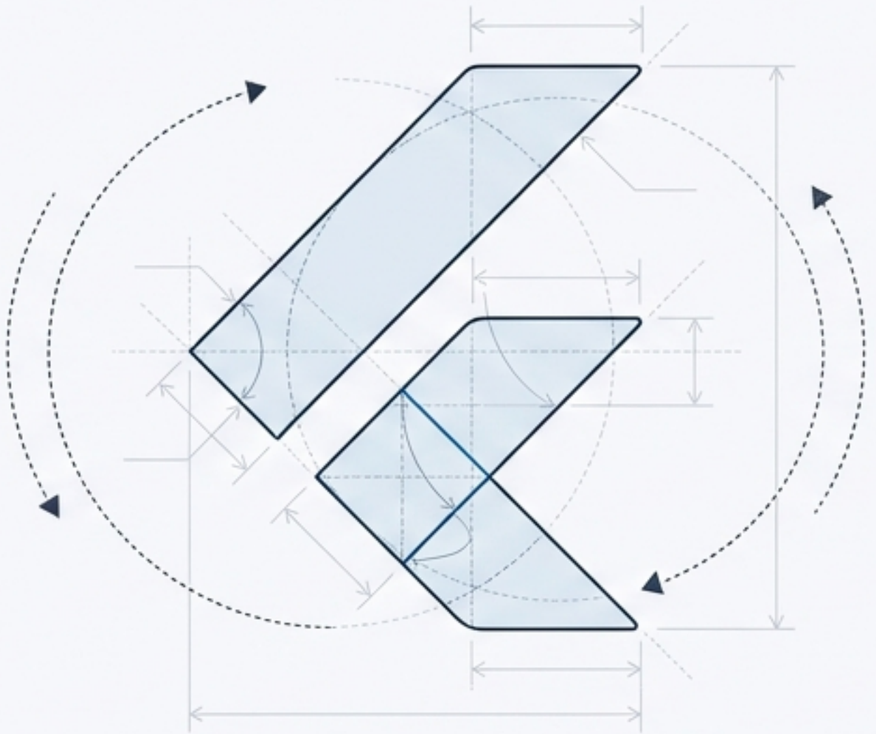
Flutter Animasyon Kütüphanesinde Ustalık

KONTROLÖRLER, İNŞA EDİCİLER VE PERFORMANS MÜHENDİSLİĞİ

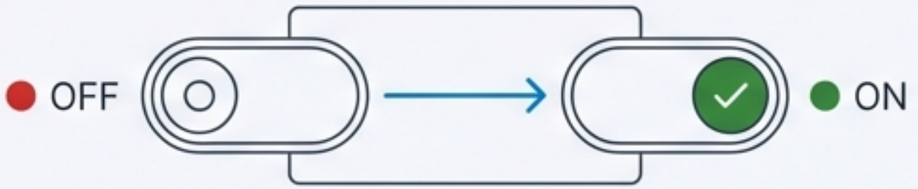


Explicit (açık) animasyonların temellerinden başlayarak, AnimationController mekaniği, AnimatedBuilder ile performans optimizasyonu ve Tweens ile estetik zenginleştirmeye giden teknik bir yolculuk.

NotebookLM

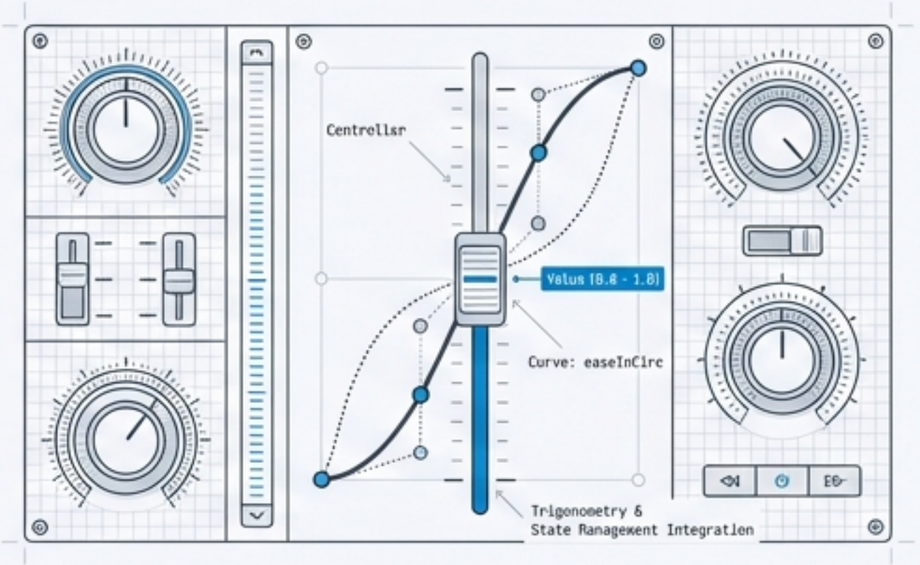
Neden Explicit (Açık) Animasyonlara İhtiyaç Duyarız?

Implicit (Hazır) Animasyonlar



Kullanımı kolaydır ancak özelleştirme yetenekleri sınırlıdır. Sadece A noktasından B noktasına gider.

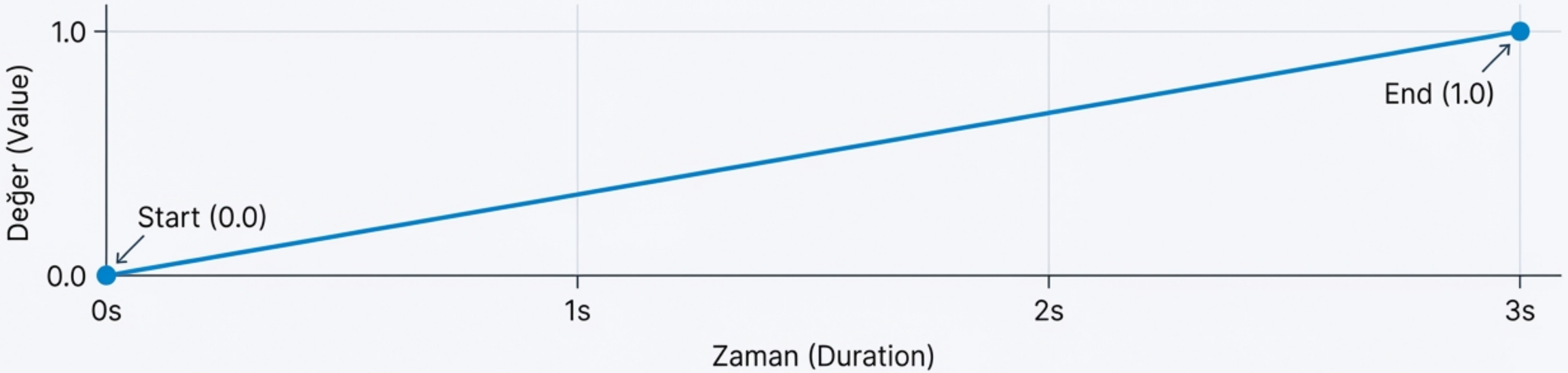
Explicit (Açık) Animasyonlar



Kontrolörler ve değerler ile çalışmayı gerektirir. Sadece başlangıç ve bitişi değil, süreci yönetmenizi sağlar. Bu noktadan itibaren trigonometri ve durum yönetimi (state management) devreye girer.

NotebookLM

Animasyonun Kalbi: AnimationController



- **Tanım:** Donanım yeni bir kare (frame) için hazır olduğunda değer üreten bir orkestra şefidir.
- **Varsayılan Davranış:** Belirlenen süre (Duration) boyunca 0.0'dan 1.0'a kadar lineer değerler üretir.
- **Süre:** duration: const Duration(seconds: 3) gibi tanımlarla animasyonun ömrü belirlenir.
- **Not:** Kontrolör tek başına görsel bir şey çizmez, sadece zamanı ve değeri yönetir.

NotebookLM

Sahne Kurulumu: State ve Vsync

```dart
class _FLSpinnerState extends State with TickerProviderStateMixin {
  late final AnimationController _controller;

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(seconds: 3),
      vsync: this, // Donanım senkronizasyonu
    );
  }

  @override
  void dispose() {
    _controller.dispose(); // Kaynak temizliği
    super.dispose();
  }
}
```

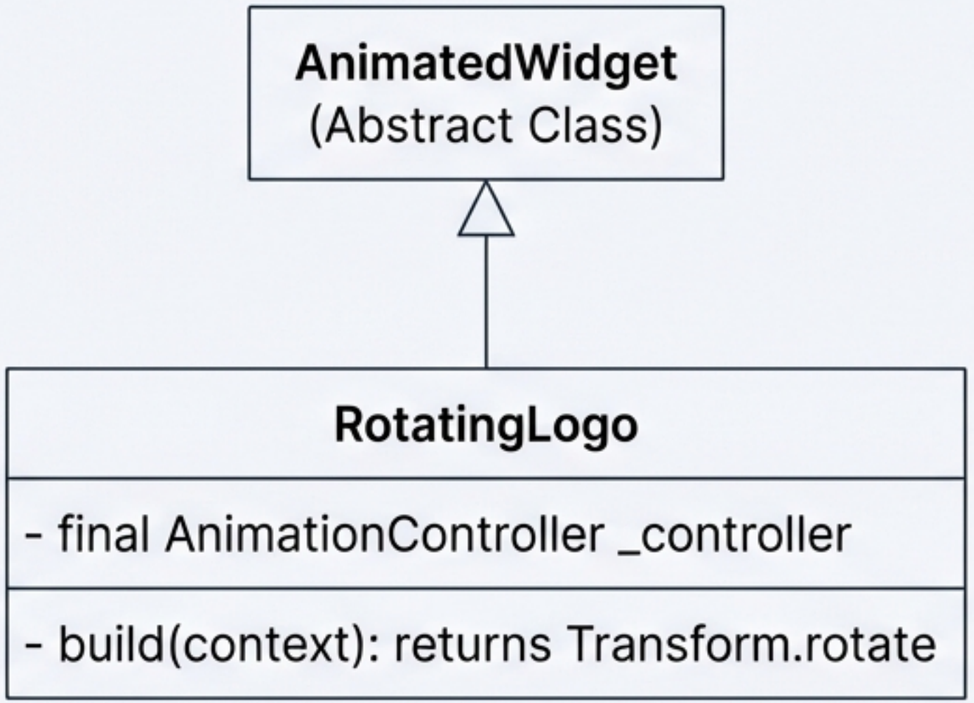
Sınıfınıza eklenmesi zorunludur. `StatefulWidget`'ın bir vsync olarak davranmasını sağlar.

Ekran dışında kalan animasyonların gereksiz kaynak tüketmesini engeller. Animasyonun donanım yenileme hızına senkronize olmasını sağlar.

Animasyonlar canlıdır. Widget yok edildiğinde, bellek sızıntısını önlemek için **_controller.dispose()** çağrılmalıdır.

NotebookLM

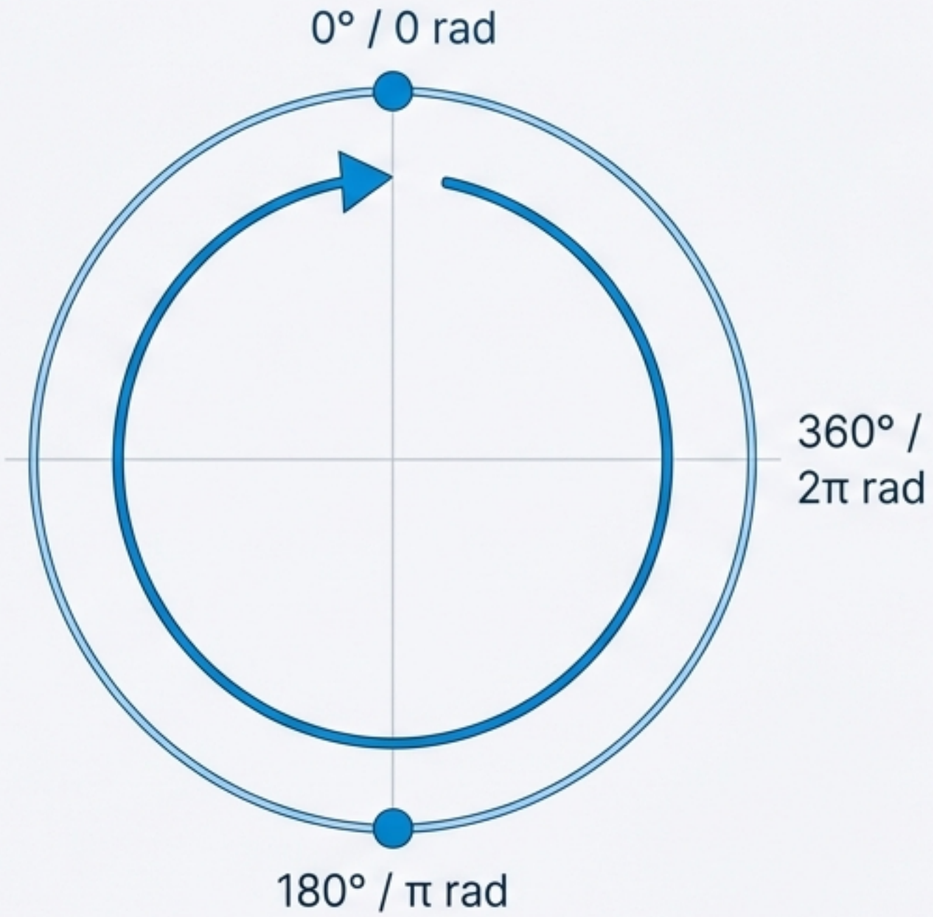
Harekete Geçiş: AnimatedWidget



- **Amaç:** Herhangi bir widget'ı alıp, animasyon yetenekleriyle donatmaktır.
- **Yapı:** AnimatedWidget sınıfından türetilir (subclassing).
- **Listenable:** Kontrolör, super(listenable: controller) aracılığıyla üst sınıfa iletilir. Bu, _controller.value her değiştiğinde widget'ın yeniden çizilmesini tetikler.
- **Bağımlılık Enjeksiyonu:** Kontrolör, widget'ın içinde oluşturulmaz, dışarıdan parametre olarak verilir.

NotebookLM

Radyanlar ve Dönüşüm Matematiği




```
Transform.rotate(
  angle: _controller.value * (2 * math.pi),
  child: ...
)
```

Transform.rotate: Widget'a hareket vermez, sadece belirli bir eğim (açı) ayarlar. Hareket, bu açının her karede güncellenmesiyle oluşur.

- Matematik: Flutter radyan kullanır. Tam bir tur (360 derece) için **2 * math.pi** formülü kullanılır.

Formül: `angle: _controller.value * _fullRotation`

Kontrolör 0'dan 1'e giderken, açı 0'dan 2π'ye (tam tur) ulaşır.

NotebookLM

Döngüyü Yönetmek: repeat, forward, reverse

forward()

[0 → 1]

Animasyonu baştan sona (0 → 1) bir kez oynatır.

reverse()

[1 → 0]

Animasyonu sondan başa (1 → 0) bir kez oynatır.

repeat()

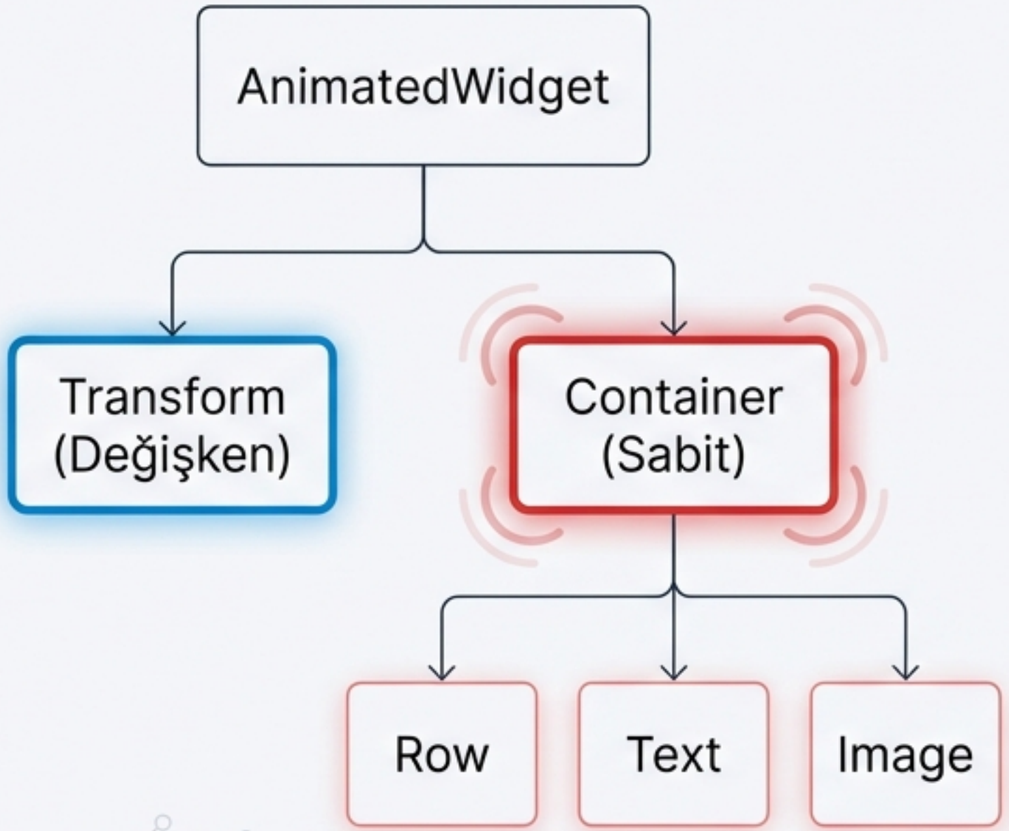
[0 → 1, 0 → 1...]

Animasyonu sonsuz bir döngüde (0 → 1, 0 → 1...) çalıştırır.

Örnek: Dönen bir logo için `_controller.repeat()` kullanılarak sonsuz hareket sağlanır.

NotebookLM

Performans Mücadelesi: 60 FPS Gerçeği



- Saniyede 60 Kare: _controller.value her değiştiğinde (saniyede 60 veya 120 kez) build metodu tetiklenir.
- Tehlike: const olmayan ve animasyondan etkilenmeyen widget'ların (Kırmızı alanlar) her karede yeniden oluşturulması kaynak israfıdır.
- Hedef: Sadece değişen kısmı (Transform) yeniden çizmek, sabit kısımları hafızada tutmak.

NotebookLM

Çözüm 1: Manuel Önbellekleme (Caching)

```dart
late final Widget _cachedTree = _buildExpensiveTree();

@override
Widget build(BuildContext context) {
  return Transform.rotate(
    angle: _controller.value * _fullRotation,
    child: _cachedTree, // Sabit referans kullanımı
  );
}
```

Yöntem

Animasyona bağlı olmayan widget ağacını bir değişkende saklamak.

Mantık

build metodu içinde `child: _cachedTree` kullanarak, Flutter'a bu parçanın değişmediğini bildiririz. bildiririz.

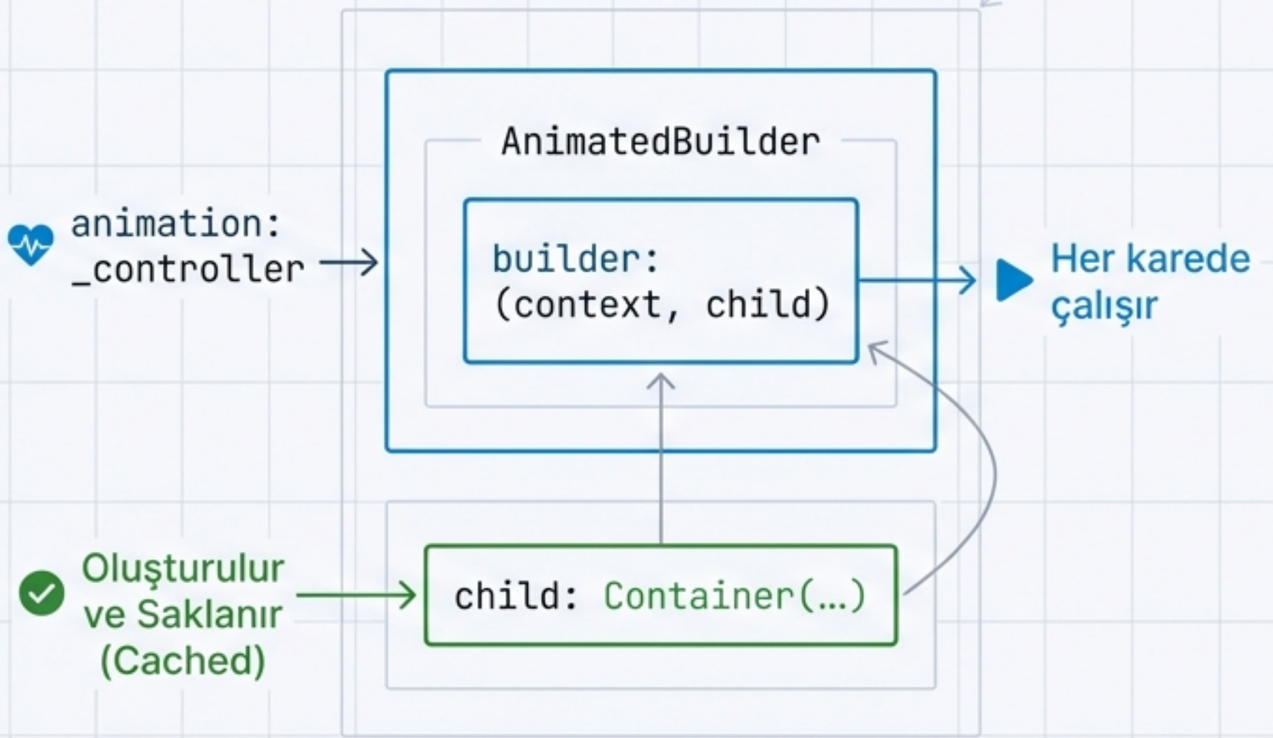
Sonuç

Framework bu parçayı her karede tekrar inşa etmez, sadece referansı kullanır. Bu, const kullanmanın manuel halidir.

NotebookLM

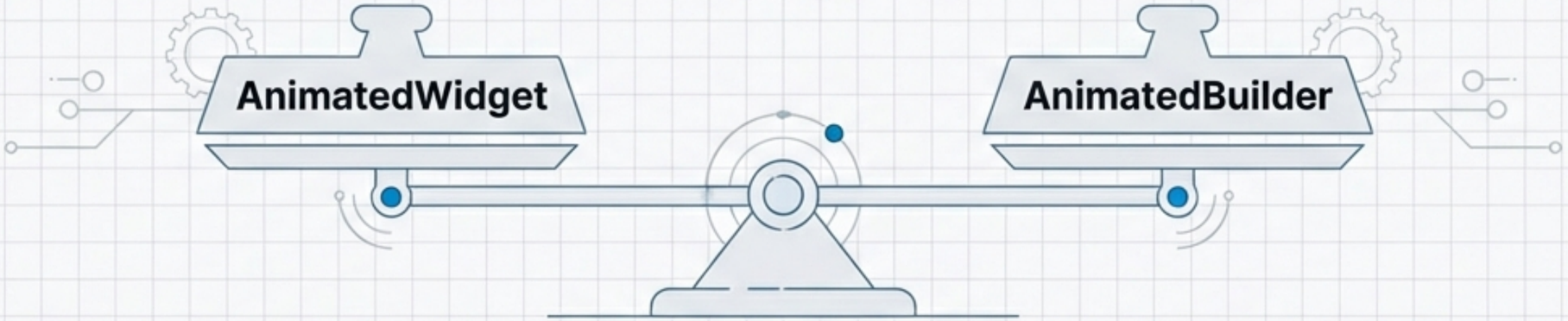
Çözüm 2: AnimatedBuilder (Önerilen Yol)

- **Nedir?:** Manuel önbellekleme işlemini sizin için otomatik yapan optimize edilmiş bir widget.
- **Child Parametresi:** Sabit kalacak (cache'lenecek) widget buraya verilir. Sadece bir kez oluşturulur.
- **Builder Parametresi:** Animasyon mantığını içerir ve child parametresini referans olarak alır.
- **Avantaj:** Daha az kod (boilerplate), daha yüksek okunabilirlik ve garantili performans.



NotebookLM

Karşılaştırma: Hangisini Seçmeli?



AnimatedWidget

- Sınıf türetmeyi (extends) gerektirir.
- Kodun modüler olmasını sağlar.
- Büyük ve tekrar kullanılan bileşenler için idealdir.

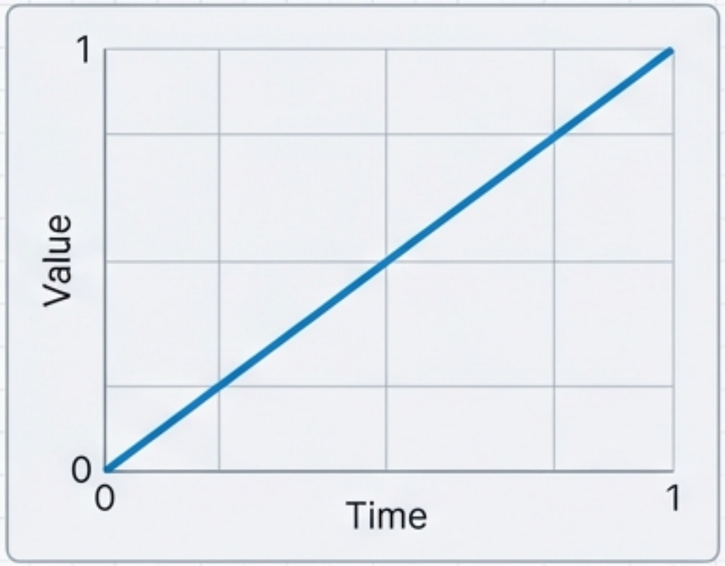
AnimatedBuilder

- Kodun akışı içinde (inline) kullanılabilir.
- Temelinde bir AnimatedWidget'tır ve otomatik ✅ caching sağlar.
- Daha temiz kod ve daha az boilerplate.

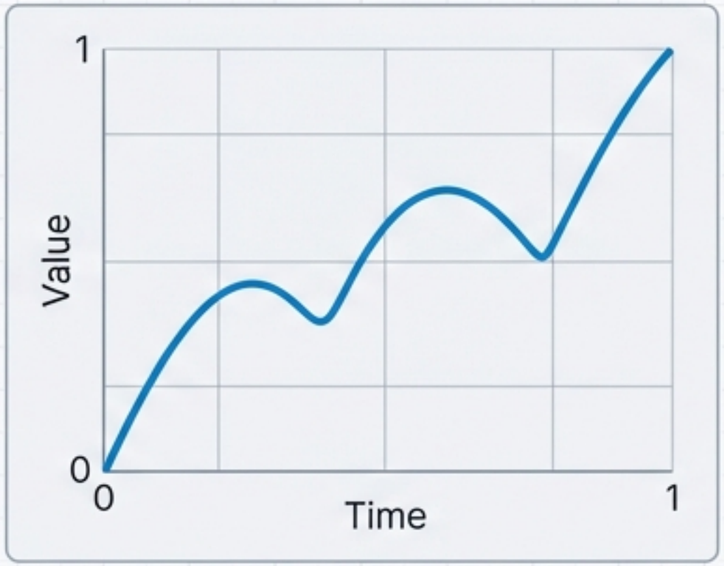
Karar: İkisi de aynı işi yapar. Ancak AnimatedBuilder genellikle daha temiz kod ve otomatik optimizasyon sağladığı için modern Flutter geliştirmesinde öncelikli tercihtir. ✅

NotebookLM

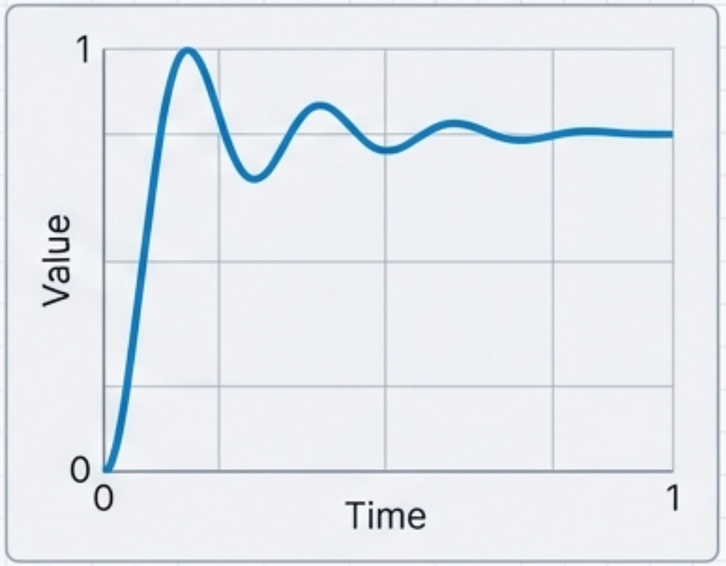
Fizik ve His: Curves (Eğriler)



Linear
Robotik



BounceIn
Sekme Efekti



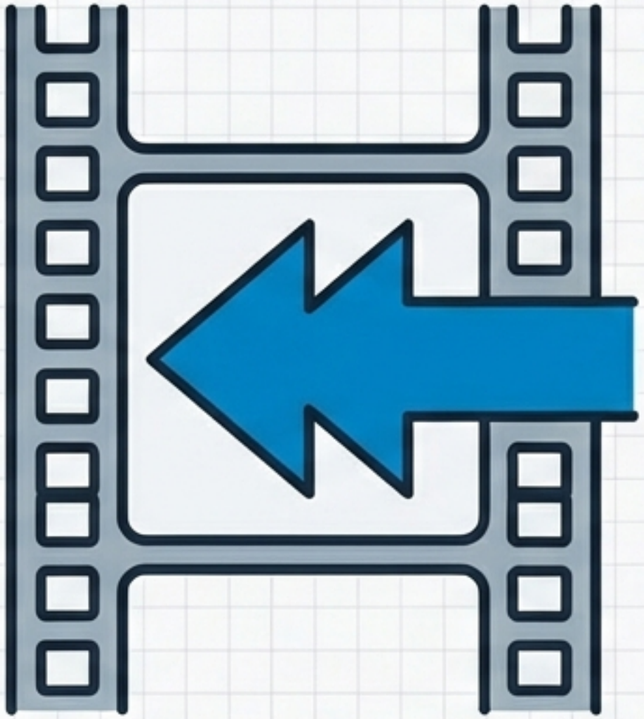
ElasticOut
Elastik Yaylanma

- **Sorun:** Varsayılan animasyonlar doğrusaldır (sabit hız), bu da robotik hissettirir.
- **Çözüm:** CurvedAnimation kullanarak hızı manipüle etmek.
- **Kullanım:** Kontrolör yerine CurvedAnimation nesnesi widget'a verilir.

```
curve: Curves.elasticOut
```

NotebookLM

Yön Değiştirme: ReverseAnimation



Senaryo: Bir animasyonu veya eğriyi tam tersi yönde çalıştırmak istiyorsunuz.

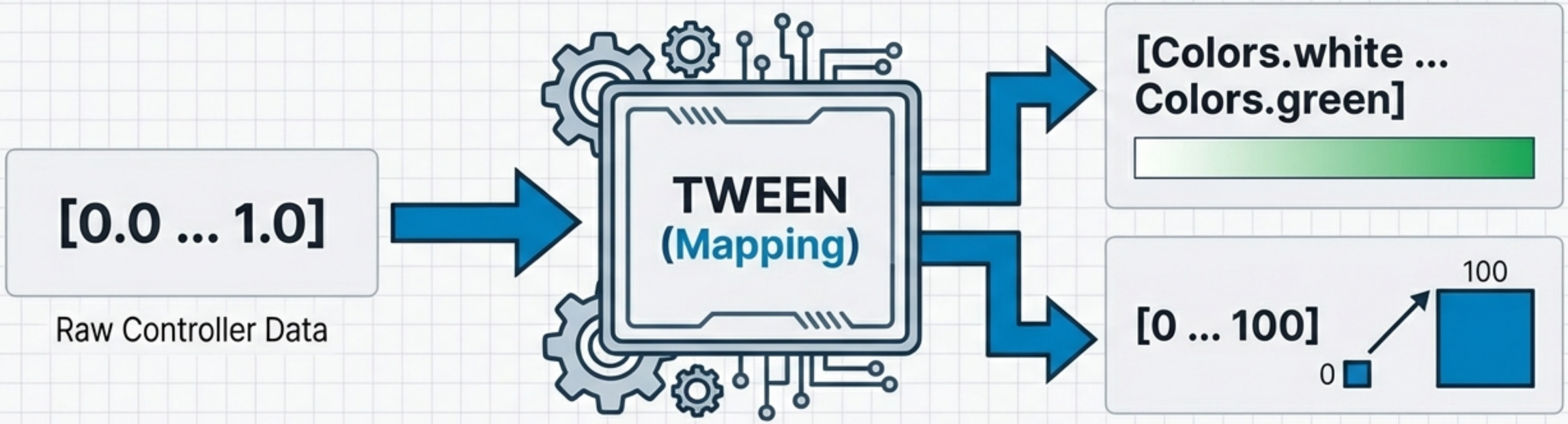
Kullanım: ReverseAnimation(_controller) veya ReverseAnimation(_curved).

```
ReverseAnimation(_controller) veya
ReverseAnimation(_curved)
```

Fark:
controller.reverse() tek seferlik bir eylemken, ReverseAnimation sarmalayıcısı (wrapper) ile sonsuz döngüde geriye doğru oynatma (repeat) yapılabilir.

NotebookLM

Sınırsız Olasılıklar: Tweens



- **Tanım:** 'Between' kelimesinden gelir. 0-1 aralığını istenilen değer aralığına (Mapping) dönüştürür.
- **Matematik:** Artık 2*pi ile çarpmaya gerek kalmaz, Tween(begin: 0, end: 2*math.pi) bu işi yapar.
- **Bağlama:** tween.animate(_controller) kullanılarak kontrolörden gelen ham zaman verisi, anlamlı verilere (renk, boyut, sınır) dönüştürülür.

NotebookLM

Özet ve Kilit Çıkarımlar

- **Kontrol (Engine):** AnimationController zamanı yönetir, dispose ile temizlenmelidir.
- **Hareket (Motion):** AnimatedWidget ve Transform veriyi görsele çevirir.
- **Performans (Optimization):** AnimatedBuilder, sabit widget'ları (child) önbelleğe alarak 60 FPS'i korur.
- **Estetik (Polish):** Curves doğallık katar, Tweens ise sayıları renklere ve şekillere dönüştürür.

"Animasyon sadece hareket değil, uygulamanıza hayat verme ve kaynakları verimli kullanma sanatıdır."

NotebookLM