Web 3

React Router

HO
GENT

Gebruikte technologieën

- Fetch API
- Axios
- React Router

HO
GENT

Consumeren van API’s

- Belangrijk onderdeel van webapplicaties tegenwoordig. Het **consumeren** van een **REST API**
- **Supercharge** onze **React** applicatie met **data**
- Twee mogelijkheden: **JavaScript Fetch API** en **Axios**
- In React maken we gebruik van de **useEffect hook** om de **data op** te **halen** van de REST API

HO
GENT

Fetch API

HO
GENT

JavaScript Fetch API

- De **`fetch()`** API is een **ingebouwde methode** in JavaScript om **data** van **server resources** of een **API** te **ontvangen**
- De **`fetch()`** API methode bezit steeds **één verplicht argument**, namelijk de **URL** naar de resource
- Deze methode **geeft** een **Promise terug** die dan de **response** terug geeft

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

HO GENT

JavaScript Fetch API

GET

```
// GET request - JSONPlaceholder API

fetch('https://jsonplaceholder.typicode.com/posts')
  .then(response => {
    return response.json();
  })
  .then(data => {
    console.log(data);
  })
  .catch(err => {
    console.log(err);
  });
```

HO GENT

JavaScript Fetch API

POST

```
// POST request - JSONPlaceholder API - data versturen + headers meegeven

fetch('https://jsonplaceholder.typicode.com/posts', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    title: 'Mijn titel',
    body: 'Mijn bericht',
    userId: 1
  })
})
.then(response => response.json())
.then(data => console.log(data))
.catch(err => console.log(err));
```

HO GENT

JavaScript Fetch API

HTTP Cookie

```
// GET request - JSONPlaceholder API - HTTP only cookies meesturen met de
request

fetch('https://jsonplaceholder.typicode.com/posts', {
  credentials: 'include'
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error(error));
```

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

HO
GENT

Axios

- **Axios** is een **gemakkelijk** te gebruiken **Promise gebaseerde HTTP client** voor de **browser** en **Node.js**
- Het **ondersteunt** ook **meer browsers** als de **Fetch API**
- **JSON** data **transformatie** wordt **automatisch** gedaan
- **Beveiliging** tegen **XSRF**

```
$ npm install axios

        OF

$ yarn add axios
```

https://www.npmjs.com/package/axios

HO GENT

Axios

Methodes

```
// GET request a.h.v. Axios - JSONPlaceholder API

axios.get('https://jsonplaceholder.typicode.com/todos/1')
  .then(response => console.log(response.data))
  .catch(error => console.error(error));
```

```
// POST request a.h.v. Axios - JSONPlaceholder API - data versturen

const data = {
  title: 'My Todo',
  completed: false
};

axios.post('https://jsonplaceholder.typicode.com/todos', data)
  .then(response => console.log(response.data))
  .catch(error => console.error(error));
```

Axios

Axios-functie

```
// GET request a.h.v. Axios - JSONPlaceholder API

axios({
  method: 'get',
  url: 'https://jsonplaceholder.typicode.com/todos/1'
})
.then(response => console.log(response.data))
.catch(error => console.error(error));
```

```
// POST request a.h.v. Axios - JSONPlaceholder API - data versturen

const data = {
  title: 'My Todo',
  completed: false
};

axios({
  method: 'post',
  url: 'https://jsonplaceholder.typicode.com/todos',
  data: data
})
.then(response => console.log(response.data))
.catch(error => console.error(error));
```

Axios

React

```
const MyComponent = (props) => {

  const [data, setData] = useState(null);
  const [isLoading, setIsLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await axios.get("https://jsonplaceholder.typicode.com/todos/1");
        setData(response.data);
      } catch(err) {
        setError(err.message);
      } finally {
        setIsLoading(false);
      }
    };

    fetchData();
  }, []);

  ...
}
```

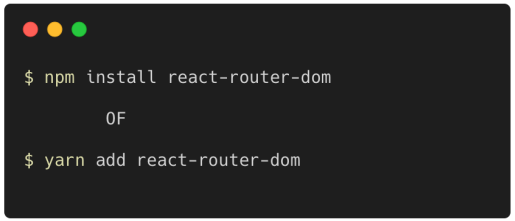
HO GENT

React Router

Routing

- Het is **gebruikelijk** om **meerdere pagina's** te hebben in websites
- Het **punt** van **React** is het zijn van een **Single Page Application** (we zijn altijd op dezelfde pagina)
- **Navigatie** moet dus **mogelijk gemaakt** worden door **React** zelf of een library natuurlijk
- Hiervoor kunnen we gebruik maken van de **React-Router library**

https://reactrouter.com/



HO GENT

Routing (2)

- In **web applicaties** maken we vooral gebruik van het **react-router-dom** pakket. De **React Router** library heeft ook **ondersteuning** voor **routing** in native applicaties (React Native)
- **Achterliggend** maakt de **BrowserRouter** gebruik van de **HTML5 history API** (pushState, replaceState, popState) om uw **UI gesynchroniseerd** te houden met de **URL**

Routing

RouterProvider

- De RouterProvider component is een **hogere-orde-component** die wordt gebruikt om de **navigatie** in de **applicatie** te **beheren**
- Verschillende soorten kunnen meegegeven worden aan de **router prop** nl.: **BrowserRouter**, **HashRouter** en **MemoryRouter**

HO GENT

Routing

createBrowserRouter

- De **createBrowserRouter methode** zorgt ervoor dat we onze **routes** op een **declaratieve manier** kunnen **definiëren**
- Deze verwacht een **lijst** met alle **route objecten** in
- Deze **methode** heeft het **voordeel** dat we ook **loaders** kunnen **meegeven** om **data** al te **laden voor** dat we naar de **route navigeren**

HO GENT

Routing

Route object

- Met een **route** object kan je **bepalen welke component gerenderd** moet worden op **basis** van de **huidige URL**
- Een **route object** bevat een **`path` property** die aangeeft **welke URL** moet **overeenkomen** met de **route** en een **`element` property** die aangeeft **welke component gerenderd** moet worden

HO GENT

Routing

Link

- De **Link** component kan gebruikt worden om **tussen verschillende pagina's** in de **applicatie** te **navigeren**
- De **URL** wordt **veranderd zonder** de pagina te **herladen**
- Er wordt hiervoor **gebruik** gemaakt van een **`to` prop** waarbij je de **URL/pad meegeeft** om te **navigeren**

```
<Link to="/notes">Notes</Link>
```

HO GENT

Routing

Voorbeeld

```
import React from 'react';
import { createBrowserRouter, RouterProvider } from 'react-router-dom';

import Home from './Home';
import About from './About';
import Contact from './Contact';
import NotFound from './NotFound';


const browserRouter = createBrowserRouter([
  {
    path: "/",
    element: <Home />
  },
  {
    path: "/about",
    element: <About />
  },
  {
    path: "/contact",
    element: <Contact />
  },
  {
    path: "*",
    element: <NotFound />
  }
])

function App() {
  return <RouterProvider router={browserRouter} />
}

export default App;
```

HO GENT

Routing

useParams()

- **Tot nu** toe steeds **statische URL's** gebruikt. We kunnen ook **parameters doorgeven** aan de **URL**.
- Vaak **gebruikt** om **dynamische content** te tonen op **basis** van de **huidige URL**
- Je voegt dit **eerst** toe aan de **Route component** met het volgende teken en de naam van de parameter bvb. **`:userId`**
- Ten slotte kan je dan **gebruik maken** van de **`useParams()` hook** om de **parameters op** te **vragen** in de **component**

HO
GENT

Routing

useParams() Voorbeeld

```
// Eerste gedeelte gedefinieerd in de createBrowserRouter methode
{
  path: "/users/:userId",
  element: <UserDetail />
}
```

```
// Tweede gedeelte in de component waar je de parameter wenst te gebruiken
import { useParams } from "react-router-dom";

const UserDetail = () => {
    const { userId } = useParams();

  return (
    <p>User met id: {userId}</p>;
  )
}
```

Routing

useNavigate()

- **Navigeren** tussen **verschillende pagina's** in uw **applicatie** kan ook gedaan worden via **`useNavigate().`**
- Wordt **meestal** gebruikt **nadat** er op een **knop** werd **geklikt** of wanneer er een **bepaalde actie** werd **voltooid**.
- **Eenvoudige manier** om te **navigeren** tussen **routes zonder** nood aan directe **toegang** tot de **browsergeschiedenis**

HO
GENT

Routing

useNavigate() Voorbeeld

```
// De useNavigate hook gebruiken om te navigeren naar een andere route
// Alsook met het meegeven van data - de about pagina moet wel bestaan

import { useNavigate } from "react-router-dom";

const Home = () => {
  const navigate = useNavigate();

  const handleClick = () => {
    // Navigeren zonder extra data mee te geven
    navigate('/about');

    // Navigeren met extra data mee te geven
    navigate('/about', { contactId: 123});


  return (
    <button onClick={handleClick}>Ga naar de about pagina</button>;
  )
}
```

OEFENING

- Maak een React applicatie met drie pagina’s
  - De **eerste pagina** is de **homepagina**. Waarop je een **welkomst boodschap** opzet en een link naar de apis pagina
  - De **tweede pagina** is de pagina waar **alle apis** op staan, een lijst met de verkregen apis. Dit zijn **linken** om naar de **derde** pagina **te** kunnen **gaan**
  - De **derde pagina** is de pagina waar een **specifieke api** staat met zijn **beschrijving** en een **link** naar de **api** in kwestie. Dit wordt getoond door middel van een **id** in het **pad** van de **URL**. Deze id mag gewoon de index zijn van de array die je hebt op de tweede pagina
- De link om alle apis te downloaden is deze https://api.publicapis.org/entries?category=development

HO GENT

OEFENING

- Je krijgt een JSON object terug waaruit je de entries moet halen als array
- Je mag gebruik maken van de Fetch API of van Axios om de data op te halen. Denk ook goed na waar je de data gaat ophalen in welke component?
- De styling mag je wederom zelf kiezen (laat je creativiteit werken)

OEFENING

- 24 Pull Requests
- Agify.io
- ApiFlash
- Apility.io
- APIs.guru
- BetterMeta
- Bitbucket
- Bored
- Browshot
- CDNJS
- Changelogs.md
- CountAPI
- DigitalOcean Status
- DomainDb Info
- Genderize.io
- GitHub
- Gitlab
- Gitter
- HTTP2.Pro
- IBM Text to Speech
- IBM Visual Recognition
- IFTTT
- Image-Charts
- import.io
- IPify
- IPinfo
- JSON 2 JSONP
- JSON Pretty Print
- JSONbin.io
- Judge0
- License-API
- MAC address vendor lookup
- Nationalize.io
- OOPSpam
- Pliino
- Postman
- ProxyCrawl
- Public APIs
- Pusher Beams
- QR code
- QR code
- QuickChart
- ReqRes
- Scraper.AI
- ScraperApi
- ScreenshotAPI.net
- SHOUTCLOUD
- StackExchange

Agify.io

Estimates the age from a first name

https://agify.io

HO
GENT