Veebiarendus

Front-End arendus

Martti Raavel

martti.raavel@tlu.ee

Tänased teemad

- Eelmise loengu meenutamine
- React Router
- Not Found leht
- React Forms

Millest rääkisime eelmisel korral?

React Router / Marsruutimine

Marsruutimine võimaldab veebirakendustel pakkuda erinevaid vaateid või lehti, mis vastavad kasutaja poolt sisestatud URL-ile. Marsruutimine võimaldab navigeerida lehtede vahel ilma kogu lehte uuesti laadimata, pakkudes seega sujuvamat ja paremat kasutajakogemust.

React Router-i komponendid

- **BrowserRouter:** Peamine konteiner, mis haldab URL-ide ajalugu ja pakub konteksti marsruutide jaoks.
- **Routes:** Komponent, mis sisaldab erinevaid marsruute.
- **Route:** Marsruudi määratlemiseks ja vastava komponendi renderdamiseks kasutatav komponent.
- **Link:** Kasutatakse, et luua linke, mis võimaldavad navigeerimist ilma lehte uuesti laadimata.

React Router-i paigaldamine

```
npm install react-router-dom
```

React Router-i kasutamine

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import Home from './Home';
import About from './About';

function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </Router>
  );
}

export default App;
```

Navigatsioon

```
import React from 'react';
import { Link } from 'react-router-dom';

function Navigation() {
  return (
    <nav>
      <ul>
        <li><Link to="/">Home</Link></li>
        <li><Link to="/about">About</Link></li>
      </ul>
    </nav>
  );
}

export default Navigation;
```

Koodi abil navigeerimine

Vahel on vaja navigeerida programmiliselt, näiteks pärast sisselogimist, kui kasutaja suunatakse teisele lehele.

```
import { useNavigate } from 'react-router-dom';

function Login() {
  const navigate = useNavigate();

  function handleLogin() {
    // Login logic
    navigate('/dashboard');
  }

  return (
    <button onClick={handleLogin}>Login</button>
  );
}
```

Not Found leht

Kui kasutaja sisestab URL-i, mis ei vasta ühelegi marsruudile, siis kuvatakse Not Found leht.

```
<Router>
  <Routes>
    <Route path="/" element={<Home />} />
    <Route path="/about" element={<About />} />
    <Route path="/login" element={<Login />} />
    <Route path="*" element={<NotFound />} />
  </Routes>
</Router>
```

React Forms

Vormid on oluline osa veebirakendustest, kuna need võimaldavad kasutajatel sisestada ja esitada andmeid. React võimaldab luua vorme, mis on dünaamilised ja reageerivad kasutaja sisestatud andmetele.

React Forms komponendid

- Kontrollitud komponendid
- Kontrollimata komponendid

Kontrollitud komponendid

Kontrollitud komponentides juhib komponentide olek vormielementide väärtusi. Kõik andmed sisendites on salvestatud komponendi olekus ja värskendatud sündmuste käsitlejate kaudu.

Kontrollitud komponendid - näide

```
import React, { useState } from 'react';

function ControlledForm() {
  const [name, setName] = useState('');

  function handleChange(event) {
    setName(event.target.value);
  }

  function handleSubmit(event) {
    event.preventDefault();
    alert(`Hello, ${name}!`);
  }

  return (
    <form onSubmit={handleSubmit}>
      <input type="text" value={name} onChange={handleChange} />
      <button type="submit">Submit</button>
    </form>
  );
}
```

Kontrollimata komponendid

Kontrollimata komponentides ei ole vormielementide väärtused seotud komponendi olekuga. Vormielementide väärtusi saab lugeda DOM-ist kasutades `ref` -e, et pääseda ligi vormi väärtustele.

> `useRef` on Reacti hook, mis loob muutuja, millel on viide DOM-ile või komponendile.

Kontrollimata komponendid - näide

```
import React, { useRef } from 'react';

function UncontrolledForm() {
  const nameRef = useRef();

  function handleSubmit(event) {
    event.preventDefault();
    alert(`Hello, ${nameRef.current.value}!`);
  }

  return (
    <form onSubmit={handleSubmit}>
      <input type="text" ref={nameRef} />
      <button type="submit">Submit</button>
    </form>
  );
}
```

Mis siis, kui on vormil palju välju?

Kui vormil on palju välju, siis on mõistlik kasutada vormi andmete haldamiseks objekti.

```
import React, { useState } from 'react';

function ComplexForm() {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    password: '',
  });

  function handleChange(event) {
    const { name, value } = event.target;
    setFormData({ ...formData, [name]: value });
  }

  function handleSubmit(event) {
    event.preventDefault();
    alert(`Hello, ${formData.name}!`);
  }
...
```

```
...
  return (
    <form onSubmit={handleSubmit}>
      <input type="text" name="name" value={formData.name} onChange={handleChange} />
      <input type="email" name="email" value={formData.email} onChange={handleChange} />
      <input type="password" name="password" value={formData.password} onChange={handleChange} />
      <button type="submit">Submit</button>
    </form>
  );
}
```

Kodune töö

- Rakenda oma veebirakenduses React Router-it
- Lisa Not Found leht
- Lisa oma rakendusele sisselogimise vorm ja loe vormilt kasutaja sisestatud andmed