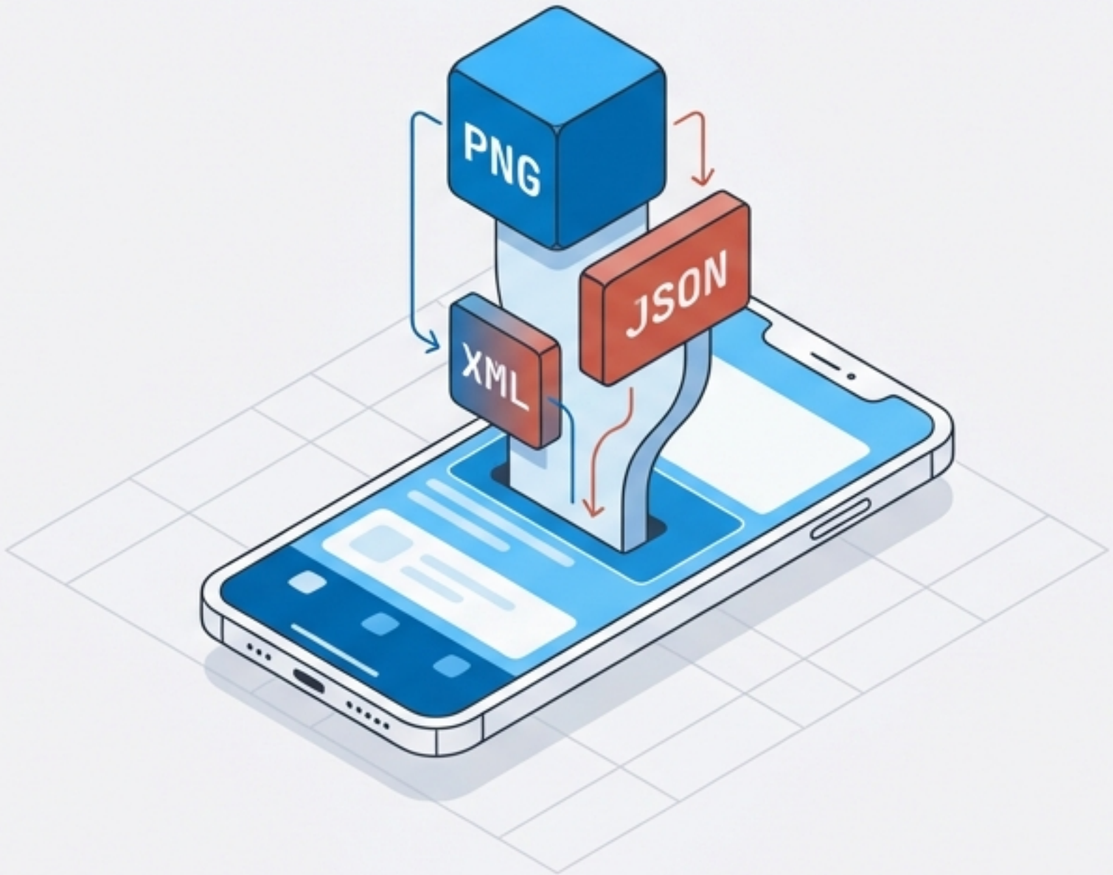
# Flutter'da Asset ve Görüntü Yönetimi

## Dosyaları Dahil Etme, Kullanma ve Optimize Etme Sanatı

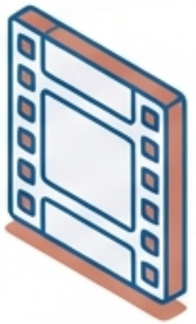



Kendi Kendine Öğrenme Rehberi

# Kodun Ötesindeki Ruh: Asset (Varlık) Nedir?

Flutter uygulamaları sadece derlenmiş koddan ibaret değildir. Uygulamanın çalışması için gereken statik veriler, binary dosya ile birlikte paketlenir. Bu dosyalar çalışma zamanında (runtime) herhangi bir kısıtlama olmaksızın kullanılabilir.

**Görseller**
(JPG, PNG, GIF)



**Multimedya**
(Videolar)

**Veri Dosyaları**
(JSON, XML, TXT)

**Veritabanları**
(SQLite)

# Gümrük Kapısı: pubspec.yaml Kayıt Defteri

Bir dosyanın kod içinde kullanılabilmesi için önce pubspec.yaml dosyasında beyan edilmesi gerekir. Bu, Flutter'ın dosyayı paketlemesi için zorunlu adımdır.

Önemli: Tanımlanmayan dosyalar derleme sırasında yoksayılır ve uygulamaya dahil edilmez.

Yaklaşım A: Klasör Bazlı

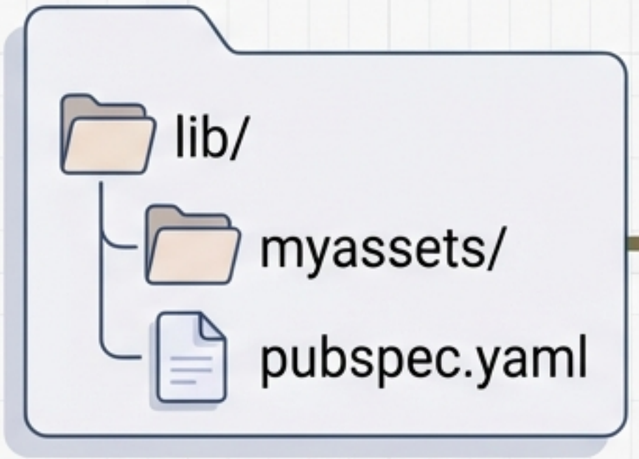
```yaml
flutter:
  assets:
    - myassets/ # Tüm klasörü dahil eder
```

Yaklaşım B: Dosya Bazlı

```yaml
flutter:
  assets:
    - myassets/logo.png
    - myassets/client_config.json
```

# Erişim Mantığı ve Klasör Yapısı

Yollar (Paths) her zaman proje kök dizinine (root) göre belirlenir. Dizinler derleme zamanında (build time) otomatik olarak çıkarılamaz, bu nedenle tam yol belirtilmelidir.




**Yanlış Kullanım**

```
load('file.txt')
```

**Doğru Kullanım**

```
load('myassets/file.txt')
```

**İpucu:** Yol tanımları slash (/) ile başlamak zorunda değildir, ancak kök dizine göre tam adresi içermelidir.

# Erişim Anahtarı: AssetBundle Sınıfı

Kodunuzdan varlıklara erişmek için AssetBundle sınıfı iki temel yöntem sunar.

## `loadString(String path)`

Metin tabanlı dosyalar (TXT, JSON, XML) için kullanılır.

**Dönüş:** `String`

## `load(String path)`

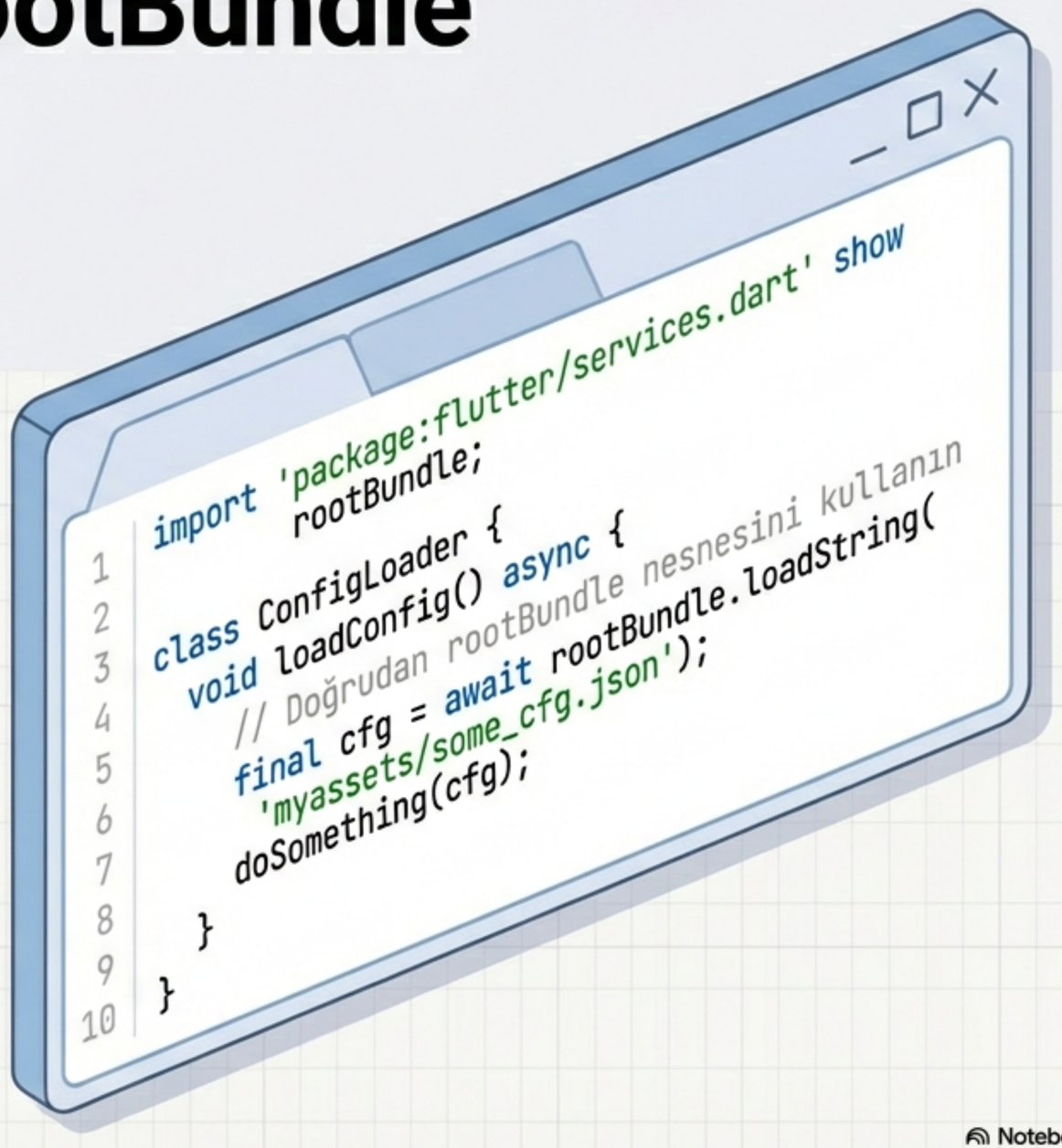

Binary dosyalar (Resim, Veritabanı) için kullanılır.

**Dönüş:** `ByteData`

# Widget Dışı Erişim: rootBundle

Bir StatelessWidget veya StatefulWidget bağlamının dışında (örneğin bir Model sınıfında) veri yüklemeniz gerektiğinde kullanılır.

**Uyarı:** rootBundle her Flutter uygulamasında erişilebilir olsa da, Widget yapısı içindeyken önerilen yöntem bu değildir.



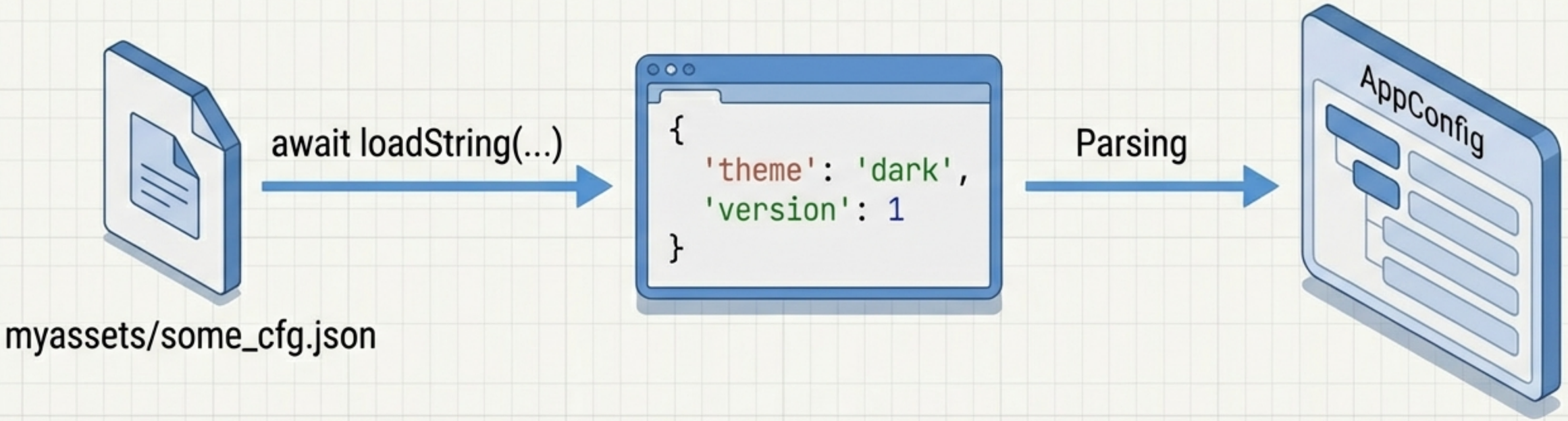
# Widget İçi Erişim: DefaultAssetBundle

Bir Widget içinde varlık yüklerken rootBundle yerine DefaultAssetBundle tercih edilmelidir. Bu yaklaşım, tema (Theme) ve yerelleştirme gibi context'e duyarlı yapılarla daha uyumludur.

Her iki yöntem de aynı tür nesneyi döndürür, ancak DefaultAssetBundle mimari açıdan daha esnektir.

# Modern Konfigürasyon Yönetimi: JSON Yükleme

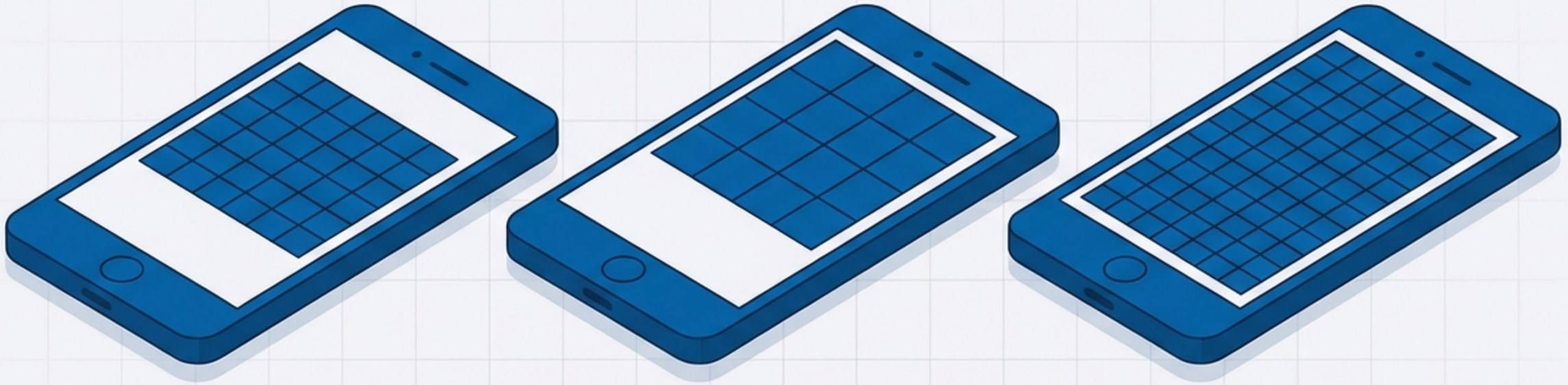
Uygulama konfigürasyonlarını kod içine gömmek yerine JSON dosyalarında saklamak ve asenkron olarak yüklemek yaygın bir yöntemdir.




Not: Basit veriler için Shared Preferences kullanımı da alternatif bir yöntemdir.

# Çözünürlük Duyarlılığı: Flutter’ın Sihri

Tek bir görsel her ekrana uymaz. Flutter, cihazın Piksel Oranı'na (Pixel Ratio) göre en iyisini seçer.



1.0x (Base)

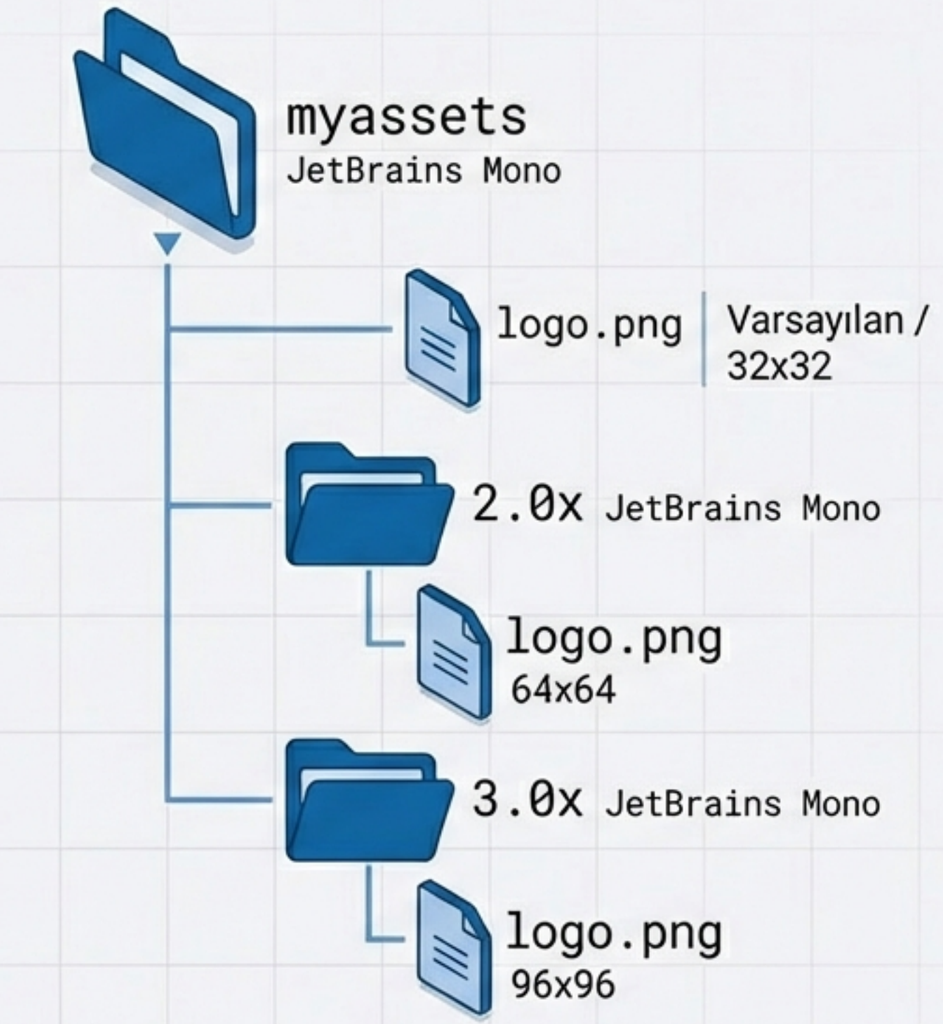
2.0x (Retina)

3.0x (High Density)

Geliştirici farklı boyutları hazırlar,
Flutter otomatik seçim yapar.

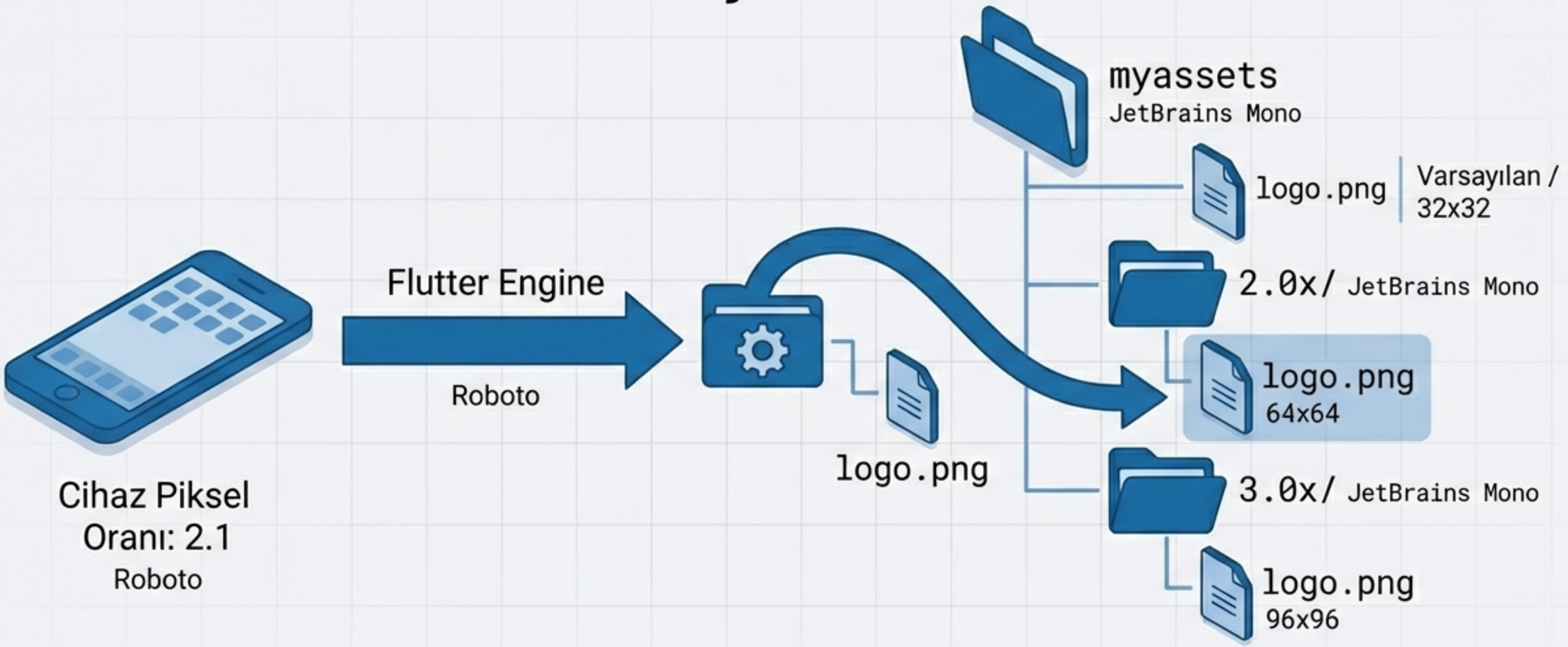
# Klasör Hiyerarşisi Stratejisi

Flutter’ın doğru görseli seçebilmesi için belirli bir klasör yapısına uyulması zorunludur. Dosya isimleri aynı kalmalı, oranlar klasör isimlerinde belirtilmelidir.




Kural: Klasör isimleri Rx/ formatında olmalıdır (Örn: 1.5x/, 2.0x/).

# Otomatik Seçim Mekanizması




**Kalite Uyarısı:** Runtime Resizing (çalışma zamanı boyutlandırma) yerine Asset Variants (hazır varyantlar) kullanmak, bulanıklığı önler ve en keskin görüntüyü garanti eder.

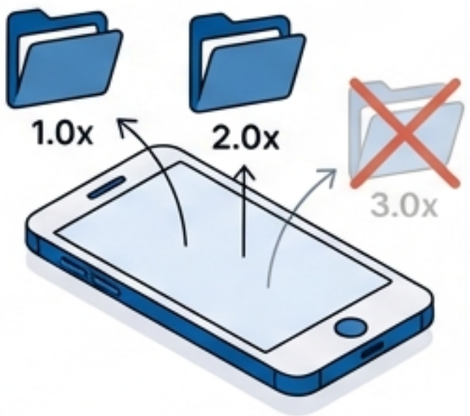
Roboto

# Performans ve Boyut Yönetimi

Uygulamaya eklenen her asset, nihai dosya boyutunu (APK/IPA) artırır.

## 1. Varyant Sayısı

Gereksiz varyantlardan kaçının; hedef cihaz kitlesine göre karar verin.

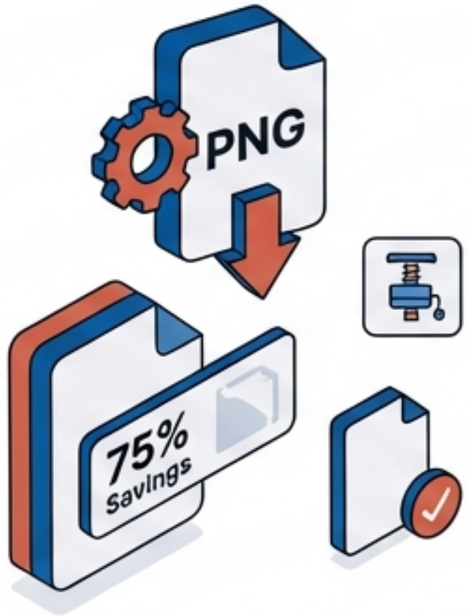



Flutter Blue #02569B

Terracotta #D85D48

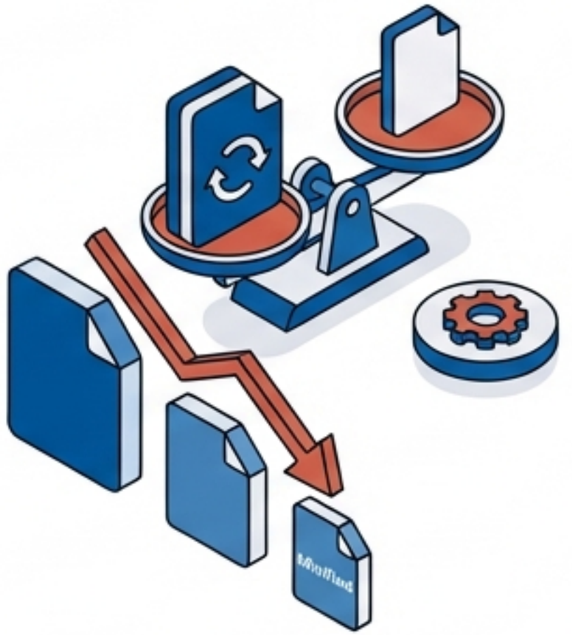
## 2. PNG Sıkıştırma

PNG dosyalarını özel araçlarla optimize edin. %75'e varan boyut tasarrufu mümkündür.

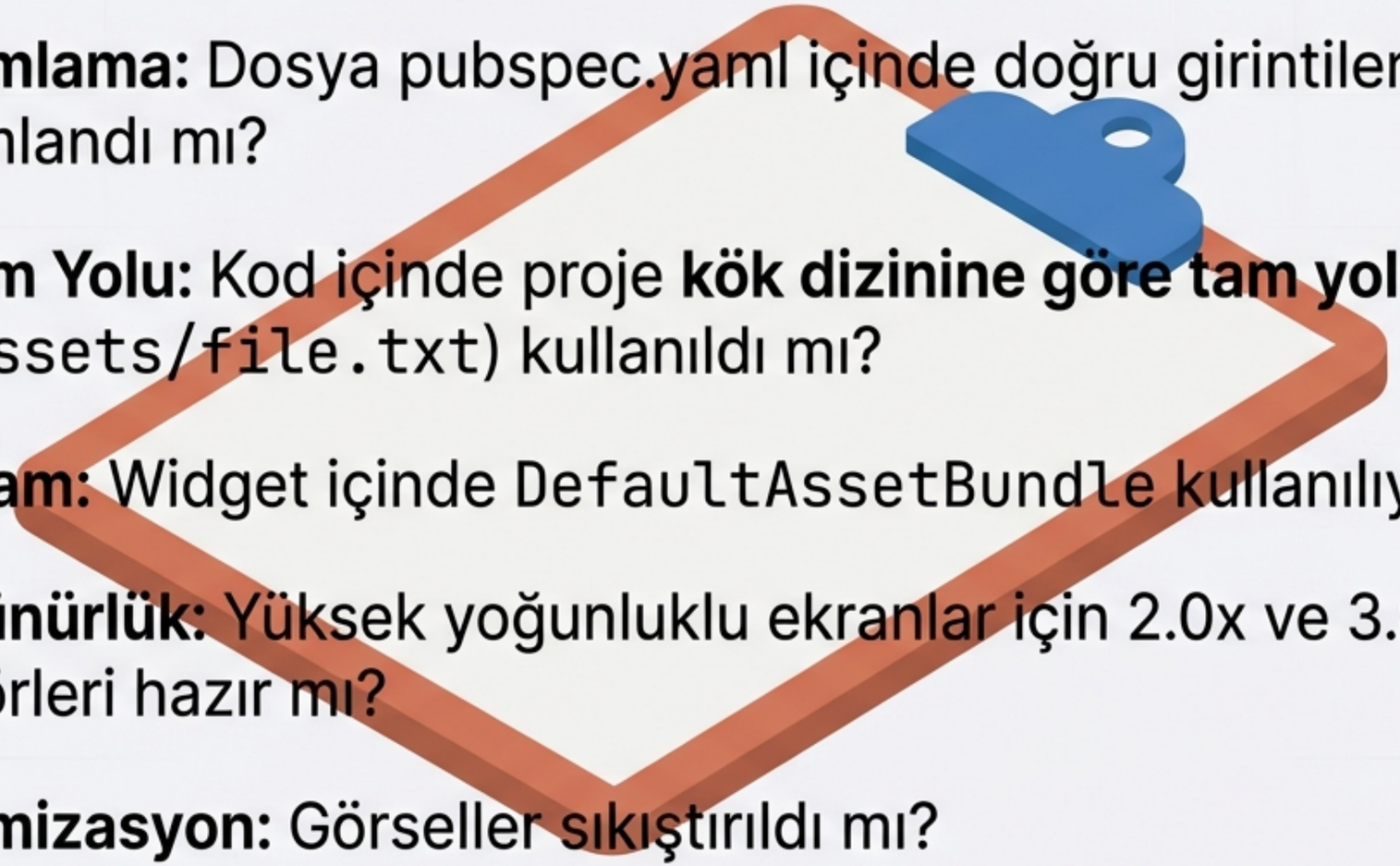



## 3. Minification

Asset boyutlarını küçültmek için minification tekniklerini kullanın.



# Kalkış Öncesi Kontrol Listesi



- [ ] **Tanımlama:** Dosya pubspec.yaml içinde doğru girintilerle tanımlandı mı?
- [ ] **Erişim Yolu:** Kod içinde proje **kök dizinine göre tam yol** (`myassets/file.txt`) kullanıldı mı?
- [ ] **Bağlam:** Widget içinde `DefaultAssetBundle` kullanılıyor mu?
- [ ] **Çözünürlük:** Yüksek yoğunluklu ekranlar için 2.0x ve 3.0x klasörleri hazır mı?
- [ ] **Optimizasyon:** Görseller sıkıştırıldı mı?

# Sonuç: Mükemmel Görüntü, Yüksek Performans

Asset yönetimi, sadece dosyaları bir klasöre kopyalamak değildir. Doğru yapılandırma, bağlama uygun erişim ve çözünürlük duyarlılığı, profesyonel bir Flutter uygulamasının temel taşlarıdır.

Artık daha akıllı, daha hızlı ve görsel olarak kusursuz uygulamalar geliştirmek için hazırsınız.