Component Lifecycle

React Native'de **component** (bileşen), kullanıcı arayüzünü oluşturmak için kullanılan temel yapı taşlarıdır. Bir component, genellikle ekranın belirli bir bölümünü temsil eder ve yeniden kullanılabilir yapıdadır.

React Native'de bir component (bileşen), sayfa üzerinde var olduğu sürece belirli aşamalardan geçer. Bu sürece "component lifecycle" denir. Başlıca aşamalar şunlardır:

- Mounting (Başlatma): Component oluşturulup ekrana ilk kez yerleştirilir. `constructor`, `render`, ve `componentDidMount` gibi metodlar bu aşamada çalışır.

- Updating (Güncelleme): Component'in durumu (`state`) veya aldığı veriler (`props`) değiştiğinde tetiklenir. `render`, `componentDidUpdate` gibi metodlar burada çalışır.

- Unmounting (Kaldırma): Component, ekrandan kaldırılmadan önce çalışır. `componentWillUnmount` bu aşamada tetiklenir.

`useState` Nedir ve Nasıl Kullanılır?

`useState`, React Native'de bir componentin içindeki durumları (state) yönetmek için kullanılan bir hook'tur. `useState`, bir değişkenin hem değerini hem de bu değeri güncelleyen bir fonksiyonu sağlar.

Kullanımı:

const [deger, setDeger] = useState(baslangicDegeri);

Burada:

- `deger`: Başlangıç değeri olarak atanan değeri saklar.

- `setDeger`: Bu değeri değiştirmek için kullanılan fonksiyondur.

Örneğin:

const [sayac, setSayac] = useState(0);

Bu kod, `sayac` değişkenini 0 ile başlatır ve `setSayac` ile sayacı güncelleyebilirsin.

`useEffect` Nedir ve Nasıl Kullanılır?

`useEffect`, React component'leri içindeki yan etkileri yönetmek için kullanılır. Örneğin, bir veri yükleme işlemi ya da bir event listener eklemek gibi işlemler bu kategoridedir.

Kullanımı:

useEffect(() => {

// Bu kod component mount olduktan sonra çalışır.

return () => {

// Bu kod component unmount olduğunda çalışır.

};

}, [bagimlilik]);

Burada:

- `bagimlilik`: Bu dizi içine koyduğun değerler değiştiğinde `useEffect` yeniden çalışır. Eğer boş bırakılırsa sadece component ilk kez render olduğunda çalışır.

Hooks Kullanmanın Avantajları

Hooks, fonksiyonel component'lerde durum (`state`) ve yan etkileri (`effect`) kolayca yönetmeyi sağlar.

Avantajlar:

- Yeniden Kullanılabilirlik: Hooks, küçük ve bağımsız kod parçaları olarak yazılabilir ve farklı component'lerde tekrar tekrar kullanılabilir.

- Daha Az Kod: Class component'lere göre daha az kodla aynı işlevselliği sağlayabilirsin.

- Fonksiyonel Componentler: Fonksiyonel componentler, class component'lere göre daha sade ve anlaşılırdır.

Kullanım Şartları:

- Hooks, sadece fonksiyonel component'lerde kullanılabilir. Class component'lerde kullanılamaz.

- Hooks, her zaman en üst seviyede çağrılmalıdır; yani bir loop, koşul ya da nested fonksiyon içinde çağrılmamalıdır.