React hakkında temel bilgiler:

Kütüphaneler ve Framework'ler

- **Library (Kütüphane)**: Önceden yazılmış kod parçalarını içeren bir depo. Örnekler: jQuery, Lodash, D3.js, React.
- **Framework (Framework)**: Uygulama geliştirmek için yapı ve araçlar sağlayan kapsamlı bir platform. Örnekler: AngularJS, Ember JS, Svelte, Vue.js.

Ön Uç Framework'leri

- **Front-end Framework (Ön Uç Framework)**: Web uygulamalarının kullanıcı arayüzünü (UI) oluşturmak için kullanılan framework'ler. Örnekler: React, AngularJS, Vue.js.

React'ın Öne Çıkan Özellikleri

- **Component-based Architecture (Bileşen Tabanlı Mimari)**: Tekrar kullanılabilir UI bileşenleri oluşturma imkanı sunar.
- **Declarative Syntax (Açıklayıcı Söz Dizimi)**: UI davranışını tanımlamaya odaklanır, React arka planda DOM'u yönetir.
- **Virtual DOM (Sanal DOM)**: Değişiklikler olduğunda gerçek DOM ile karşılaştırma yaparak yalnızca gerekli kısımları günceller.
- **One-way Data Binding (Tek Yönlü Veri Bağlama)**: Veri akışının yalnızca üstten alta doğru olmasını sağlar.
- **JSX (JavaScript XML)**: HTML benzeri kod yazmayı kolaylaştıran bir JavaScript sözdizimi uzantısıdır.
- **Hooks**: React 16.8 ile tanıtılan, durum ve diğer React özelliklerini yönetmeyi kolaylaştıran bir özellik.

Vite Build Tool

- **Vite**: Modern web geliştirme için geliştirilmiş bir build tool'dur. React, Angular veya düz JavaScript ile kullanılabilir.
- **Speed (Hız)**: Vite, JavaScript modüllerini yalnızca gerektiğinde birleştirerek geliştirme sürecini hızlandırır.

React Projesi Oluşturma Adımları 1. **npm create command**: Terminalde bu komutu yazarak yeni bir proje oluşturabilirsiniz. 2. **Project Name (Proje Adı)**: Projeniz için bir isim verin. 3. **Framework Selection (Framework Seçimi)**: Açılan listeden React'ı seçin. 4. **Scripting Language (Betik Dili)**: JavaScript olarak seçin. 5. **Project Structure (Proje Yapısı)**: Proje oluşturulduktan sonra, gerekli dosyalar ve dizinler otomatik olarak oluşturulur.

React Klasör Yapısı

- **node_modules**: Tüm bağımlılıkların bulunduğu dizin.
- **public**: Statik varlıkların (HTML, resimler, fontlar) bulunduğu dizin.

- **src**: Uygulamanın kaynak kodunun bulunduğu dizin. İçinde:
    - **main.jsx**: Uygulamanın giriş noktası.
    - **App.jsx**: Uygulamanın ana bileşeni.
- **package.json**: Proje hakkında meta veriler ve bağımlılıkları içerir.
- **vite.config.js**: Vite yapılandırma ayarlarını içerir.

ECMAScript 6 (ES6) Nedir?

- ES6, JavaScript'in 2015 yılında çıkan bir sürümüdür ve önemli değişiklikler içermektedir.
- ES.next, gelecekteki ECMAScript sürümlerini ifade eden dinamik bir isimdir.

Yeni Özellikler

- **let** ve **const**: Değişken tanımlamak için kullanılır. let, değişkenin kapsamını (scope) sınırlarken, const sabit değerler tanımlar.
- **Arrow Functions**: Fonksiyonları daha kısa ve temiz bir şekilde tanımlamak için kullanılır. Örneğin, () => {} şeklinde yazılabilir.
- **Promises**: Asenkron işlemlerin tamamlanmasını temsil eder. Bir promise, "pending" (beklemede), "fulfilled" (tamamlanmış) veya "rejected" (reddedilmiş) durumlarında olabilir.
- **Class**: JavaScript'te nesne yönelimli programlamayı (object-oriented programming) mümkün kılar. Sınıflar, nesne oluşturmak için bir şablon (template) görevi görür.

Örnek Kullanımlar

- let ve const ile değişken tanımlarken, let yerel kapsamda (local scope) geçerlidir.
- Arrow functions, normal fonksiyonlar gibi çağrılabilir ve parametre alabilir.
- Class yapısı, nesne oluşturmak için new anahtar kelimesi ile kullanılır ve miras alma (inheritance) özelliği vardır.

JSX'in Temel Özellikleri

- JSX, HTML benzeri bir sözdizimi kullanır ve açılış ve kapanış etiketleri ile öğeleri tanımlar.
- JSX, JavaScript ifadelerini destekler ve bu sayede dinamik içerik oluşturulabilir.

JSX'in Derlenmesi

- Tarayıcılar JSX'i anlamaz, bu nedenle JSX kodunun standart JavaScript nesnelerine dönüştürülmesi için Babel gibi bir derleyici (compiler) kullanmak gerekir.
- create react app komutu, JSX'in derlenmesini otomatikleştirir.

JSX'in Avantajları

- JSX, HTML ile daha tanıdık bir yapı sunarak, JavaScript bilmeyenlerin de kodu anlamasını kolaylaştırır.
- Hatalar, derleme sırasında daha erken tespit edilir, bu da hata ayıklamayı (debugging) kolaylaştırır.
- JSX, kodu daha basit ve şık tutar, ayrıca performansı artırır.
- Güvenlik açısından, JSX otomatik olarak çıktı sanitizasyonu (output sanitation) yaparak, kötü niyetli kodların çalıştırılmasını engeller.

React bileşenlerinin (components) temellerini anlamanıza yardımcı olmayı amaçlamaktadır.

Bileşenlerin Tanımı

- React uygulamaları, modüler kod parçaları olan bileşenlerle oluşturulur. Bileşenler, kullanıcı arayüzünü (user interface) ayrı parçalara ayırarak daha karmaşık arayüzlerin inşasını kolaylaştırır.
- Bir React bileşeni, isteğe bağlı girdi (input) alır ve ekranda render edilen (render) bir React nesnesi döndürür.

Bileşenlerin Özellikleri

- Bileşenler, üç ana unsurdan (aspects) oluşur: özellikler (properties), olaylar (events) ve durumlar (states).
- Özellikler, bir üst bileşenden (parent component) bir alt bileşene (child component) veri (data) geçişini sağlar. Olaylar, kullanıcının etkileşimleri sonucunda belge nesne modeli (DOM) üzerinde değişiklikler yapar.

React Bileşen Türleri

- Üç tür React bileşeni vardır: sınıf bileşenleri (class components), fonksiyonel bileşenler (functional components) ve yüksek düzey bileşenler (higher-order components).
- Fonksiyonel bileşenler, JavaScript fonksiyonları gibi yazılır ve JSX (JavaScript XML) döndürebilir. React 16.8'den itibaren, fonksiyonel bileşenler durum (state) yönetimi yapabilir hale gelmiştir.

Sınıf Bileşenleri

- **Class Component**: React'te bir sınıf bileşeni, React kütüphanesinden **React.Component**'i genişleten bir JavaScript sınıfıdır.
- **Render Method**: Her sınıf bileşeninde bulunan ve bileşenin kullanıcı arayüzünü temsil eden JSX öğelerini döndüren bir yöntemdir.

Durum Yönetimi

- **State**: Bileşenin render edilmesi ve kullanıcı etkileşimlerine yanıt vermesi için gereken veriyi temsil eder. **this.state** ile başlatılır ve **this.setState** ile güncellenir.

- **Event Handling**: Kullanıcı etkileşimlerine yanıt vermek için olayları yönetme sürecidir. Örneğin, bir butona tıklandığında bir uyarı göstermek için **onClick** özelliği kullanılır.

Props Kullanımı

- **Props**: Üst bileşenden alt bileşene veri göndermek için kullanılır. Alt bileşen, üst bileşenden aldığı verileri yalnızca okumak için kullanır.
- **Destructuring**: Alt bileşende, üst bileşenden gelen props'ları almak için **this.props**'dan parçalama işlemi yapılır.

State (Durum)

- **State**, bir bileşenin mevcut durumunu temsil eden düz bir JavaScript nesnesidir. Bileşenin davranışını ve nasıl render edileceğini belirler.
- **Local State** (Yerel Durum) yalnızca bir bileşende bulunur ve diğer bileşenler tarafından erişilemez. Örneğin, bir bileşenin bilgi gösterip gizlemesi için yerel durum kullanılır.

Props (Özellikler)

- **Props**, bileşenler arasında veri geçişi sağlamak için kullanılır. Veri, yalnızca üst bileşenden alt bileşene doğru akar.
- **Props** değiştirilemez (immutable) ve bileşenin içinde modifiye edilemez. Eğer bir değişkenin değiştirilmesi gerekiyorsa, bu bileşenin **state**'ine ait olmalıdır.

State ve Props Karşılaştırması

- **State**, bileşenin kendi verilerini yönetirken, **props** dışarıdan veri alır.
- **State**'i bileşenin içinde değiştirebilirsiniz, ancak dışarıdan erişemezsiniz. **Props** ise yalnızca üst bileşenden alt bileşene geçer ve değiştirilemez.

Bileşen Yaşam Döngüsü

- React bileşenleri, yaşamları boyunca üç aşamadan geçer: **mounting** (montaj), **updating** (güncelleme) ve **unmounting** (sökme).
- Her aşamada, React'in erişebileceğiniz yaşam döngüsü yöntemleri vardır.

Montaj Aşaması (Mounting Phase)

- Bu aşamada, bileşen sınıfı **constructor** (yapıcı) kullanılarak oluşturulur ve varsayılan bir **state** (durum) atanır.
- İlk kez **render** (görüntüleme) yöntemi çağrılır. **componentWillMount** ve **componentDidMount** yöntemleri bu aşamada kullanılır.

Güncelleme Aşaması (Updating Phase)

- Kullanıcı olayları veya arka uçtaki değişiklikler nedeniyle bileşenin **state** veya **props** (özellikler) değişir.

- Bu
  aşamada, **getDerivedStateFromProps**, **getSnapshotBeforeUpdate** ve **componentDidUpdate** yöntemleri kullanılır.
- **getDerivedStateFromProps**, güncellenen özellikleri bileşen durumuna yansıtmak için kullanılır.

Sökme Aşaması (Unmounting Phase)

- Bu aşamada, bileşen sayfadan kaldırılır. **componentWillUnmount** yöntemi, bileşen kaldırılmadan önce çağrılır ve gerekli temizleme işlemlerini yapar.