QUERY

React Query, öncelikle React tabanlı uygulamalarda veri getirme (data fetching), önbellekleme (caching), hata yönetimi ve senkronizasyon gibi işlemleri kolaylaştıran güçlü bir kütüphanedir. Modern web ve mobil uygulamalarda API ile etkileşim kurmak kaçınılmazdır. API isteklerini doğru yönetmek, veriyi önbelleğe almak ve güncel tutmak önemli görevlerdir. React Query bu zorlukları çözmek için tasarlanmıştır.

React Query, HTTP isteklerini yönetirken birçok problemle başa çıkmamıza yardımcı olur:

* API'den veri çekmek
* Veri önbellekleme
* Veri senkronizasyonu
* Yükleme ve hata durumlarının otomatik yönetimi
* İstek tekrarları ve yeniden deneme politikaları

React Query özellikle **RESTful** API'lerle ya da **GraphQL** gibi veri kaynaklarıyla çalışırken büyük kolaylık sağlar.

**Temel Kavramlar**

**Query**:

-> Bir "query", dış bir veri kaynağından veri çekmek için kullanılan bir işlem olarak tanımlanabilir. Bir API'ye veri isteği gönderir ve bu isteğin sonucunda gelen veriyi yönetir.

-> React Query, bir query'nin nasıl çalışacağını tanımlayan fonksiyonlara dayanır. Bu sorgularla ilgili tüm yükleme, hata ve başarılı olma durumlarını yönetir.

**Mutation**:

-> "Mutation" veri üzerinde bir değişiklik yapmak için kullanılan işlemlerdir (örneğin, POST, PUT, DELETE gibi istekler). Bir sorgudan farkı, bir mutation'ın durumu güncellemesi ve etki yaratmasıdır.

-> React Query mutation işlemlerini yönetmek için useMutation hook'u sağlar.

**Query Key**:

-> Query'yi benzersiz kılan bir anahtar (key) kullanılır. Bu key, React Query'nin verileri önbelleğe almasını ve aynı key ile yapılan sorguların tekrarlanmasını engellemesini sağlar.

**Query Function**:

-> Query’nin veriyi nasıl alacağını belirleyen fonksiyondur. Genellikle API istekleri bu fonksiyon içinde yapılır. Fonksiyon, asenkron olmalı ve bir "promise" döndürmelidir.

**Caching (Önbellekleme)**:

-> React Query otomatik olarak veriyi önbelleğe alır ve belirli bir süre boyunca aynı sorgular çalıştırıldığında veriyi önbellekten getirir. Bu da API isteklerinin sayısını azaltır ve kullanıcı deneyimini iyileştirir.

**Stale-Time & Refetching**:

-> **Stale-Time**: Verinin ne kadar süreyle "geçerli" (fresh) sayılacağını belirleyen süredir. Bu süre geçtikten sonra veri "eski" (stale) kabul edilir ve yeniden sorgulanır.

-> **Refetching**: Verinin geçerliliğini yitirdiği durumda yeniden sorgulanması işlemidir.

### React Query Kullanımı

Öncelikle React Query kütüphanesini kurmamız gerekir.

*npm install @tanstack/react-query*

Daha sonra uygulamamuzda QueryClient ve QueryClientProvider kullanarak React Query'yi başlatmamız gerekir.

useQuery hook'u, API'den veri çekmek için kullanılır.

React Query sadece veri çekmekle kalmaz, aynı zamanda veri üzerinde değişiklik yapmamıza da olanak tanır. Bunun için useMutation kullanılır.

### React Query'nin Sağladığı Önemli Özellikler

**Önbellekleme ve Senkronizasyon**: React Query her sorguyu otomatik olarak önbelleğe alır ve veriyi en optimal şekilde yeniden senkronize eder.

**Yükleme, Hata ve Başarı Durumlarını Yönetme**: isLoading, error, isFetching gibi durumlarla veri çekme işleminin farklı aşamalarını yönetebiliriz.

**Manuel Yenileme (Refetch)**: Bir veri güncellendikten sonra, React Query'nin otomatik olarak bu veriyi güncellemesini sağlayabilir veya manuel olarak refetch fonksiyonu ile veriyi yeniden çekebiliriz.

**Query Key ile Tekrarlanan İstekleri Engelleme**: Aynı sorgu anahtarı ile yapılan istekler önbellekten gelir, gereksiz API çağrıları engellenir.

**Arka Planda Veri Güncelleme**: Veriyi yeniden yüklemeye gerek kalmadan, arka planda güncelleyerek kullanıcıya her zaman en güncel veriyi sunabilirsiniz.

**Özetle;**

React Query, özellikle modern web uygulamalarında veri yönetimi, önbellekleme, yükleme durumu, hata yönetimi gibi kritik konularda büyük bir kolaylık sağlar. Sorgu tabanlı veri çekme işlemlerinde oldukça esnektir ve büyük projelerde performansı artırır. React Query, veri çekme işlemlerini karmaşıklaştırmadan yönetmemizi sağlayan, oldukça popüler ve güçlü bir kütüphanedir.