React nedir?

React, Facebook tarafından geliştirilen ve ilk olarak 2013 yılında tanıtılan bir JavaScript kütüphanesidir. Tek sayfa uygulamaları (SPA) ve kullanıcı arayüzleri (UI) oluşturmak için kullanılır. React, özellikle kullanıcı arayüzlerinin hızlı, interaktif ve dinamik bir şekilde geliştirilmesini sağlamak amacıyla tasarlanmıştır.

**React'in Temel Özellikleri:**

1. **Bileşen Tabanlı Yapı:**
   * React, uygulamaları küçük, bağımsız ve yeniden kullanılabilir bileşenlere (components) bölmenize olanak tanır. Her bileşen, kendi durumunu (state) ve mantığını yönetebilir. Bu, uygulamanın daha düzenli ve yönetilebilir olmasını sağlar.
2. **Virtual DOM (Sanal DOM):**
   * React, gerçek DOM yerine sanal bir DOM kullanır. Virtual DOM, bir hafif JavaScript nesnesidir ve gerçek DOM’un hafif bir kopyasıdır. Bir bileşenin durumu değiştiğinde, React önce Virtual DOM’da bu değişiklikleri uygular ve ardından yalnızca gerekli olan kısımları gerçek DOM’a günceller. Bu, performansı artırır ve uygulamanın daha hızlı çalışmasını sağlar.
3. **Tek Yönlü Veri Akışı (One-Way Data Binding):**
   * React, tek yönlü veri akışı sağlar. Bu, veri akışının bileşenler arasında yukarıdan aşağıya doğru (parent to child) gerçekleştiği anlamına gelir. Bu özellik, uygulamanın nasıl çalıştığını anlamayı ve hata ayıklamayı kolaylaştırır.
4. **JSX:**
   * JSX, JavaScript XML anlamına gelir ve JavaScript içinde HTML benzeri bir sözdizimi kullanmanıza olanak tanır. JSX, bileşenlerin nasıl görüneceğini tanımlarken daha okunabilir ve anlaşılır bir yapı sunar.
5. **Hooks:**
   * React, fonksiyon bileşenlerinde durum (state) ve yan etkileri (side effects) yönetmek için Hooks adlı bir özellik sunar. useState, useEffect gibi Hooks, fonksiyon bileşenlerinde state ve lifecycle özelliklerini kullanmanıza olanak tanır.
6. **Ekosistem ve Topluluk Desteği:**
   * React, geniş bir ekosistem ve büyük bir topluluk tarafından desteklenir. Bu, çok sayıda hazır bileşen, araç ve kütüphane anlamına gelir, bu da geliştirme sürecini hızlandırır.

**React'in Kullanım Alanları:**

* **Tek Sayfa Uygulamaları (SPA):**
  + React, modern web uygulamalarında yaygın olarak kullanılır, özellikle kullanıcıların sayfa yeniden yüklenmeden farklı sayfalara geçiş yapabildiği tek sayfa uygulamaları geliştirmek için idealdir.
* **Mobil Uygulamalar:**
  + React Native, React'in mobil uygulama geliştirmeye uyarlanmış bir versiyonudur ve bu sayede hem iOS hem de Android için yerel (native) mobil uygulamalar geliştirmek mümkündür.
* **Kurumsal Uygulamalar:**
  + Büyük ölçekli, karmaşık ve etkileşimli kullanıcı arayüzleri geliştiren firmalar, React'i tercih eder. Netflix, Airbnb, Facebook gibi büyük şirketler React kullanmaktadır.

React, modern web geliştirme süreçlerinde yaygın olarak tercih edilen bir kütüphane olup, özellikle interaktif ve performans odaklı kullanıcı arayüzleri geliştirmek isteyenler için güçlü bir araçtır.

**React** yüklemek için:

1. **Node.js ve npm/Yarn Yükleyin:** [Node.js'i](https://nodejs.org/) yükleyin, npm ile birlikte gelir. Yarn kullanıyorsanız, Yarn'i yükleyin.
2. **Yeni Proje Oluşturma:**
   * **Create React App ile:** npx create-react-app my-app veya yarn create react-app my-app
   * **Proje Dizini:** cd my-app
   * **Başlatma:** npm start veya yarn start
3. **Mevcut Projeye React Ekleme:**
   * **React ve ReactDOM Yükleme:** npm install react react-dom veya yarn add react react-dom

**React public klasörü kısaca şunları içerir:**

* **index.html:** Uygulamanın HTML şablonu, React bileşenlerinin render edileceği kök div içerir.
* **favicon.ico:** Tarayıcı sekmesinde görünen simge.
* **manifest.json:** PWA ayarları ve uygulamanın görünümü için yapılandırma dosyası.
* **robots.txt:** Arama motorlarına hangi sayfaların indekslenmesi gerektiğini belirten dosya.
* **Statik Dosyalar:** Resimler, fontlar gibi doğrudan erişilebilen dosyalar.

src klasörü

components klasörü ,

componentin içinde diyelim NotFound componenti

NotFound.js   
ve stil dosyası notFound.module.css

**constants** klasörü, uygulamanın çeşitli yerlerinde tekrar eden veya sabit kalması gereken değerleri ve yapılandırmaları merkezi bir yerden yönetmenizi sağlar. Bu, kodun daha düzenli ve bakımı kolay olmasını sağlar.

**Hooks klasörü** genellikle iş mantığını yönetmek için özel React Hooks’larını içerir. Bu Hooks'lar, fonksiyonel bileşenlerde kullanılacak işlevleri merkezi bir yerde toplar. İşte **hooks** klasöründe neler bulunabilir ve ne iş yapar:

**Hooks Klasöründe Bulunanlar**

1. **Özel Hook'lar**
   * **Açıklama:** Özel işlevler ve iş mantığını yöneten React Hook'larıdır. Bu Hook'lar, bileşenlerde tekrarlanabilen işlevleri kapsüller.
   * **Örnek:** useAuth, useFetch, useForm
2. **Genel Amaçlı Hooks**
   * **Açıklama:** Yaygın olarak kullanılan ve birden fazla bileşende işe yarayan genel işlevsellik sağlayan Hooks'tur.
   * **Örnek:** useLocalStorage, useDebounce, useWindowSize

**Hooks Klasöründe Neler Olur?**

1. **useAuth.js**
   * **Açıklama:** Kimlik doğrulama işlemleri için kullanılan Hook. Kullanıcının oturum açma, kapama ve kullanıcı bilgilerini yönetir.
   * **İçerik:** Kullanıcı bilgilerini alır, oturum açma ve kapama işlevlerini sağlar.
2. **useFetch.js**
   * **Açıklama:** API çağrıları yapmak için kullanılan Hook. Veriyi alır ve yüklenme, hata durumlarını yönetir.
   * **İçerik:** API'den veri çekme işlevini kapsar, veriyi, yüklenme durumunu ve hataları yönetir.
3. **useForm.js**
   * **Açıklama:** Form yönetimini kolaylaştıran Hook. Form verilerini yönetir, doğrulama yapar.
   * **İçerik:** Form verilerini saklar, formu sıfırlar, doğrulama yapar.
4. **useLocalStorage.js**
   * **Açıklama:** Tarayıcıda local storage kullanımı için Hook. Verileri local storage'da saklar ve okur.
   * **İçerik:** Local storage işlemlerini basitleştirir.
5. **useDebounce.js**
   * **Açıklama:** Debouncing işlevi sağlar, yani bir fonksiyonun sürekli tetiklenmesini kontrol eder.
   * **İçerik:** Belirli bir süre boyunca fonksiyon çağrılarını birleştirir.
6. **useWindowSize.js**
   * **Açıklama:** Pencere boyutunu izleyen Hook. Ekran boyutunu ve değişikliklerini takip eder.
   * **İçerik:** Ekran boyutunu izler ve değişiklikleri yönetir.

**Örnek useAuth Hook’u:**

js

Kodu kopyala

// hooks/useAuth.js

import { useState, useEffect } from 'react';

const useAuth = () => {

const [user, setUser] = useState(null);

const [loading, setLoading] = useState(true);

useEffect(() => {

// Kullanıcı bilgilerini al

const fetchUser = async () => {

const response = await fetch('/api/user');

const data = await response.json();

setUser(data.user);

setLoading(false);

};

fetchUser();

}, []);

const signIn = async (email, password) => {

const response = await fetch('/api/signin', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ email, password }),

});

const data = await response.json();

setUser(data.user);

};

const signOut = async () => {

await fetch('/api/signout', { method: 'POST' });

setUser(null);

};

return { user, loading, signIn, signOut };

};

export default useAuth;

**Kısacası:**

**Hooks klasörü** iş mantığını yöneten, çeşitli işlevsellik sağlayan ve birden fazla bileşende kullanılabilen React Hook'larını içerir. Her Hook, belirli bir işlevi kapsüller ve bileşenlerde bu işlevleri kullanarak kodun daha düzenli ve yeniden kullanılabilir olmasını sağlar.  
  
  
js dosyaları

React’te js dosyaları kullanılır

**Interceptors**, HTTP istekleri ve yanıtları üzerinde önceden tanımlı işlemler gerçekleştirmek için kullanılır. Bu, istekleri ve yanıtları özelleştirmek, yönetmek ve hata durumlarını ele almak için faydalıdır. İstemci kütüphanelerinde, interceptors genellikle isteklerin ve yanıtların merkezi bir şekilde yönetilmesini sağlar.

Pages klasörü vardir .

Aynı component gibi içeriği

 **Interceptors:** HTTP isteklerinin ve yanıtlarının öncesinde veya sonrasında müdahale eder, genel olarak ağ iletişimini ve hata yönetimini merkezileştirir.

 **Services:** İş mantığını ve veri etkileşimlerini yönetir, API çağrıları ve işlevsellikleri modüler bir şekilde sunar.

**App.js**:

import AppRoutes from './AppRoutes';

import Header from './components/Header/Header';

import Loading from './components/Loading/Loading';

import { useLoading } from './hooks/useLoading';

import { setLoadingInterceptor } from './interceptors/loadingInterceptor';

import { useEffect } from 'react';

function App() {

const { showLoading, hideLoading } = useLoading();

useEffect(() => {

setLoadingInterceptor({ showLoading, hideLoading });

}, []);

return (

<>

<Loading />

<Header />

<AppRoutes />

</>

);

}

export default App;

AppRoutes :

import React from 'react';

import { Route, Routes } from 'react-router-dom';

import CartPage from './pages/Cart/CartPage';

import FoodPage from './pages/Food/FoodPage';

import HomePage from './pages/Home/HomePage';

import LoginPage from './pages/Login/LoginPage';

import RegisterPage from './pages/Register/RegisterPage';

import AuthRoute from './components/AuthRoute/AuthRoute';

import CheckoutPage from './pages/Checkout/CheckoutPage';

import PaymentPage from './pages/Payment/PaymentPage';

import OrderTrackPage from './pages/OrderTrack/OrderTrackPage';

import ProfilePage from './pages/Profile/ProfilePage';

import OrdersPage from './pages/Orders/OrdersPage';

import Dashboard from './pages/Dashboard/Dashboard';

import AdminRoute from './components/AdminRoute/AdminRoute';

import FoodsAdminPage from './pages/FoodsAdmin/FoodsAdminPage';

import FoodEditPage from './pages/FoodEdit/FoodEditPage';

import UsersPage from './pages/UsersPage/UsersPage';

import UserEditPage from './pages/UserEdit/UserEditPage';

export default function AppRoutes() {

return (

<Routes>

<Route path="/" element={<HomePage />} />

<Route path="/search/:searchTerm" element={<HomePage />} />

<Route path="/tag/:tag" element={<HomePage />} />

<Route path="/food/:id" element={<FoodPage />} />

<Route path="/cart" element={<CartPage />} />

<Route path="/login" element={<LoginPage />} />

<Route path="/register" element={<RegisterPage />} />

<Route

path="/checkout"

element={

<AuthRoute>

<CheckoutPage />

</AuthRoute>

}

/>

<Route

path="/payment"

element={

<AuthRoute>

<PaymentPage />

</AuthRoute>

}

/>

<Route

path="/track/:orderId"

element={

<AuthRoute>

<OrderTrackPage />

</AuthRoute>

}

/>

<Route

path="/profile"

element={

<AuthRoute>

<ProfilePage />

</AuthRoute>

}

/>

<Route

path="/orders/:filter?"

element={

<AuthRoute>

<OrdersPage />

</AuthRoute>

}

/>

<Route

path="/dashboard"

element={

<AuthRoute>

<Dashboard />

</AuthRoute>

}

/>

<Route

path="/admin/foods/:searchTerm?"

element={

<AdminRoute>

<FoodsAdminPage />

</AdminRoute>

}

/>

<Route

path="/admin/addFood"

element={

<AdminRoute>

<FoodEditPage />

</AdminRoute>

}

/>

<Route

path="/admin/editFood/:foodId"

element={

<AdminRoute>

<FoodEditPage />

</AdminRoute>

}

/>

<Route

path="/admin/users/:searchTerm?"

element={

<AdminRoute>

<UsersPage />

</AdminRoute>

}

/>

<Route

path="/admin/editUser/:userId"

element={

<AdminRoute>

<UserEditPage />

</AdminRoute>

}

/>

</Routes>

);

}

axiosConfig

import axios from 'axios';

axios.defaults.baseURL =

process.env.NODE\_ENV !== 'production' ? 'http://localhost:5000' : '/';

index.css

index.js

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

import { BrowserRouter } from 'react-router-dom';

import CartProvider from './hooks/useCart';

import './axiosConfig';

import { AuthProvider } from './hooks/useAuth';

import { ToastContainer } from 'react-toastify';

import 'react-toastify/dist/ReactToastify.css';

import { LoadingProvider } from './hooks/useLoading';

import './interceptors/authInterceptor';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<BrowserRouter>

<LoadingProvider>

<AuthProvider>

<CartProvider>

<App />

<ToastContainer

position="bottom-right"

autoClose={5000}

hideProgressBar={false}

newestOnTop={false}

closeOnClick

rtl={false}

pauseOnFocusLoss

draggable

pauseOnHover

theme="light"

/>

</CartProvider>

</AuthProvider>

</LoadingProvider>

</BrowserRouter>

</React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

reportWebVitals

const reportWebVitals = onPerfEntry => {

if (onPerfEntry && onPerfEntry instanceof Function) {

import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {

getCLS(onPerfEntry);

getFID(onPerfEntry);

getFCP(onPerfEntry);

getLCP(onPerfEntry);

getTTFB(onPerfEntry);

});

}

};

export default reportWebVitals;

jsconfig.json, JavaScript projelerinde kodun yapılandırmasını ve import yollarını yönetmek için kullanılan bir dosyadır. Bu dosya, editörlerin ve IDE'lerin modül yollarını doğru şekilde çözmesini sağlar ve proje kök dizinini belirtir. Genellikle uzun import ifadelerini kısaltmak ve daha temiz kod yazmak için kullanılır.

**package-lock.json** dosyası, Node.js projelerinde kullanılan bir dosyadır ve proje bağımlılıklarının kesin sürümlerini kilitler. Bu dosya, **npm** (Node Package Manager) tarafından otomatik olarak oluşturulur ve proje ile ilgili tüm bağımlılıkların ve alt bağımlılıkların sürüm bilgilerini içerir.

**package.json** dosyası, Node.js projelerinde kullanılan ve proje hakkında bilgi sağlayan bir yapılandırma dosyasıdır.