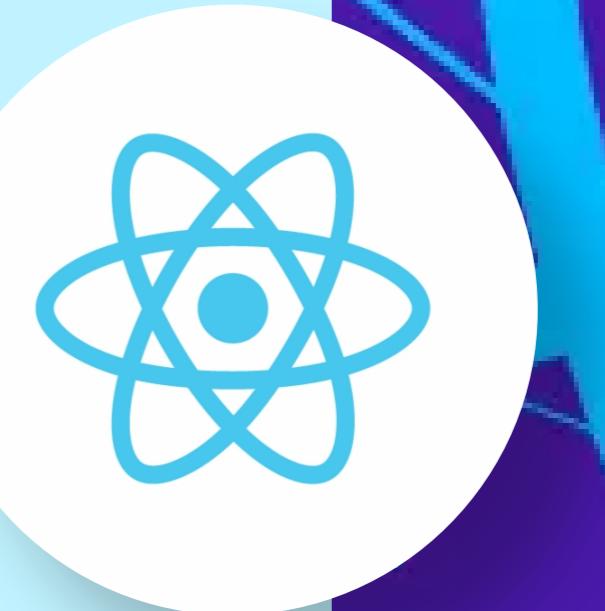


REACT basics

Les 16:

React Router & Fetch() API



Onderwerpen les:

- Routes en Route
- Link en NavLink
- Fetch() API

Leerdoelen

- ✓ Aan het einde van de dag kun je React Router gebruiken om componenten te laden aan de hand van een path
- ✓ Aan het einde van de dag kun je een nav bouwen geschikt voor React Router
- ✓ Aan het einde van de dag kun je data ophalen met Fetch()

Routes en Route

Wat is React Router DOM?

React ROUTER DOM is een npm package die er voor zorgt dat je dynamische routing kan implementeren in een web app. Hiermee kunnen bezoekers navigeren door verschillende pagina's zonder dat de gehele pagina herladen wordt.

<BrowserRouter>

De Router is de parent component dat wordt gebruikt om alle components in te plaatsen. Hierbinnen zal alles deel uitmaken van de routing functionaliteit.

Route

Dit component checked de URL en toont de bijbehorende component die horen bij het pad.

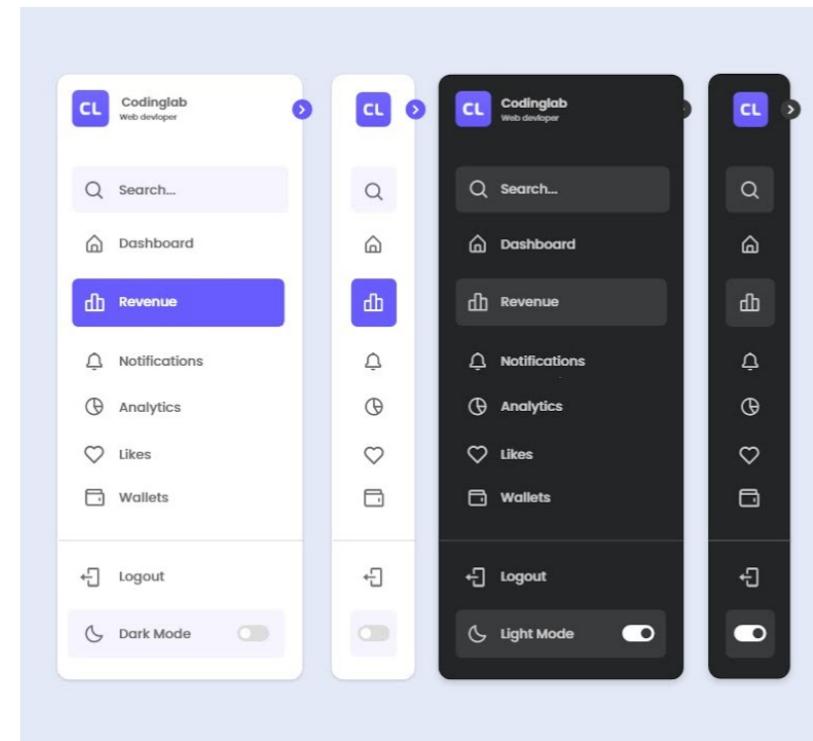
Link / NavLink

Link en NavLink components worden gemaakt om naar de verschillende Routes te linken.

Routes en Route

Link en NavLink

In plaats van de a tag gebruik je met React Router de Link of NavLink component. In de DOM zal dit worden omgezet naar ee a tag. Met to="/paginanaam" geef je aan waar de link naar toe verwijst. De NavLink krijgt een active class die je kan stijlen in je css.



Routes en Route

Router gebruiken

React ROUTER DOM is een npm package die er voor zorgt dat je dynamische routing kan implementeren in een web app. Hiermee kunnen bezoekers navigeren door verschillende pagina's zonder dat de gehele pagina herladen wordt.

Stap 1: Installeren react router dom

Ga in command/prompt naar het project folder en typ:
npm i react-router-dom

Stap 2: <BrowserRouter> importeren en gebruiken

Importeer in index.js de BrowserRouter en plaats de app component in de BrowserRouter component

Gebruik Routes en Route in de app component

Maak een Routes component en plaats daar de Route components in. Deze components krijgen een path (relatief pad naar de pagina) en een element of component waar wordt verwezen naar de juiste Component.

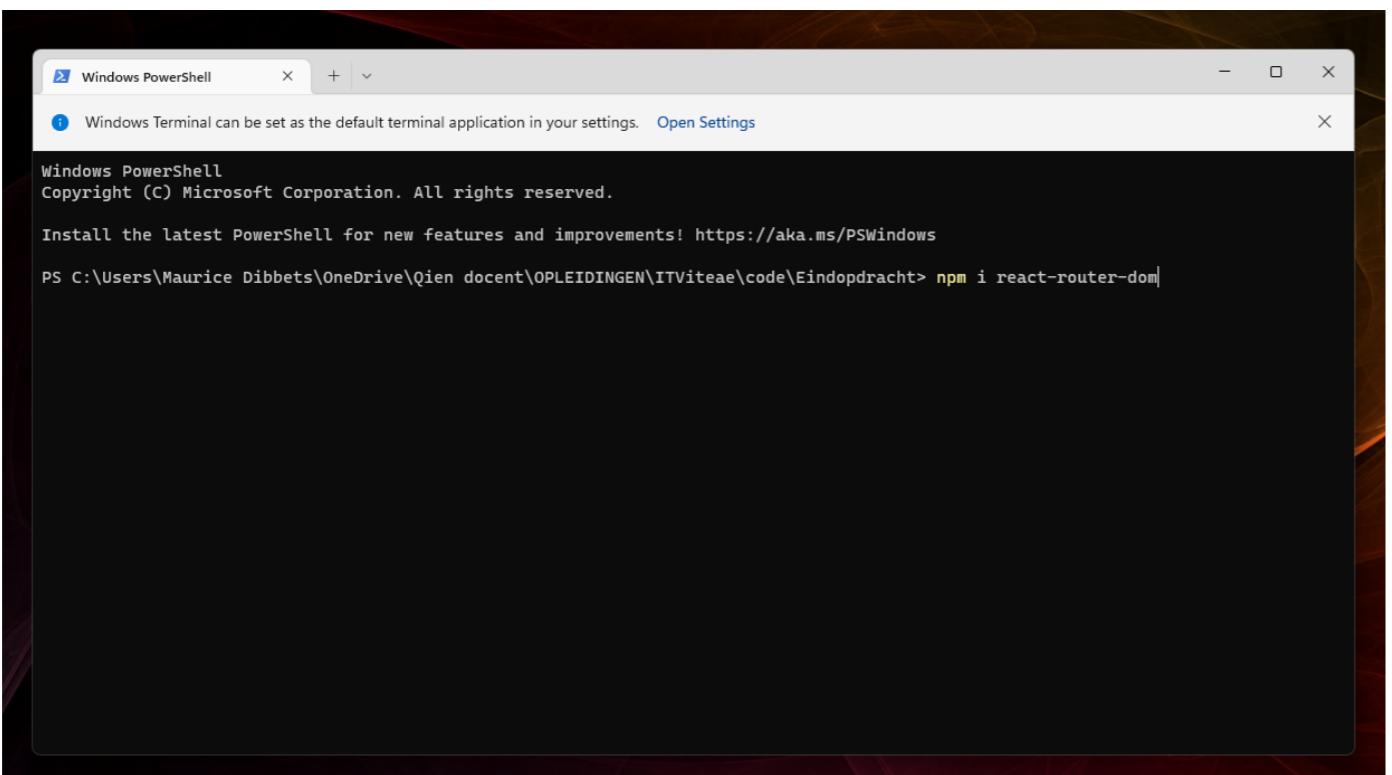
Gebruik Link of NavLink om te verwijzen naar een url

I.p.v. a tags gebruiken we Link en NavLink componenten voor onze urls.

Routes en Route

Router gebruiken

1 instal react router dom



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window shows the command "npm i react-router-dom" being typed at the prompt. The terminal interface includes standard PowerShell branding like the "Windows PowerShell" logo and the Microsoft copyright notice.

- Ga naar de juiste folder: cd "path here"
- npm i react-router-dom

Routes en Route

Router gebruiken

2 BrowserRouter

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6 import { BrowserRouter } from 'react-router-dom';
7
8 const root = ReactDOM.createRoot(document.getElementById('root'));
9 root.render(
10   <BrowserRouter>
11     <App />
12   </BrowserRouter>
13 );
```

- Importeer BrowserRouter
- Plaats de App component in de BrowserRouter component

Routes en Route

Router gebruiken

3 Routes maken

```
42 <Routes>
43   <Route path="/" element={<Home />} /> {/*mag ook JSX zijn bv een h1*/}
44   <Route path="/characters" element={<Characters />} />
45   <Route path="/planets" element={<Planets />} />
46   <Route path="/character/:id" element={<Character />} />
47 </Routes>
```

- Plaats binnen een Routes component alle Route components die je nodig hebt
- Geef path en element/component een value

Routes en Route

Router gebruiken

4 Link / NavLink

```
29 <nav>
30   <ul>
31     <li>
32       <NavLink to="/">Home</NavLink>
33     </li>
34     <li>
35       <NavLink to="/characters">Characters</NavLink>
36     </li>
37     <li>
38       <NavLink to="/planets">Planets</NavLink>
39     </li>
40   </ul>
41 </nav>
```

- Maak Link of NavLink components aan en verwijst naar de juiste URL

Fetch() API

Wat is Fetch API?

Met de Fetch API beschik je over een globale fetch methode waarmee je makkelijk data zoals JSON asynchroon kan ophalen van een endpoint of bestand. Fetch is dus een tegenhanger van de XML Http Requests die in een vorige les is behandeld.

```
fetch('URL')
```

Een simpele fetch neemt 1 argument en dat is het pad van de bron waarvan je iets wil ophalen (endpoint)

```
.then((response) => response.json())
```

Het response object is de gehele HTTP response en hiervan extraheren we de JSON.

```
.then((data) => console.log(data))
```

Het resultaat van de response.json() kan gebruikt worden als data. Hier kan je mee doen wat je wil. (gebruiken in de state van een object etc)

```
.catch((error) => {  
  console.log(error)  
})
```

Is er een error vang deze dan op met een catch block. Dit werkt alleen met netwerk errors. Andere errors zul je zelf kunnen opvangen in de response.

Fetch() API

Fetch promises

Een promise object representeert de uiteindelijke voltooiing (of mislukking) van een asynchrone bewerking en de resulterende waarde.

Wanneer een belofte is vervuld, betekent dit dat de asynchrone bewerking is voltooid en dat de belofte een waarde heeft. Wanneer een belofte wordt afgewezen, betekent dit dat de asynchrone bewerking is mislukt en dat de belofte nooit zal worden vervuld.

ComponentDidMount()

componentDidMount() wordt aangeroepen onmiddellijk nadat een component is aangekoppeld (ingevoegd in de boomstructuur). Initialisatie waarvoor DOM-knooppunten nodig zijn, moet hier komen. Als je gegevens van een extern eindpunt moet laden, is dit een goede plaats om het netwerkverzoek te instantiëren.

Je kunt setState() onmiddellijk aanroepen in componentDidMount(). Het zal een extra weergave activeren, maar het zal gebeuren voordat de browser het scherm bijwerkt. Dit garandeert dat hoewel render() in dit geval twee keer wordt aangeroepen, de gebruiker de tussenliggende status niet ziet.

Fetch() API

Fetch toepassen

```
14  componentDidMount() {
15    fetch('https://swapi.dev/api/people/' + this.state.id)
16      .then((response) => response.json())
17      .then((data) => {
18        console.log(data);
19        this.setState({character: data});
20        console.log(this.state.character.name);
21      })
22      .catch((error) => {
23        console.error(error);
24      });
25 }
```