Temel Konular

1. **React'e Giriş**:
    - **React Nedir?**: Kullanıcı arayüzleri oluşturmak için kullanılan açık kaynaklı bir JavaScript kütüphanesidir. Facebook tarafından geliştirilmiştir.
    - **React'in Temel Yapısı**: React, bileşen tabanlı bir yapıya sahiptir. Her bileşen, bir kullanıcı arayüzünün parçalarını temsil eder ve bu bileşenler, JSX (JavaScript XML) kullanılarak tanımlanır.
2. **React CLI (Komut Satırı Arayüzü)**:
    - **Create React App**: React projelerini hızlıca başlatmak için kullanılan bir araçtır. Projeyi başlatmak için npx create-react-app my-app komutu kullanılır.
    - **Diğer Araçlar**: Vite veya Next.js gibi diğer CLI araçları da kullanılabilir.
3. **Bileşenler**:
    - **Fonksiyon Bileşenleri**: Bir fonksiyon olarak tanımlanır ve JSX döner.
    - **Sınıf Bileşenleri**: ES6 sınıflarını kullanarak tanımlanır, ancak genellikle fonksiyon bileşenlerinin yerini almıştır.

4. **JSX**:
    - **JSX Nedir?**: JavaScript'te HTML benzeri bir sözdizimi sağlar. HTML etiketlerine benzer şekilde yazılır, ancak JavaScript kodlarıyla birleşir.
5. **Durum Yönetimi**:
    - **useState Hook**: Bileşenlerin yerel durumunu yönetmek için kullanılır.
    - **useReducer Hook**: Daha karmaşık durum yönetimi için kullanılır. Özellikle büyük uygulamalarda tercih edilir.
6. **Props (Özellikler)**:
    - **Props Nedir?**: Bileşenlere veri ve işlevler geçirmek için kullanılır. Bileşenler arası veri akışını sağlar.
7. **Olay Yönetimi**:
    - **Olaylar**: Kullanıcı etkileşimlerini yönetmek için onClick, onChange gibi olay işleyicileri kullanılır.

Ara Konular

1. **Formlar ve Doğrulama**:
    - **Kontrol Edilen ve Kontrol Edilmeyen Bileşenler**: Kontrol edilen bileşenler (controlled components), form verilerini React durumuyla

yönetir. Kontrol edilmeyen bileşenler (uncontrolled components) ise DOM üzerinde doğrudan işlem yapar.

- **Doğrulama**: Form verilerini doğrulamak için çeşitli yöntemler ve kütüphaneler (örneğin, Formik, Yup) kullanılır.

2. **Yönlendirme**:
   - **React Router**: Tek sayfalık uygulamalarda sayfalar arası geçişleri yönetmek için kullanılır. BrowserRouter, Route, ve Link gibi bileşenleri içerir.
3. **Context API**:
   - **Context API Nedir?**: Bileşenler arasında veri paylaşmak için kullanılan bir mekanizmadır. Props drilling (veri akışı) sorununu çözer ve global durumu yönetir.
4. **HTTP İstekleri**:
   - **Axios ve Fetch API**: Sunucu ile veri alışverişi yapmak için kullanılan kütüphanelerdir. fetch yerleşik bir API iken, axios ise daha fazla özellik sunar.
5. **Özel Hook'lar**:
   - **Özel Hook'lar**: Tekrarlanan mantığı bir araya getirmek için kullanılır. Örneğin, useForm gibi.
6. **Hata Sınırları**:

- **Hata Sınırları**: JavaScript hatalarını yakalamak ve kullanıcıya dostça bir hata mesajı göstermek için kullanılır. componentDidCatch metodu ile uygulanır.

İleri Düzey Konular

1. **Gelişmiş Durum Yönetimi**:
   - **Redux ve MobX**: Daha karmaşık durum yönetimi için kullanılan kütüphanelerdir. Redux, tek yönlü veri akışı sağlar, MobX ise daha reaktif bir yaklaşımdır.
2. **Performans Optimizasyonu**:
   - **Memoization**: React.memo ve useMemo gibi araçlarla bileşenlerin gereksiz yeniden render edilmesinin önüne geçilir.
   - **Kod Bölme**: React.lazy ve Suspense ile uygulamanın yüklenme süresi optimize edilir.
3. **Sunucu Tarafı Render (SSR)**:
   - **Next.js**: Sunucu tarafında render yaparak, sayfa yükleme sürelerini iyileştirir ve SEO dostu uygulamalar oluşturur.
4. **Statik Site Üretimi (SSG)**:
   - **Gatsby**: React kullanarak statik siteler oluşturmak için kullanılır. İçerik önceden oluşturulur ve sayfalar hızlıca sunulur.

5. **Test Etme**:
    - **Jest ve React Testing Library**: React bileşenlerini test etmek için kullanılır. Jest, bir test koşum aracıdır, React Testing Library ise kullanıcı etkileşimlerini simüle eder.
6. **Animasyonlar**:
    - **React Spring ve Framer Motion**: React uygulamalarında dinamik animasyonlar ve geçişler oluşturmak için kullanılır.
7. **TypeScript ile React**:
    - **TypeScript**: React uygulamalarında tip güvenliğini sağlamak için kullanılan bir dil uzantısıdır. TypeScript, daha sağlam ve hata öncesi kod yazılmasına yardımcı olur.
8. **Progresif Web Uygulamaları (PWA)**:
    - **PWA**: Mobil ve masaüstü cihazlarda offline çalışma yeteneği sunan web uygulamalarıdır. React ile PWA'lar oluşturulabilir.
9. **Erişilebilirlik**:
    - **Erişilebilirlik (a11y)**: Uygulamanın herkes tarafından kullanılabilir olmasını sağlamak için erişilebilirlik en iyi uygulamalarını uygulamak önemlidir.

Ek Konular

1. **Üçüncü Taraf Kütüphaneler ve Bileşenler**:
    - **Material-UI, Ant Design, Bootstrap**: Kullanıcı arayüzlerini geliştirmek için kullanılan popüler kütüphanelerdir.
2. **React CLI ile Proje Oluşturma ve Yapılandırma**:
    - **Projeleri Yönetme**: create-react-app gibi araçlarla projeleri hızlıca başlatmak ve üretim için yapılandırmak.
3. **Sunucu İletişimi**:
    - **GraphQL ve REST API'leri**: Backend servisleri ile iletişim kurma yöntemleridir. GraphQL, esnek veri sorgulama ve alma sağlar.
4. **Kod Bölme**:
    - **Kod Bölme (Code Splitting)**: Uygulamanın performansını iyileştirmek için sadece gerekli kodların yüklenmesini sağlar.
5. **Statik Veri Üretimi**:
    - **Statik Veri**: React uygulamalarında önceden oluşturulmuş veriler kullanılarak performans iyileştirmeleri yapılabilir.