Hooks'un Amacı

- Hooks, React 16.8 sürümünde tanıtılmıştır ve sınıf bileşenleri yazmanın getirdiği zorlukları hafifletir.
- Fonksiyon bileşenleri ile sınıf bileşenlerinin aynı yeteneklerine sahip olmanızı sağlar; yani, **state** (durum) erişimi sağlar.

Hooks'un Avantajları

- Daha okunabilir kod yazmanıza olanak tanır.
- Kod parçaları daha yönetilebilir hale gelir.
- **Performance** (performans) artışı sağlar ve karmaşık kullanıcı arayüzü davranışlarını daha basit bir şekilde yazmanıza yardımcı olur.

En İyi Uygulamalar

- Hooks yalnızca fonksiyon bileşenleri ile kullanılmalıdır.
- Bir hook, bileşen ağacının en üst seviyesinde çağrılmalıdır; yani, alt bileşenlerde çağrılamaz.
- Normal JavaScript fonksiyonları içinde veya döngüler, koşullu ifadeler ya da iç içe fonksiyonlar içinde çağrılamaz.

Yaygın Kullanılan Hooks

- **useState**: Fonksiyon bileşenlerinde durum erişimi sağlar.
- **useEffect**: Yan etkileri yönetir, örneğin, veri çekme işlemleri.
- **useContext**: **Context** (bağlam) değişikliklerini yönetir.
- **useReducer**: Redux durum değişikliklerini yönetir.

Özel Hooks

- Özel kancalar, uygulamanıza benzersiz işlevsellik eklemenizi sağlar ve genellikle 'use' ile başlar (örneğin, **useLocalStorage** veya **useAuthentication**).

UseEffect ve SiteEffects Tanımı

- **UseEffect**: React'te işlevsel bileşenlerde yan etkileri (side effects) gerçekleştirmek için kullanılan bir React hook'udur.
- **SiteEffect**: Sayfa yüklendiğinde hemen gerçekleştirilmesi gereken işlemleri ifade eder; örneğin, bir API'den veri çekmek, olaylara abone olmak veya DOM'u manipüle etmek.

UseEffect Kullanımı

- **State Management with UseState**: Bileşen, **UseState** hook'unu kullanarak bir state değişkeni tanımlar. Örneğin, const [foods, setFoods] = useState([]); ile bir gıda listesi için başlangıçta boş bir dizi oluşturulur.

- **Data Fetching**: **UseEffect** içinde bir fetch isteği yapılarak API'den veri alınır. Gelen yanıt, response.json() ile JSON formatına dönüştürülür ve ardından setFoods ile state güncellenir.

Dependencies (Bağımlılıklar)

- **Empty Dependency Array**: Eğer bağımlılık dizisi boş bırakılırsa, etki yalnızca bileşen ilk yüklendiğinde çalışır.
- **Specific Dependencies**: Belirli değerler sağlandığında, etki yalnızca bu değerler değiştiğinde çalışır. Örneğin, count değişkeni bağımlılık dizisine eklendiğinde, yalnızca count güncellendiğinde etki yeniden çalışır.

Custom Hook Kullanımı

- **Custom Hook**: React'te karmaşık mantığı yeniden kullanılabilir hale getirmek için özel hook'lar oluşturulabilir. Örneğin, **UseToggle** adında bir custom hook, bir butonun durumunu yönetmek için kullanılabilir.

Dış Hizmetler (External Services)

- Dış hizmetler, uygulamanızın diğer uygulamalarla bağlantı kurmasını sağlayan programlar veya platformlardır.
- Uygulamanızın ihtiyaç duyduğu özellikleri veya araçları sağlayan hizmetlerdir.

API Kullanımı (Using APIs)

- Uygulama programlama arayüzleri (APIs), uygulamanızın dış hizmetlerle veri alışverişi yapmasını sağlar.
- Üçüncü taraf hizmetler, verileri almak için Fetch API veya Axios kütüphanesi gibi yöntemler sunar.

Fetch API ile Veri Alma (Fetching Data with Fetch API)

- fetch metodu, dış API'ye GET isteği göndermek için kullanılır.
- then metodu, fetch işlemi başarılı olduğunda çalışacak bir geri çağırma fonksiyonu alır ve yanıt nesnesinin JSON metodunu çağırarak veriyi işler.

Axios ile Veri Alma (Fetching Data with Axios)

- Axios, HTTP istekleri için popüler bir JavaScript kütüphanesidir.
- axios.get metodu, belirtilen URL'ye GET isteği yapmak için kullanılır ve otomatik JSON ayrıştırma gibi ek özellikler sunar.

Hata Yönetimi (Error Handling)

- catch metodu, fetch işlemi sırasında oluşabilecek hataları yönetmek için kullanılır.
- Hata mesajları, hata ayıklama sürecinde yardımcı olmak için konsola yazdırılır.

Formların Temel Amacı

- Formlar, kullanıcıların web sayfasında veri ile etkileşimde bulunmasını sağlar. Örneğin, kullanıcı kaydı, anketler veya sipariş verme gibi durumlarda kullanılır.
- Kullanıcıların veri girişi yaptığı alanlara "fields" (alanlar) denir. Bu alanlar, metin kutuları (text boxes), açılır menüler (drop-down menus), radyo butonları (radio buttons) ve onay kutuları (checkboxes) gibi çeşitli türlerde olabilir.

Kontrollü ve Kontrolsüz Bileşenler

- **Kontrolsüz Bileşenler (Uncontrolled Components)**: Değer, tarayıcı tarafından yönetilir ve React, sadece sayfada değeri yerleştirir. Bu, daha az kod gerektirir ancak daha az kontrol sağlar.
- **Kontrollü Bileşenler (Controlled Components)**: React durumu (state) form verilerini yönetir. Bu, durumu oluşturmak ve güncellemek için açıkça kod yazmayı gerektirir.

Form Yönetimi ve Doğrulama

- Formlar, kullanıcı girişini kabul etmek için bir gönderim mantığı (submission logic) sağlamalıdır. Örneğin, kullanıcı bir "Submit" (Gönder) butonuna tıklamadan şifreyi kabul etmemelidir.
- Doğrulama (validation) işlemleri, kullanıcıların şifre oluştururken genellikle iki kez yazmalarını ve belirli kurallara uymalarını gerektirir.

React Hook Form Kütüphanesi

- React Hook Form, form yönetimi ve doğrulama için yardımcı olur. Bu, geliştiricilerin yazması gereken kod miktarını azaltır ve performansı artırır.
- Kütüphaneyi kurmak için "install react hook form" komutunu kullanabilirsiniz.