React'te **Function Components** ile **Props** ve **Event Handling** konularını ele alıyor. Aşağıda önemli noktaları bulabilirsiniz:

Props Nedir?

- **Props** (özellikler), bir bileşenden diğerine veri geçişini sağlar. Bir bileşenin özelliklerini saklar ve bu özellikleri bir üst bileşenden alt bileşene geçirir.
- Props'lar **immutable** (değiştirilemez) olup, alt bileşende değiştirilemezler.

Props Kullanımı

- Örnek olarak, bir **App** bileşeni, **EmployeeData** adlı bir alt bileşene veri geçirir. **name** özelliği "John" olarak atanır.
- Props nesnesi, bileşene geçirilen tüm özellikleri içerir ve bu özelliklere **dot operator** (nokta operatörü) ile erişilir.

Varsayılan Props Değerleri

- Bileşenin **DefaultProps** nesnesinde varsayılan değerler tanımlanabilir. Eğer üst bileşen belirli bir prop değeri sağlamazsa, alt bileşen varsayılan değeri kullanır.

Props İlkeleri

- **Reusability** (yeniden kullanılabilirlik): Farklı props'lar geçirerek aynı bileşenin varyasyonlarını oluşturabilirsiniz.
- **Unidirectional Data Flow** (tek yönlü veri akışı): Verilerin nasıl geçtiğini ve güncellendiğini görmek kolaydır.

Event Handling ile Props Kullanımı

- **UseState** hook'u, bileşenin durumunu yönetir. Örneğin, bir butona tıklandığında bir durumu güncelleyerek UI'da belirli bilgileri gösterir.

component composition (bileşen bileşimi) kavramını açıklamaktadır ve bu kavramın nasıl kullanılacağını anlamanıza yardımcı olacaktır.

Component Composition

- **Purpose**: Bileşen bileşimi, birden fazla bileşeni bir araya getirerek karmaşık işlevsellik oluşturmanıza olanak tanır.
- **Function Components**: Küçük bileşenleri birleştirerek daha büyük bir kullanıcı arayüzü (UI) yapısı oluşturabilirsiniz.

Principles of Function Component Composition

- **Abstraction**: Yeniden kullanılabilir bileşenler oluşturmanıza olanak tanır ve büyük bir UI'yi daha küçük parçalara ayırır. Bu, kodunuzu düzenli ve sürdürülebilir hale getirir.
- **Hierarchy**: Bileşenleri hiyerarşik bir yapıda düzenlemenizi sağlar. Ebeveyn ve çocuk bileşenleri arasında veri geçişini kolaylaştırır.

Higher-Order Components (HOCs)

- **Props**: Ebeveyn bileşenden çocuk bileşene veri geçişi sağlar. Bileşenleri diğer bileşenlere geçirebilirsiniz.
- **HOCs**: Bileşenleri girdi olarak kabul eden veya geliştirilmiş işlevsellik ile dönen fonksiyonlardır. Bu, bileşenlerin özelliklerini eklemenizi sağlar.

Bu video, React'te **state management** (durum yönetimi) kavramını ve **useState hook** (kancası) kullanımını açıklamaktadır. İşte önemli noktalar:

State Management

- **State**: Bir bileşenin belirli bir zamandaki durumunu temsil eder ve zamanla değişebilen verileri yönetmek için kullanılır.
- **Local State**: Sadece bir bileşen içinde geçerlidir ve diğer bileşenler tarafından erişilemez.

useState Hook

- **useState**: Fonksiyon bileşenlerine durum yönetimi işlevselliği ekler. Bu kanca, bir durum değeri ve bu değeri güncellemek için bir fonksiyon döndürür.
- **Array Destructuring**: useState'den dönen dizi, iki değeri ayırmak için kullanılır: **stateName** (durum adı) ve **setStateName** (durum güncelleme fonksiyonu).

Örnek Kullanım

- Bir bileşen oluşturulurken, **useState** fonksiyonu ile başlangıç durumu belirlenir. Örneğin, bir isim durumu "John" olarak ayarlanabilir.
- **setStateName** fonksiyonu, durumu güncellemek için kullanılır. Örneğin, bir butona tıklandığında ismi "John Doe" olarak değiştirebiliriz.

Fonksiyonel Bileşenlerin Yaşam Döngüsü

- **Mounting Phase (Montaj Aşaması)**: React, fonksiyonel bileşeni başlatır ve DOM'da render etmeye hazırlar. Bu aşamada, bileşenin işlevselliği ve başlangıç durumu ayarlanır.
- **State Initialization (Durum Başlatma)**: useState hook'u kullanılarak bileşen içinde durum değişkenleri tanımlanır. Örneğin, const [count, setCount] = useState(0); ile count durumu 0 olarak başlatılır.

Güncelleme Aşaması (Updating Phase)

- React, bileşenin durumu veya props'larındaki değişikliklere yanıt verir ve bileşenin işlevini yeniden çağırır. Bu, kullanıcı etkileşimleriyle bileşenin güncellenmesini sağlar. Örneğin, bir butona tıklanması durumunda setCount(count + 1); ile count değeri artırılır.

Unmounting Phase (Kaldırma Aşaması)

- Bileşen DOM'dan kaldırıldığında, React temizlik işlemleri gerçekleştirir. Bu, olay dinleyicileri ve zamanlayıcılar gibi kaynakların serbest bırakılmasını içerir.

Örneğin, useEffect içinde bir temizlik fonksiyonu döndürülerek clearInterval ile zamanlayıcı temizlenir.

Hata Yönetimi Aşaması (Error Handling Phase)

- Eğer bir hata oluşursa, React bu hatayı en yakın hata sınırına yönlendirir. Hata sınırları, hataları yakalayan özel bileşenlerdir ve uygulamanın diğer kısımlarının çalışmaya devam etmesini sağlar.

Testin Tanımı ve Önemi

- **Testing**: Kodunuzun nasıl çalışacağını satır satır inceleme işlemidir.
- **Regression**: Daha önce düzeltilmiş bir hatanın tekrar ortaya çıkmasını önler.

Testin Avantajları ve Dezavantajları

- **Avantajlar**:
  - Beklenmedik regresyonları önler.
  - Geliştiricinin mevcut göreve odaklanmasını sağlar.
- **Dezavantajlar**:
  - Daha fazla kod yazmanız ve bakım yapmanız gerekir.
  - Kritik olmayan test hataları, sürekli entegrasyon (Continuous Integration) sürecinde uygulamanın reddedilmesine neden olabilir.

React Bileşen Test Yaklaşımları

- **Unit Testing**: Bileşen ağaçlarını basit bir test ortamında render etme ve çıktısını doğrulama.
- **End-to-End Testing**: Tam bir uygulamanın gerçekçi bir tarayıcı ortamında çalıştırılması.

Test Aşamaları

- **Arrange**: Bileşen özelliklerinin hazırlandığı aşama.
- **Act**: Bileşenin DOM'unu kullanıcı arayüzüne render etmesi ve kullanıcı eylemlerini kaydetmesi.
- **Assert**: Beklentilerin ayarlandığı ve bileşen markup'ı üzerinde belirli yan etkilerin doğrulandığı aşama.

Test Araçları

- **Jest**: Facebook tarafından React bileşenlerini test etmek için geliştirilmiş bir test koşucusu ve assertion kütüphanesidir.
- **React Testing Library**: Bileşenlerin uygulama detaylarına bağımlı olmadan test edilmesini sağlayan yardımcı fonksiyonlar sunar.

Dizilerin Tanımı ve Önemi

- **Array (Dizi)**: JavaScript'te birden fazla değeri tek bir değişkende saklamak için kullanılan bir veri yapısıdır. Dizi, köşeli parantezler içinde virgülle ayrılmış elemanlar ile tanımlanır.
- Diziler, sayılar, dizeler (strings), nesneler (objects) veya diğer diziler gibi herhangi bir veri türünü içerebilir.

Dizilerin Tanımlanması ve Kullanımı

- Diziler, React bileşenlerinde **useState (durum yönetimi)** kancası ile saklanabilir. Örneğin, const [items, setItems] = useState([]); şeklinde tanımlanabilir.
- **Dynamic Construction (Dinamik Yapı)**: Uygulama mantığına veya alınan verilere dayalı olarak diziler dinamik olarak oluşturulabilir.

Dizilerin Gezinilmesi

- Diziler üzerinde gezinmek için **map**, **forEach**, ve **for...of** döngüleri gibi yöntemler kullanılabilir. Örneğin, items.map(item => <li key={index}>{item}); ile her bir eleman için yeni bir JSX öğesi oluşturulabilir.
- **Key Attribute (Anahtar Özelliği)**: React'in her liste öğesini verimli bir şekilde tanımlamasına yardımcı olmak için kullanılır.

Dizilerle Etkileşim

- **Adding Items (Öğe Ekleme)**: Kullanıcıdan alınan verilerle dizilere yeni öğeler eklenebilir. Örneğin, bir butona tıklandığında setItems([...items, newItem]); ile yeni bir öğe eklenebilir.
- **Removing Items (Öğe Çıkarma)**: Belirli bir indeksteki öğeyi çıkarmak için splice yöntemi kullanılabilir. Örneğin, items.splice(index, 1); ile öğe silinebilir.

Koşullu Render (Conditional Rendering)

- Dizinin içeriğine bağlı olarak bileşenler koşullu olarak render edilebilir. Örneğin, dizi boşsa "No items available" mesajı gösterilebilir.

React'te Virtual DOM'un nasıl çalıştığını açıklamaktadır. Aşağıda önemli noktaları bulabilirsiniz:

Document Object Model (DOM)

- DOM, web sayfalarını ve belgeleri temsil eden bir arayüzdür ve HTML yapısını ağaç benzeri bir yapı olarak gösterir.
- DOM, içerik, yapı ve stil üzerinde dinamik olarak erişim ve manipülasyon sağlar.

Virtual DOM

- Virtual DOM, React gibi frameworklerde performansı optimize etmek için kullanılan bir kavramdır. Gerçek DOM'un bellekteki bir soyutlamasıdır.
- React, bileşenlerin durumu veya özellikleri değiştiğinde yeni bir Virtual DOM temsili oluşturur ve önceki ile karşılaştırarak en küçük değişiklikleri belirler.

Avantajları

- Virtual DOM, DOM değişikliklerinin sıklığını azaltarak web uygulamalarının hızını artırır.
- Geliştiriciler, DOM manipülasyonunun karmaşıklıklarıyla ilgilenmeden, deklaratif kod geliştirmeye odaklanabilirler.
- Sunucu tarafı render'ı (Server-Side Rendering) ile başlangıç sayfa yükleme performansını iyileştirir.

Normal DOM ile Virtual DOM Arasındaki Farklar

- Normal DOM, tarayıcıda HTML içeriği analiz edilerek oluşturulurken, Virtual DOM bellek içinde yer alır ve doğrudan tarayıcının render motoruna bağlı değildir.
- React, güncellemeleri en az geçiş ve güncelleme ile gerçek DOM'a uygular, bu da performansı artırır.

DOM'un (Document Object Model) bileşenleri, web sayfalarının yapısını ve içeriğini temsil eden temel unsurlardır. İşte DOM'un ana bileşenleri:

1. **Düğümler (Nodes)**: DOM, bir ağaç yapısı şeklinde düzenlenmiş düğümlerden oluşur. Her düğüm, belgenin bir parçasını temsil eder. Düğümler, element düğümleri, metin düğümleri ve yorum düğümleri gibi farklı türlerde olabilir.
2. **Elementler (Elements)**: HTML belgesinin yapı taşlarıdır. Her bir HTML etiketi (örneğin, , <p>, <a>) bir element düğümünü temsil eder. Elementler, sayfanın içeriğini ve yapısını belirler.
3. **Öznitelikler (Attributes)**: Elementlere ek bilgi sağlayan özelliklerdir. Örneğin, bir <img> etiketinin src ve alt gibi öznitelikleri vardır. Bu öznitelikler, elementlerin davranışını ve görünümünü etkiler.
4. **Metin (Text)**: Elementlerin içinde bulunan yazılı içeriktir. Metin düğümleri, kullanıcıya gösterilen metni temsil eder.
5. **Olaylar (Events)**: Kullanıcı etkileşimleri (örneğin, tıklama, fare hareketi) gibi olayları temsil eder. JavaScript, bu olaylara yanıt vererek sayfanın dinamik olarak güncellenmesini sağlar.

Bu bileşenler, web sayfalarının dinamik olarak oluşturulmasını ve manipüle edilmesini sağlar. DOM, geliştiricilerin sayfanın içeriğini, yapısını ve stilini programatik olarak değiştirmesine olanak tanır.