React Projekt – Web API

Noter

React Terms

useState, useEffect, useRef, useMemo, useCallback, useContext, Suspense

1. **useState** gør det muligt at huske data mellem rendering af UI, når værdien ændres opdateres UI automatisk.
2. **useEffect** gør det muligt at håndtere sideeffekter så som API kald, det køre efter renderingen, og hvis man sender et tomt [] med så køre den kun én gang, ellers køre den når den variablen man sender med ind i [] ændres.
3. **useRef** gør lidt af det samme som useState, dog uden at trigger én re-render af UI, så den kan også holde på data mellem renderinger, bliver i mit tilfælde brugt til at huske på hvilken page vi er på når vi bruger pagination.
4. **useMemo** Husker tunge beregninger, så man kun laver beregningen én gang og bruger resultatet igen ved reload.
5. **useCallback** gør det samme som useMemo her er det bare funktioner den husker og ikke beregninger.
6. **useContext** gør det muligt at dele data mellem komponenter uden at sende det ned igennem hele komponenttræet, nu kan det bruges af alle komponenter.
7. **Suspense** bruges til at vise tekst eller andet imens siden loader.

React-router-dom Terms

BrowserRouter, NavLink, useLocation, useSearchParams, Outlet, useNavigate, Routes, Route

1. **BrowserRouter** er den øverste router komponent som holder styr på browserens historik og URL'er, hele appen skal pakkes ind i den her for at routing fungere.
2. **Routes** er en container som sørger for at kun den route der matcher den aktuelle URL bliver renderet.
3. **Route** definere den enkelte sti i applikationen og hvilket komponent som skal vises.
4. **NavLink** virker som et almindeligt Link bare smartere, den kan automatisk tilføje en aktiv CSS klasse eller stil når linket matcher den nuværende URL.
5. **useNavigate** bruges til programmatisk navigation, så vi kan sende brugeren et andet sted hen uden et link.
6. **useLocation** returnere information om den nuværende URL så som pathname, search, hash.
7. **useSearchParams** til at læse og ændre query-parametre
8. **Outlet** er et placeholder element hvor child routes bliver vist, så vi kun ændre på selve indholdet og ikke vores Header og Navbar og Footer.

Javascript Terms

Map, Slice, Fetch

1. **Map** svare til Select i C#, vi kan transformere hvert element i en samling til noget nyt.
2. **Slice** svare til en kombination af Skip og Take i LINQ, vi kan via slice udvælge hvad vi vil have med ud i en ny samling.
3. **Fetch** bruges til hente fra et API eller URL, den returnere en Promise then eller async await.