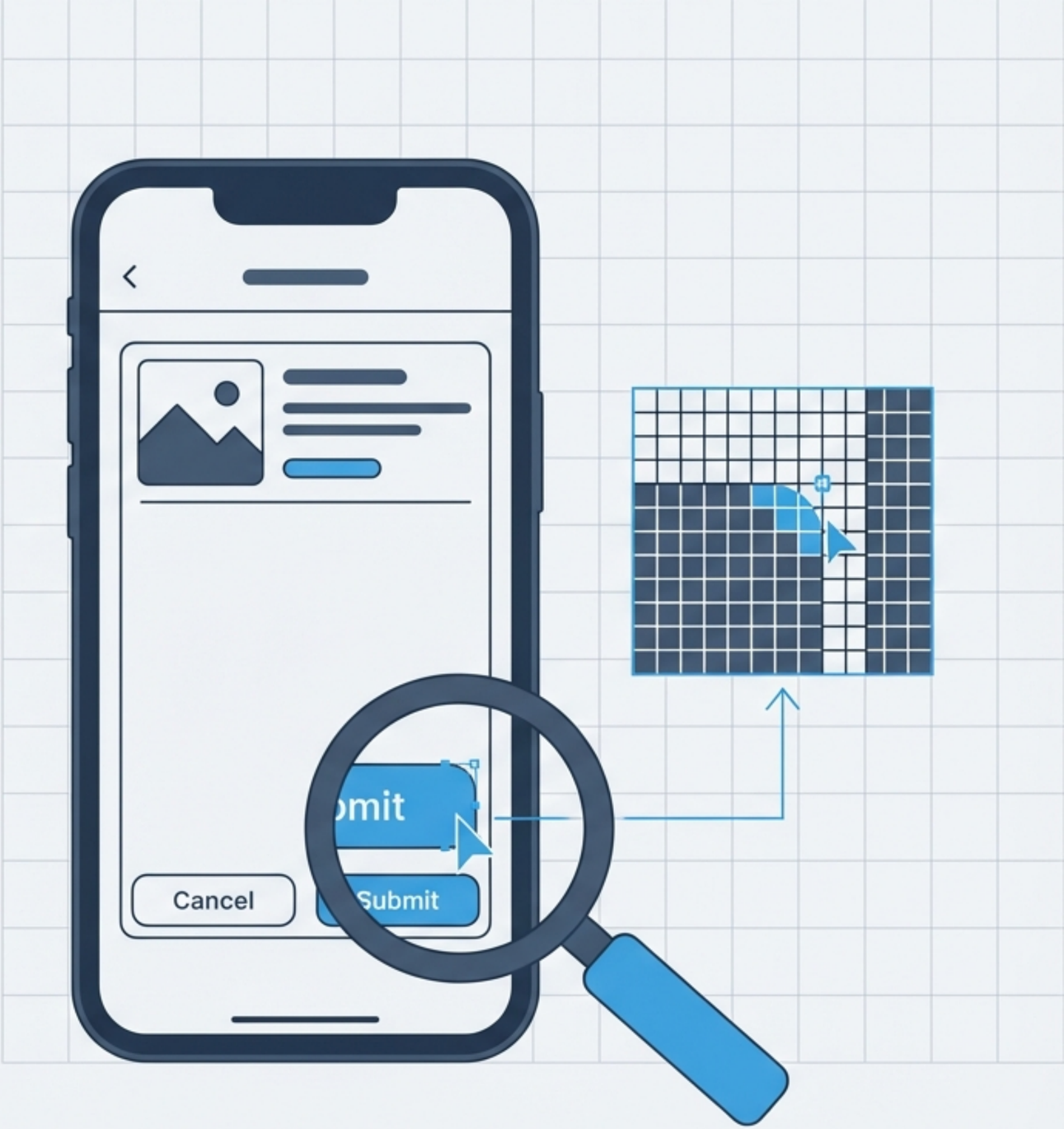
Bu kılavuz, Flutter SDK içinde hazır bulunan araçları kullanarak, uygulamanızın kullanıcı deneyimini minimum eforla nasıl zenginleştireceğinizi anlatır.



# Her Pikselin Kontrolü Sizde

Flutter geliştiricilere UI üzerindeki her bir pikseli kontrol etme gücü verir. Animasyonlar bu sürecin bir istisnası değildir; bir süsleme değil, tasarımın temel parçasıdır.

Sıfırdan bir yapı kurabilir veya Flutter'ın yerleşik widget'larını kullanabilirsiniz. Hedefimiz, karmaşık görünen bu süreci '**zahmetsiz**' hale getirmektir.



# Animasyonun Üç Katmanı

Level of Difficulty

## 1. Implicit Animations (Odak Noktamız)

SDK içine entegre edilmiş, önceden oluşturulmuş widget setleri. Çoğu zaman kurulum bile gerektirmez; varsayılan ayarlar yeterlidir.

## 2. Animation Library

Özel animasyonlar oluşturmanıza yardımcı olan yardımcı sınıflar. Performansı manuel olarak artırma imkanı ve daha fazla kontrol sunar.

## 3. Custom Animations

`Matrix4` veya `Transform` kullanarak sıfırdan oluşturulan animasyonlar. 3D uzayı doğrudan kontrol ederek trigonometri ve matematik gerektiren 'düşük seviye' işlemlerdir.

# Implicit Büyüsü: Sadece İsim Değiştirin

Standart widget'ın başına 'Animated' ekleyin ve bir süre atayın.

**Önce**

```
Container(
  height: counter,
  width: counter,
  color: Colors.blue,
);
```

**Sonra**

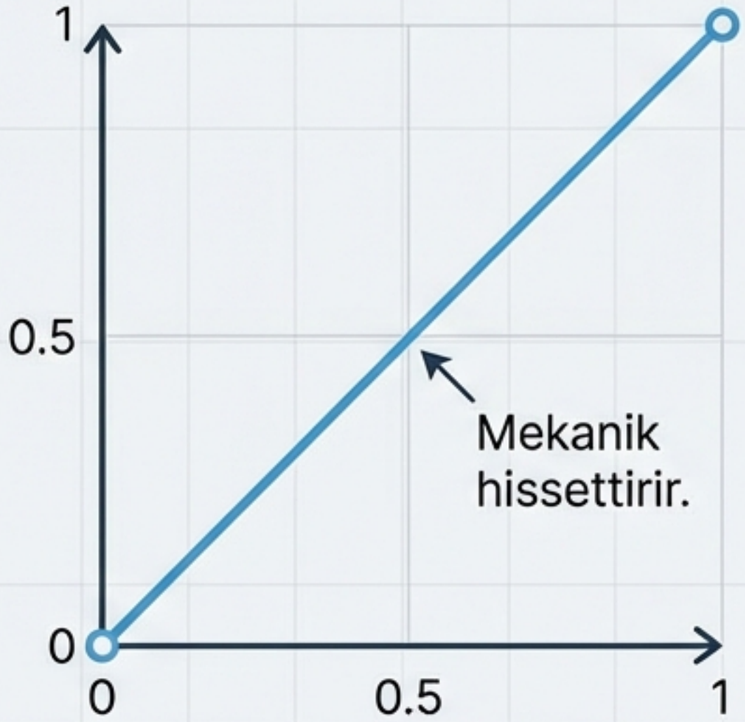
```
AnimatedContainer(    // Değişiklik 1
  height: counter,
  width: counter,
  color: Colors.blue,
  duration: const Duration(seconds: 2) // Değişiklik 2
);
```

**counter** değişkeni değiştiğinde, standart **Container** anında boyut değiştirirken; **AnimatedContainer** bu değişimi 2 saniyeye yayarak akıcı bir geçiş sağlar.

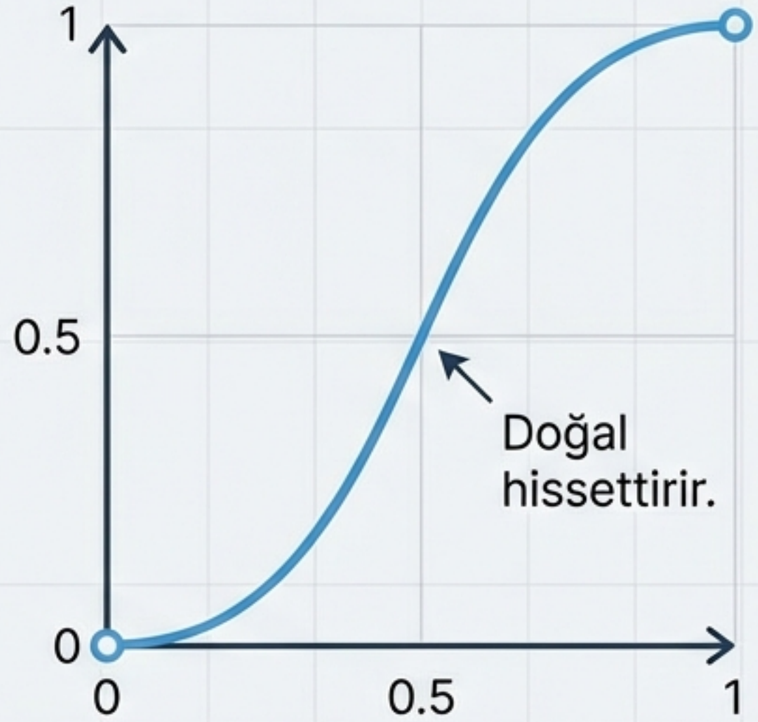
# Matematiğin Estetiği: Eğriler (Curves)

Bir eğri (curve), değerlerin zaman içinde nasıl değiştiğini ifade eden matematiksel bir fonksiyondur.

**Linear (Doğrusal)**




**EaseInOut**

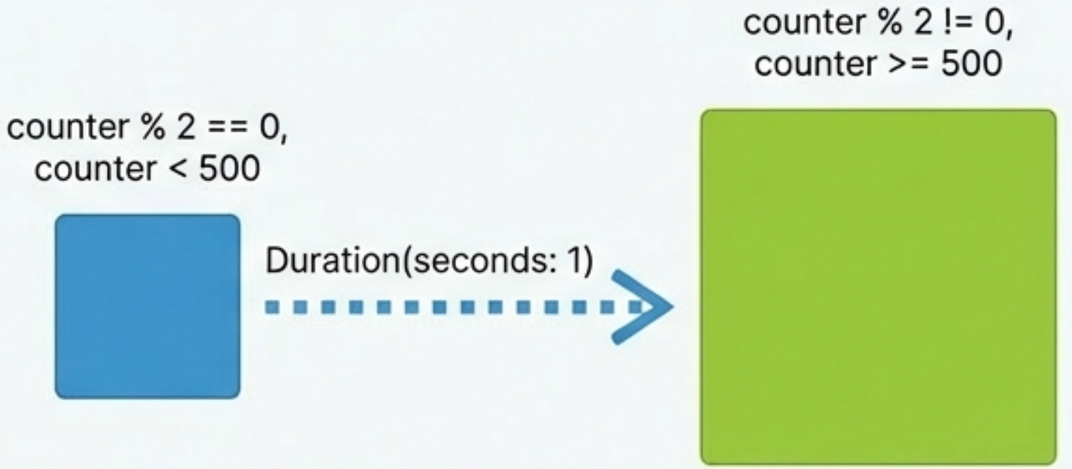



Varsayılan eğri genellikle `Curves.linear` olsa da, curve parametresi ile bunu `Curves.easeInOut` veya `Curves.elasticInOut` gibi daha organik formlara dönüştürebilirsiniz.

# Temel Yapı Taşı: AnimatedContainer

Kullanım: Boyut, renk, sınır çizgileri ve hizalama gibi özellikleri canlandırmak için kullanılır.

Değişken güncellendiğinde, Flutter eski ve yeni değerler arasındaki ara değerleri otomatik olarak hesaplar ve assıandiğind, Flutter edk çıll ara değerleri otomatik olarak hesaplar ve animasyonu gerçekleştirir.



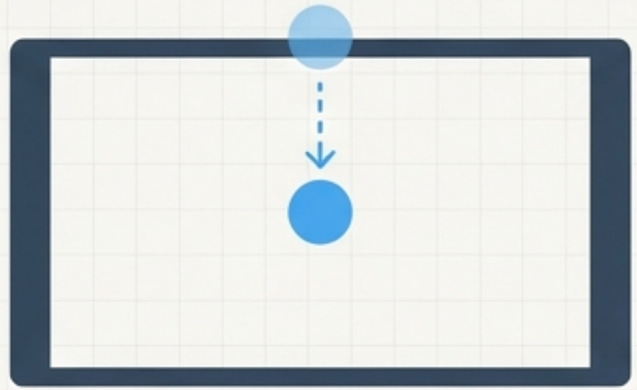

```
AnimatedContainer(
  duration: Duration(seconds: 1),
  height: counter,
  width: counter,
  color: counter % 2 == 0 ? Colors.blue : Colors.lime,
  alignment: counter < 500 ? Alignment.center : Alignment.topCenter,
);
```

# Konumlandırma ve Boşluk Yönetimi

## AnimatedAlign

Bir alt widget'ın konumunu ekranın bir bölgesinden diğerine kaydırır.

```
alignment: value % 2 == 0 ? Alignment.center :
           Alignment.topCenter
```

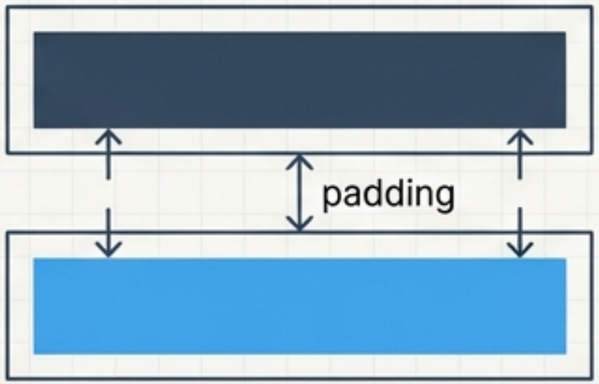


---

## AnimatedPadding

Widget'lar arasındaki dolgu/boşluk (padding) değerlerini dinamik olarak değiştirir.

```
padding: EdgeInsets.only(top: value)
```



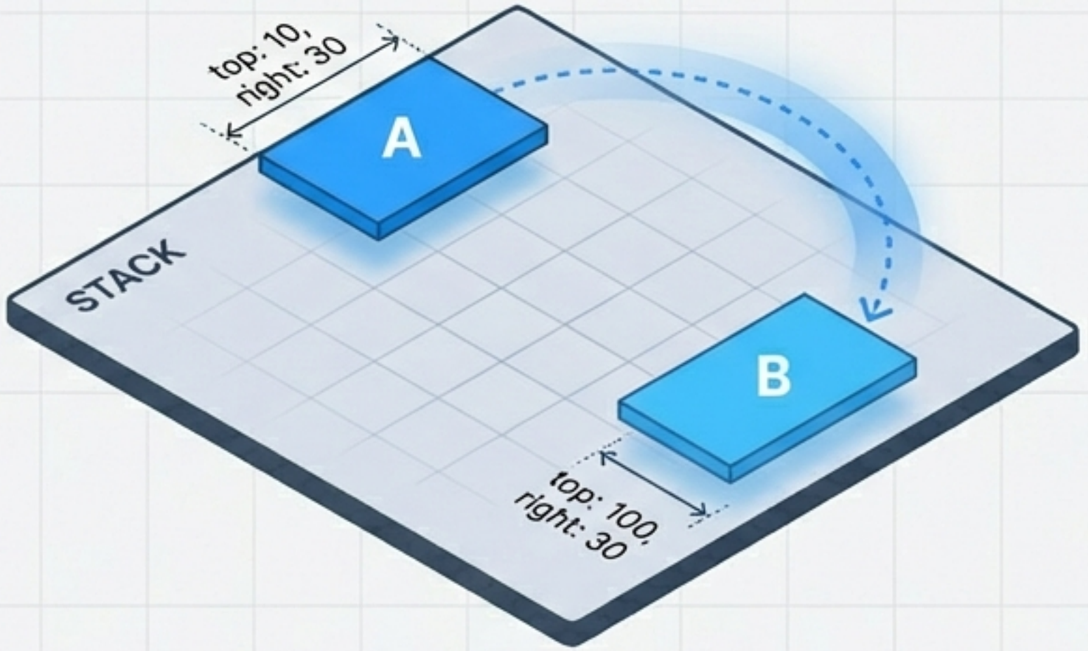

Her iki widget da duration parametresine ihtiyaç duyar ve setState tetiklendiğinde otomatik geçiş sağlar.

# Stack İçinde Hareket: AnimatedPositioned

Sadece bir Stack widget'ı içinde çalışır.

Öğenin `top`, `left`, `right`, `bottom` koordinatlarını animasyonlu şekilde değiştirir.

```
AnimatedPositioned(
  duration: Duration(seconds: 1),
  top: value,
  right: 30,
  child: ...
)
```




**Dikkat:** Animasyon sonunda alt widget'ın boyutu değişebilir. Eğer boyutların sabit kalması gerekiyorsa, bu widget yerine SlideTransition kullanmayı değerlendirin.

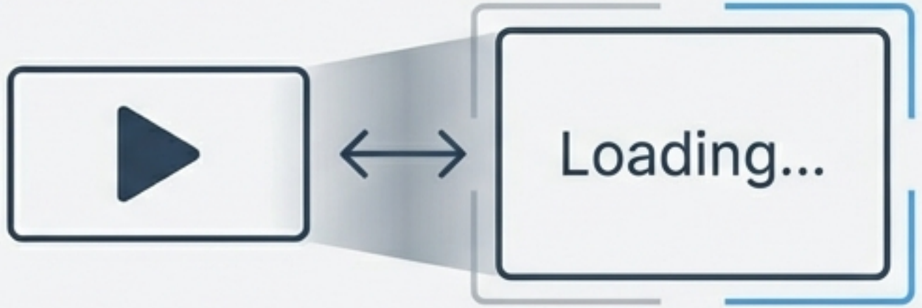
# Görünürlük ve Geçişler

## AnimatedOpacity

Opaklık değerini 0 ile 1 arasında değiştirir.

⚠️ **Performans Uyarısı:** Ara bellek (buffer) gerektirdiği için pahalı olabilir.

## AnimatedCrossFade




İki farklı widget arasında boyut farkını da gözeterek geçiş yapar.

```
crossFadeState: showFirst ?
   CrossFadeState.showFirst :
   CrossFadeState.showSecond
```

# Metin ve Tipografi: AnimatedDefaultTextStyle

Bir **Text** widget'ının stilini aniden değiştirmek yerine yumuşak bir geçiş uygular.

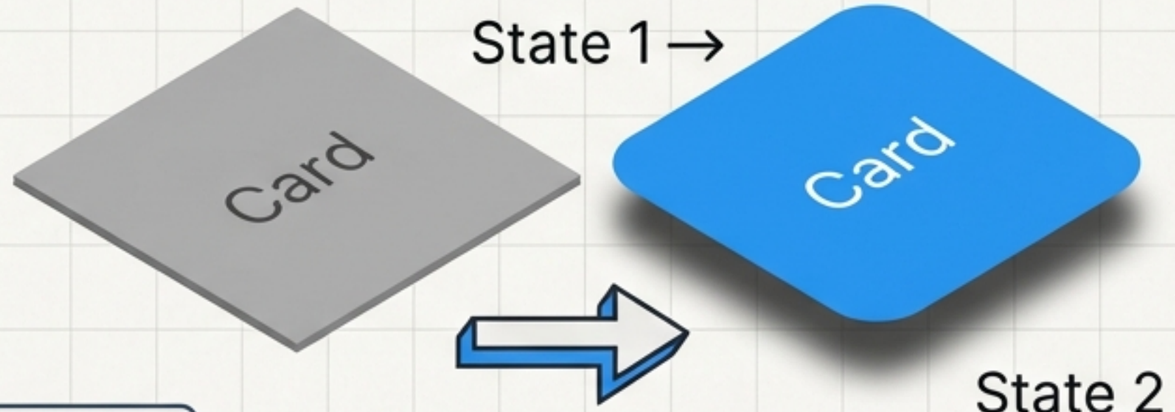
Bu widget, bir Column içindiski içindeki tüm metinlere stil uygulayabilir.

```
AnimatedDefaultTextStyle(
  duration: Duration(milliseconds: 500),
  style: TextStyle(
    fontSize: fontSize,
    color: color,
    fontWeight: FontWeight.bold,
  ),
  child: Text('Flutter'),
)
```

**Not:** textAlign, softWrap veya maxLines gibi özellikler animasyona dahil edilmez.

# Fiziksel Özellikler: AnimatedPhysicalModel

Derinlik (elevation), renk, gölge rengi ve köşe yuvarlatma (borderRadius) değerleri arasında geçiş yapar.




```
AnimatedPhysicalModel(
  elevation: value,
  color: Colors.white,
  shadowColor: Colors.black,
  borderRadius: BorderRadius.circular(value),
  animateColor: true, // Önemli!
  ...
)
```

**Key Detail:** Şekil (Shape) animasyonu desteklenmez. Arka plan rengi geçişi için `animateColor: true` yapılmalıdır.

# Dinamik Boyutlandırma: AnimatedSize

İçindeki çocuk widget'ın boyutu değiştiğinde, ebeveynin boyutlarını animasyonlu bir şekilde yeni sınırlara ayarlar. Genişleyen listeler için idealdir.

```
AnimatedSize(
  duration: Duration(seconds: 1),
  curve: Curves.easeIn,
  child: Container(height: value),
)
```

Soru 1: Flutter Nedir?

Soru 2: AnimatedSize Nasıl Çalışır?

Cevap: Boyut değişikliği tetiklendiğinde, ebeveyn widget'ın boyutları belirtilen süre ve eğriye göre animasyonlu olarak güncellenir.

# İkon Morfolojisi: AnimatedIcon

Bir ikonun şeklini değiştirerek başka bir ikona dönüşmesini sağlar.

State 1 → State 2

Gereksinim: Diğer implicit widget'lardan farklı olarak, animasyonun ilerleme durumunu (progress) yönetmek için bir **Controller** gerektirir.

```
AnimatedIcon(
  icon: AnimatedIcons.menu_arrow,
  progress: controller,
)
```

# Özet: Zahmetsiz Animasyon Algoritması


Implicit Animations, Flutter'ın 'Düşük Efor / Yüksek Ödül' mekanizmasıdır.

# Koddan Deneyime

Karmaşık matematiklere ve trigonometriye girmeden, sadece parametreleri değiştirerek profesyonel bir kullanıcı deneyimi elde edebilirsiniz.
Flutter sizin için zor işi üstlenir.

Daha fazla detay ve görsel örnekler için: flutter.dev