**Stream**

Bir kaynaktan bize veriler gelebilir ve ne zaman, kaç tane geleceğini bilemeyiz bu durumlarda stream kullanılabilir. Örneğin API’lardan istekler alıyor,sensörlerden veriler alıyor fakat bu veriler anında bize yanıt döndürmüyor ve bu yanıt hazır olduğunda bize dönmesi için asenkron programlama stream kullanılır.

**1-Network, Api veya paketlerden**

**2-Stream döndüren bir asenkron fonksiyondan**

**3-StreamController sınıfından**

programımıza stream, veriler gelebilir.

**Future’dan Farkı?**

Future da bize bir zamanda bir veri dönecek ve işlemler yapılacak. Ama Stream de ise birden çok arka arkaya verilerin akışı var. İkiside asenkron işlemlerdir.

**StreamBuilder**

Verileri daima dinleyen bir widgettır ve streamden gelen veriler hazır olduğunda her yeni veri geldiğinde, builder fonksiyonu tekrar çağırıp widget ağacını tekrar yaratıyor. Bu sayede yeni verileri görebiliyoruz.

StreamBuilder bize çeşitli bağlantı durumlarında (snapshot.ConnectionState) , veri olmadığında (!snapshot.hasData) veya hata ile karşılaştığında (snapshot.hasError) ne olacağına kadar tam kontrol sağlıyor.

StreamBuilder da önemli olacak nokta ise Stream objesini siz kapatmadıkça o verileri dinleyecektir ve ağır bir yüke neden olacaktır.

**FutureBuilder dan farkı ise** FutureBuilder yalnızca build anında bir defa tetiklenir ve veri ekrana basıldıktan sonra veri değişse dahi FutureBuildera yansımaz, yeni veriye ulaşılmaz.

**Async vs Async\***

Async yalnızca bir tane veri döndürürken(Future), async\* ise birden fazla veriyi farklı zamanlarda dinler ve döndürür yani Streamdir.

**Return vs Yield**

Return bir fonksiyonu sonlandırırken , yield değeri döndürür,saklar ve fonksiyonu çağırma devam eder.

Örnek;

Stream döndüren bir asenkron fonksiyon kullanımı;

Listen metodu bize StreamSubscription<dynamic> döndürür.



StreamController örneği;

