Veebiarendus

Front-End arendus

Martti Raavel

martti.raavel@tlu.ee

Tänased teemad

- Meenutame eelmist loengut
- Tokeni dekodeerimine
- Context API
- Sisselogimisandmete haldamine konteksti abil
- Kasutajate nimekirja komponent ja selle kuvamine

Millest rääkisime eelmisel korral?

JWT

Nagu eelmisel korral nägime, siis kasutame sageli JWT-d (JSON Web Token) kasutaja autentimiseks.

- JWT on kodeeritud string, mis sisaldab kasutaja informatsiooni
- JWT koosneb kolmest osast: päis, keha ja allkiri

JWT dekodeerimine React-is

Kasutame teeki `jwt-decode`, et dekodeerida JWT React-is.

```
npm install jwt-decode
```

Kasutamine

```
import { jwtDecode } from 'jwt-decode';

const decoded = jwtDecode(token);

console.log(decoded);
```

Context API

Konteksti API on React-i poolt pakutav lahendus, mis võimaldab andmete jagamist komponentide vahel. See on alternatiiv `props` -ide edastamisele ja aitab vältida "prop drilling" probleemi, kus me peame andmeid edastama läbi mitmete komponentide.

Context API komponendid

Context API koosneb järgmistest komponentidest:

- `React.createContext()` - loob konteksti objekti
- `Provider` - komponent, mis määrab jagatavad andmed
- `Consumer` - komponent, mis tarbib konteksti andmeid
- `useContext` - hook, mis võimaldab konteksti andmetele ligipääsu

Context API kasutamine

Loome eraldi faili, kus sees hakkame hoidma autentimisega seotud andmeid.

```
// AuthContext.js
import { createContext } from 'react';

// Loome konteksti objekti ja ekspordime selle
export const AuthContext = createContext();
```

Context API kasutamine

Lisame samasse faili `Provider` komponendi, mis määrab jagatavad andmed.

```
// AuthContext.js
import { createContext, useState } from 'react';

export const AuthContext = createContext();

const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);

  return (
    <AuthContext.Provider value={{ user, setUser }}>
      {children}
    </AuthContext.Provider>
  );
};

export default AuthProvider;
```

Context API kasutamine

Nüüd tuleb `AuthProvider` komponent lisada `App` komponendi sisse nii, et see ümbritseks kõiki komponente, mis vajavad autentimisega seotud andmeid.

```
// App.js

import { AuthProvider } from './AuthContext';

const App = () => (
  <AuthProvider>
    <Main />
  </AuthProvider>
);
```

Context API kasutamine

Nüüd saame kasutada `useContext` hooki, et ligipääseda konteksti andmetele.

```
// Login.js

import { useContext } from 'react';

import AuthContext from './AuthContext';

const Login = () => {
  // Võtame konteksti andmed - pane tähele, et siin on tegemist objektiga, mitte massiiviga
  const { user, setUser } = useContext(AuthContext);

  return (
    <div>
      {
        user ? (
          <p>Tere, {user.name}</p>
        ) : (
          <button onClick={() => setUser
          ({ name: 'User' })}>Logi sisse</button>
        )
      }
    </div>
  );
};

export default Login;
```

Context API kasutamine

Samuti saame konteksti lisada funktsioonid, mis muudavad konteksti andmeid.

```
// AuthContext.js

export const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);

  const login = () => {
    setUser({ name: 'User' });
  };

  const logout = () => {
    setUser(null);
  };

  return (
    <AuthContext.Provider value={{ user, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
};
```

Context API kasutamine

Nüüd saame kasutada `login` ja `logout` funktsioone `Login` komponendis.

```
// Login.js

import { useContext } from 'react';

import AuthContext from './AuthContext';

const Login = () => {
  const { user, login, logout } = useContext(AuthContext);

  return (
    <div>
      {
        user ? (
          <button onClick={logout}>Logi välja</button>
        ) : (
          <button onClick={login}>Logi sisse</button>
        )
      }
    </div>
  );
};

export default Login;
```

Kasutajate nimekirja komponendi kuvamine

Põhimõtteliselt saame nüüd ka oma rakenduse navigeerimise menüüs ümber kirjutada selliselt, et kasutajate nimekirja komponent kuvatakse ainult siis, kui kasutaja on sisse logitud ja ta on administraator.

```
import { AuthContext } from './AuthContext';

function NavBar() {
  const { user } = useContext(AuthContext);

  return (
...
    <Nav.Link as={Link} to='/posts'>Posts</Nav.Link>
    {user?.role === 'admin' && <Nav.Link as={Link} to='/users'>Users</Nav.Link>}
    <Nav.Link as={Link} to='/about'>About</Nav.Link>
  </Nav>
...
  );
};
```