Veebiarendus

Front-End arendus

Martti Raavel

martti.raavel@tlu.ee

Tänased teemad

- Meenutame eelmist loengut
- Pagination
- React rakenduse deploy-mine
- Millest me kursuse jooksul ei rääkinud
- Kursuse kokkuvõte ja tagasiside
- Mis edasi?

Millest rääkisime eelmisel korral?

Pagination (lehekülgede jagamine)

Pagination on levinud viis suurte andmekogumite jagamiseks väiksemateks osadeks. See aitab vähendada lehe laadimise aega ja muudab kasutajaliidese kasutajasõbralikumaks.

Samuti aitab see vähendada serveri ja interneti ühenduse koormust, kuna me ei lae korraga kõiki andmeid.

Kuidas pagination töötab?

Põhimõtteliselt eeldab see seda, et meil on võimalik API-st pärida andmeid lehtede kaupa. Näiteks, kui meil on 1000 kasutajat, siis me ei lae korraga kõiki kasutajaid, vaid näiteks 10 kasutajat korraga. Sageli saame selleni jõuda, kasutades päringu parameetreid nagu `page` ja `limit`. Näiteks, `/users?page=1&limit=10`. Võidakse kasutada ka teistsuguseid päringu parameetreid nagu `from` ja `per_page` jms.

> Oluline on meeles pidada, et server ei pruugi alati toetada pagination-it. Ja kui toetab, siis võib see olla lahendatud erinevalt.

React Bootstrap Pagination komponent

React Bootstrap pakub `Pagination` komponenti, mis võimaldab meil luua lihtsa ja stiilse pagination-i ja selle abil saame ehitada ise eraldiseisva pagination komponendi, mida saame kasutada oma rakendustes erinevates kohtades.

Näide: Pagination React-is koos React Bootstrap-iga

Lisame oma rakendusele eraldi `PaginationComponent` komponendi, mis võimaldab meil kuvada andmeid lehtede kaupa.

PaginationComponent

```
import React from 'react';
import { Pagination } from 'react-bootstrap';

const PaginationComponent = ({ currentPage, totalPages, onPageChange }) => {
  const pageNumbers = [];

  for (let i = 1; i <= totalPages; i++) {
    pageNumbers.push(i);
  };

  return (
    <Pagination>
      <Pagination.First onClick={() => onPageChange(1)} disabled={currentPage=== 1} />
      <Pagination.Prev onClick={() => onPageChange(currentPage - 1)} disabled={currentPage === 1} />
      {pageNumbers.map(number => (
        <Pagination.Item key={number} active={number === currentPage} onClick={() => onPageChange(number)}>
          {number}
        </Pagination.Item>
      ))}
      <Pagination.Next onClick={() => onPageChange(currentPage + 1)} disabled={currentPage === totalPages} />
      <Pagination.Last onClick={() => onPageChange(totalPages)} disabled={currentPage === totalPages} />
    </Pagination>
  );
};

export default PaginationComponent;
```

PaginationComponent-i kasutamine

Kui me impordime `PaginationComponent` -i oma rakendusse, siis selle kasutamiseks peame nüüd arvestama mitme asjaga:

- meil peab olema võimalik API-st pärida andmeid lehtede kaupa
- me peame järge pidama selle üle, milline on praegune aktiivne lehekülg
- me peame teadma seda, mitu lehekülge on kokku (ja mitu elementi on ühel lehel)
- kui kasutaja valib uue lehekülje, siis peame tegema uue päringu API-sse
- kui API-sse teha päring, siis peame uuendama ka meie rakenduse olekut

PaginationComponent-i kasutamine - komponent ise

```
import PaginationComponent from './PaginationComponent';
...
{posts && <PaginationComponent
  totalPages={pagination.totalPages}
  currentPage={currentPage}
  onPageChange={(pageNumber) => setCurrentPage(pageNumber)}
/>
}
...
```

- näitame `PaginationComponent`-i ainult siis, kui meil on andmed olemas
- anname `PaginationComponent`-ile edasi vajalikud parameetrid
- kui kasutaja vahetab lehekülge, siis uuendame rakenduse olekut

API päringu tegemine

```
...
const [pagination, setPagination] = useState(null); // paginatsiooni andmed
const [currentPage, setCurrentPage] = useState(1); // praegune lehekülg
const limit = 10; // lehekülje pikkus (postituste arv lehel)
...
const token = localStorage.getItem('token');
  const response = await axios.get(`https://blog.hk.tlu.ee/posts?page=${currentPage}&limit=${limit}`, {
    headers: {
      Authorization: `Bearer ${token}`,
    },
  });
  setPosts(response.data.posts);
  setPagination(response.data.pagination);
```

- päringu tegemisel peame arvestama, et me anname API-le edasi ka lehekülje numbri ja lehekülje suuruse
- API tagastab meile ka paginatsiooni andmed, mida me kasutame `PaginationComponent` -i jaoks
- kui me saame vastuse, siis uuendame rakenduse olekut (`posts` ja `pagination`)

Nimekirja värskendamine lehekülgede vahetamisel

```
useEffect(() => {
  fetchPosts();
}, [currentPage]);
```

- kasutame `useEffect` hook-i, et teha päring serverisse, kui komponent laetakse
- kui kasutaja vahetab lehekülge, siis käivitame uuesti `fetchPosts` funktsiooni (kuna `currentPage` on sõltuvus)

React rakenduse deploy-mine

Kui me oleme oma React rakenduse valmis saanud, siis soovime tõenäoliselt selle ka kuhugi üles laadida, et seda saaks üle interneti kasutada. Selleks tuleb teha kõigepealt mõned ettevalmistused ja seejärel valida sobiv meetod ja koht.

Sel korral räägime sellest, kuidas deploy-da oma React rakendus GitHub Pages-i.

Deploy-mise ettevalmistused

- Loo omale projekti jaoks repositoorium
- Loo/kopeeri sinna oma React-i rakendus

Deploy-mise sammud

- Paigalda `gh-pages` moodul
  - `npm install gh-pages`

Package.json faili konfigureerimine

Lisa `package.json` faili järgmised read:

Esimene rida määrab, kus meie rakendus asub:

```
"homepage": "https://<Githubi kasutajanimi>.github.io/<repositooriumi nimi>",
```

Lisa `scripts` jaotisse järgmised read:

```
  "predeploy": "npm run build",
  "deploy": "gh-pages -d build"
```

Deploy töövoo käivitamine

```
npm run deploy
```

GitHub Pages-i seadistamine

- Mine projekti seadistustesse:
  - Settings
  - Pages
    - Source: `Deploy from branch`
    - Branch: `gh-pages` `/root`
  - Mine aadressile: `https://<Githubi kasutajanimi>.github.io/<repositooriumi nimi>`

Edasised sammud

Kui kõik on õigesti tehtud, peaks teie React-i rakendus olema edukalt deploy-d GitHub Pages-i.

Nüüd edaspidi, kui soovite oma rakendust uuendada, siis peate tegema järgmised sammud:

- Tee muudatused
- Käivita `deploy` töövoog: `npm run deploy`

React Router ja GitHub Pages

Kui kasutad oma akenduses React Router-it, siis Github Pages ei pruugi töötada nii nagu peaks. Selleks, et Router korralikult töötaks, peame oma rakenduses vahetama `BrowserRouter` `HashRouter` vastu.

```
import { HashRouter as Router } from 'react-router-dom';
```

See muudatus tagab, et React Router töötab korralikult ka GitHub Pages-i puhul.

Millest me kursuse jooksul ei rääkinud?

- Front-End rakenduse testimine
- React-i server-side renderdamine (SSR)
- Dokumentatsiooni loomine (pisut rääkisime, kuid teema on oluliselt laiem)
- Failide üleslaadimine (nt pildid, dokumendid)
- Failide optimeerimine (pildid, skriptid, stiilid)
- SEO (Search Engine Optimization)
- Keelevalik
- ...

Kursuse kokkuvõte ja tagasiside

- Mis oli hästi?
- Mida saaks paremini teha?
- Millest oleks veel tahtnud kuulda?
- Veel midagi?

Mis edasi?