React 2 (React Movie Explorer - concept 13/1)

Vite + React

Projectnaam: "React Movie Explorer"

Beschrijving: Studenten bouwen een interactieve webapplicatie waarmee gebruikers informatie over films kunnen opzoeken. De applicatie gebruikt een externe API, zoals de OMDb API of een andere gratis films-API. Gebruikers kunnen zoeken naar films, details bekijken (zoals titel, genre, plot, en poster), en een lijst van favoriete films bijhouden.

Functionaliteiten:

1. **Zoekfunctionaliteit:**
   - Gebruiker kan een film zoeken op naam.
   - Zoekresultaten worden dynamisch weergegeven.
2. **Film Details:**
   - Klik op een film in de lijst om meer details te bekijken (bijvoorbeeld een aparte pagina of een popup).
3. **Favorieten:**
   - Gebruikers kunnen films toevoegen aan een lijst met favorieten.
   - Favorieten worden opgeslagen in de `localStorage`, zodat ze blijven bestaan na het herladen van de pagina.
4. **Responsief Design:**
   - De app werkt goed op zowel desktop als mobiel.
5. **Vite + React:**
   - Vite zorgt voor een snelle ontwikkelomgeving.

- Studenten leren component-gebaseerde ontwikkeling met React, inclusief statebeheer met `useState` en `useEffect`.

Extra Opties voor Gevorderde Studenten:

- **Filteren op genre of jaar:** Voeg extra filters toe.
- **Paginering:** Zorg dat zoekresultaten verdeeld worden over meerdere pagina's.
- **Themawisselaar:** Laat gebruikers wisselen tussen een licht en donker thema.

Benodigde tools en kennis:

1. **Tools:**
   - Vite: Voor snelle projectopzet.
   - React: Voor het bouwen van componenten.
   - API zoals OMDb API (registreer voor een gratis API-sleutel).
   - CSS-framework (optioneel): Tailwind, Bootstrap, of handgemaakte CSS.
2. **Kennis:**
   - Basis JavaScript en ES6+ (bijv. `fetch` API).
   - React-beginselen (`useState`, `useEffect`, props, componenten).
   - Basis HTML/CSS.

Projectopbouw:

1. **Les 1:** Introductie en opzet
   - Installeer Node.js, Vite, en maak een nieuw project.
   - Begrijp de folderstructuur van Vite + React.
   - Bouw een eenvoudige React-component die een "Hello World" bericht toont.
2. **Les 2:** API-integratie
   - Leer hoe je de OMDb API gebruikt met `fetch`.
   - Bouw een zoekbalk en toon resultaten.
3. **Les 3:** Statebeheer en favorieten

- Gebruik `useState` voor statebeheer.
- Voeg een favorietenfunctie toe en sla favorieten op in `localStorage`.

4. **Les 4:** Styling en afronding
   - Voeg styling toe met CSS of een framework.
   - Maak de app responsive.
   - Voeg een themawisselaar toe als bonus.

Les 1: Introductie en opzet

Doel van de les

In deze les leren studenten hoe ze een Vite + React-project opzetten, de folderstructuur begrijpen en een eenvoudige React-component bouwen die een "Hello World" bericht toont.

Stap 1: Installeren van Node.js

1. **Controleer of Node.js is geïnstalleerd:**
   - Open een terminal (Command Prompt, PowerShell of een andere terminal).
   - Typ het volgende commando:

     ```
     node -v
     ```

   - Als er een versie wordt weergegeven (bijvoorbeeld `v18.16.0`), dan is Node.js al geïnstalleerd.
2. **Installeer Node.js als dit nog niet is gebeurd:**
   - Ga naar de officiële Node.js-website.
   - Download de LTS-versie (Long Term Support).
   - Volg de installatie-instructies.
3. **Controleer ook npm (Node Package Manager):**
   - Typ in de terminal:

     ```
     npm -v
     ```

   - Dit toont de versie van npm. Het wordt automatisch met Node.js geïnstalleerd.

Stap 2: Een Vite + React-project opzetten

1. **Maak een nieuw project aan met Vite:**
   - Typ in de terminal:

     ```
     npm create vite@latest my-react-app
     ```

   - Kies de volgende opties:
     - **Projectnaam:** `my-react-app` (of een andere naam naar keuze).
     - **Framework:** `React`.
     - **Variant:** `JavaScript` (voor beginners).
2. **Navigeer naar de projectmap:**

   ```
   cd my-react-app
   ```

3. **Installeer de benodigde afhankelijkheden:**

   ```
   npm install
   ```

4. **Start de ontwikkelserver:**

   ```
   npm run dev
   ```

   - Noteer het adres (meestal `http://localhost:5173`) dat in de terminal wordt weergegeven.
   - Open dit adres in een webbrowser. Je zou een standaard Vite-react-startpagina moeten zien.

Stap 3: Begrijp de folderstructuur

In de projectmap zie je de volgende belangrijke mappen en bestanden:

- **`src/`**
  - **`main.jsx`**: Het startpunt van de applicatie. Hier wordt React geïntegreerd met de browser.
  - **`App.jsx`**: De hoofdcomponent van de applicatie.
- **`index.html`**: De HTML-pagina waarin de React-app wordt geladen.
- **`vite.config.js`**: Configuratiebestand voor Vite.

Leg uit dat `src/` de map is waar alle React-code zich bevindt.

Stap 4: Bouw een eenvoudige React-component

1. **Open de `App.jsx` in een code-editor (bijv. VS Code).**
2. **Pas de inhoud aan om een eenvoudige "Hello World"-component te maken:**

```
function App() {
  return (
    <div>
      <h1>Hello World</h1>
      <p>Welkom bij je eerste React-app!</p>
    </div>
  );
}

export default App;
```

3. **Opslaan en bekijken:**
   - Sla het bestand op.
   - Ga terug naar de browser en vernieuw de pagina. Je zou nu de tekst "Hello World" en "Welkom bij je eerste React-app!" moeten zien.

Extra uitdaging (optioneel)

- Voeg je eigen tekst of een afbeelding toe aan de component.
- Experimenteer met HTML-tags zoals `<button>` en `<input>`.

Samenvatting van de les

- Studenten hebben geleerd hoe ze Node.js, Vite en React instellen.
- Ze begrijpen de basis van de Vite + React-folderstructuur.
- Ze hebben een eenvoudige "Hello World" React-component gebouwd.

Voorbereiding voor les 2

- Zorg dat je `npm run dev` kunt uitvoeren en de app werkt in de browser.
- Lees alvast over de `useState`-hook in React (optioneel).

Les 2: API-integratie

Doel van de les

Studenten leren hoe ze de OMDb API kunnen gebruiken met de `fetch`-methode in JavaScript en bouwen een eenvoudige zoekfunctionaliteit om filmresultaten weer te geven in hun React-app.

Stap 1: De OMDb API begrijpen

1. **Wat is de OMDb API?**
   - Een gratis API waarmee je informatie over films kunt ophalen, zoals titel, jaar, genre, en een poster.
   - API-documentatie: OMDb API.
2. **Vraag een API-sleutel aan:**
   - Ga naar de website van OMDb en registreer je voor een gratis API-sleutel.
   - Noteer deze sleutel, want deze is nodig om de API te gebruiken.

Stap 2: Een zoekfunctionaliteit toevoegen

1. **Bewerk de `App.jsx`:**
   - Open het bestand `App.jsx`.
   - Vervang de bestaande code door onderstaande basisopzet:

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

function App() {
  const [searchTerm, setSearchTerm] = useState('');
  const [movies, setMovies] = useState([]);

  const API_KEY = 'JOUW_API_SLEUTEL_HIER';

  const searchMovies = async () => {
```

```
    const response = await
fetch(`https://www.omdbapi.com/?apikey=${API_KEY}&s=${searchTerm}`);
    const data = await response.json();
    if (data.Search) {
      setMovies(data.Search);
    } else {
      setMovies([]);
    }
  };

  return (
    <div>
      <h1>React Movie Explorer</h1>
      <div>
        <input
          type="text"
          placeholder="Zoek een film..."
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
        />
        <button onClick={searchMovies}>Zoeken</button>
      </div>
      <div>
        {movies.length > 0 ? (
          <ul>
            {movies.map((movie) => (
              <li key={movie.imdbID}>
                <h2>{movie.Title}</h2>
                <p>{movie.Year}</p>
                <img src={movie.Poster} alt={movie.Title} />
              </li>
            ))}
          </ul>
        ) : (
          <p>Geen films gevonden. Probeer een andere zoekterm.</p>
        )}
      </div>
    </div>
  );
```

```
}


export default App;
```

2. **Uitleg over de code:**
   - **Statebeheer:**
     - `searchTerm`: Houdt bij wat de gebruiker intypt in de zoekbalk.
     - `movies`: Slaat de zoekresultaten op.
   - **`fetch`:**
     - Roept de OMDb API aan met de ingevoerde zoekterm.
   - **Dynamische rendering:**
     - De lijst met films wordt weergegeven op basis van de API-resultaten.

Stap 3: Test de zoekfunctionaliteit

1. **Start de ontwikkelserver:**

```
npm run dev
```

2. **Open de app in de browser:**
   - Typ een zoekterm in (bijvoorbeeld "Avengers") en klik op de knop "Zoeken".
   - Controleer of de resultaten worden weergegeven met titel, jaar, en poster.

Stap 4: Styling toevoegen (optioneel)

1. **Basis CSS toevoegen:**
   - Maak een nieuw bestand `App.css` in de `src/` map:

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  text-align: center;
}
```

```
input {
  padding: 8px;
  margin: 10px;
  width: 200px;
}

button {
  padding: 8px 12px;
  background-color: #007BFF;
  color: white;
  border: none;
  cursor: pointer;
}

button:hover {
  background-color: #0056b3;
}

img {
  max-width: 150px;
  margin: 10px;
}

ul {
  list-style-type: none;
  padding: 0;
}

li {
  display: inline-block;
  margin: 10px;
}
```

2. **Importeer de CSS in `App.jsx`:**

```
import './App.css';
```

3. **Bekijk het resultaat:**
   - Herlaad de pagina om te zien hoe de app er nu uitziet met styling.

Samenvatting van de les

- Studenten hebben geleerd hoe ze de OMDb API kunnen gebruiken met `fetch`.
- Ze hebben een zoekbalk gemaakt en filmresultaten weergegeven.
- Optioneel: Styling is toegevoegd om de app visueel aantrekkelijk te maken.

Voorbereiding voor les 3

- Lees over het gebruik van `localStorage`.
- Bedenk hoe je een favorietenlijst kunt maken waarin gebruikers films kunnen opslaan.

Les 3: Statebeheer en favorieten

Doel van de les

Studenten leren hoe ze statebeheer met `useState` toepassen in React en een favorietenfunctionaliteit implementeren. Favorieten worden opgeslagen in `localStorage`, zodat deze behouden blijven na het herladen van de pagina.

Stap 1: Favorieten toevoegen aan state

1. **Pas de `App.jsx` aan:**
   - Open het bestand `App.jsx`.
   - Voeg een nieuwe state toe voor het opslaan van favorieten:

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

function App() {
  const [searchTerm, setSearchTerm] = useState('');
  const [movies, setMovies] = useState([]);
  const [favorites, setFavorites] = useState([]);

  const API_KEY = 'JOUW_API_SLEUTEL_HIER';

  const searchMovies = async () => {
    const response = await
```

```
fetch(`https://www.omdbapi.com/?apikey=${API_KEY}&s=${searchTerm}`);
    const data = await response.json();
    if (data.Search) {
      setMovies(data.Search);
    } else {
      setMovies([]);
    }
  };

  const addToFavorites = (movie) => {
    const updatedFavorites = [...favorites, movie];
    setFavorites(updatedFavorites);
    saveToLocalStorage(updatedFavorites);
  };

  const removeFromFavorites = (movie) => {
    const updatedFavorites = favorites.filter(fav => fav.imdbID !==
movie.imdbID);
    setFavorites(updatedFavorites);
    saveToLocalStorage(updatedFavorites);
  };

  const saveToLocalStorage = (items) => {
    localStorage.setItem('favorites', JSON.stringify(items));
  };

  const loadFavoritesFromLocalStorage = () => {
    const storedFavorites = JSON.parse(localStorage.getItem('favorites')) ||
[];
    setFavorites(storedFavorites);
  };

  useEffect(() => {
    loadFavoritesFromLocalStorage();
  }, []);

  return (
    <div>
      <h1>React Movie Explorer</h1>
```

```
      <div>
        <input
          type="text"
          placeholder="Zoek een film..."
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
        />
        <button onClick={searchMovies}>Zoeken</button>
      </div>
      <div>
        <h2>Zoekresultaten</h2>
        {movies.length > 0 ? (
          <ul>
            {movies.map((movie) => (
              <li key={movie.imdbID}>
                <h2>{movie.Title}</h2>
                <p>{movie.Year}</p>
                <img src={movie.Poster} alt={movie.Title} />
                <button onClick={() => addToFavorites(movie)}>Voeg toe aan
favorieten</button>
              </li>
            ))}
          </ul>
        ) : (
          <p>Geen films gevonden. Probeer een andere zoekterm.</p>
        )}
      </div>
      <div>
        <h2>Favorieten</h2>
        {favorites.length > 0 ? (
          <ul>
            {favorites.map((favorite) => (
              <li key={favorite.imdbID}>
                <h2>{favorite.Title}</h2>
                <p>{favorite.Year}</p>
                <img src={favorite.Poster} alt={favorite.Title} />
                <button onClick={() =>
removeFromFavorites(favorite)}>Verwijder uit favorieten</button>
              </li>
```

```
            ))}
          </ul>
        ) : (
          <p>Geen favorieten toegevoegd.</p>
        )}
      </div>
    </div>
  );
}

export default App;
```

Stap 2: Uitleg over de code

1. **Statebeheer:**
   - `favorites`: Houdt een lijst bij van favoriete films.
2. **LocalStorage:**
   - Functie `saveToLocalStorage`: Slaat de favorieten op in de browser.
   - Functie `loadFavoritesFromLocalStorage`: Laadt favorieten uit `localStorage` bij het laden van de app.
3. **Favorieten toevoegen/verwijderen:**
   - **Toevoegen:** De film wordt toegevoegd aan de favorietenlijst en opgeslagen in `localStorage`.
   - **Verwijderen:** De film wordt uit de favorietenlijst verwijderd en de wijzigingen worden opgeslagen.
4. **Gebruik van `useEffect`:**
   - `useEffect` wordt gebruikt om de opgeslagen favorieten te laden bij het opstarten van de applicatie.

Stap 3: Test de applicatie

1. **Start de ontwikkelserver:**

   ```
   npm run dev
   ```

2. **Voer een zoekopdracht uit:**
   - Zoek een film en voeg deze toe aan de favorieten.
3. **Controleer de favorietenlijst:**
   - Zie hoe de favorieten verschijnen onder de sectie "Favorieten".
4. **Herlaad de pagina:**
   - Controleer of de favorieten behouden blijven dankzij `localStorage`.

Extra uitdaging (optioneel)

- Voeg een knop toe om alle favorieten in één keer te verwijderen.
- Voeg een melding toe wanneer een film al in de favorieten staat.

Samenvatting van de les

- Studenten hebben geleerd hoe ze statebeheer toepassen met `useState`.
- Ze hebben een favorietenfunctionaliteit geïmplementeerd.
- Ze hebben `localStorage` gebruikt om gegevens op te slaan en te laden.

Voorbereiding voor les 4

- Lees over CSS-styling en het verbeteren van gebruikersinterfaces.
- Denk na over hoe je filters of paginering kunt toevoegen aan de zoekresultaten.

Les 4: Film Details en Responsief Design

Doel van de les

Studenten leren hoe ze een gedetailleerde weergave van filminformatie kunnen implementeren (bijvoorbeeld in een popup of aparte pagina) en hoe ze een responsief ontwerp kunnen toevoegen zodat de applicatie goed werkt op zowel desktop als mobiel.

Stap 1: Gedetailleerde weergave van een film

1. **Voeg een functie toe voor het ophalen van filmgegevens:**
   - Pas de `App.jsx` aan om een gedetailleerde weergave van een geselecteerde film te tonen.
   - Voeg state toe om de details van de geselecteerde film bij te houden:

```
const [selectedMovie, setSelectedMovie] = useState(null);

const fetchMovieDetails = async (movieId) => {
  try {
    const response = await
fetch(`https://www.omdbapi.com/?apikey=${API_KEY}&i=${movieId}`);
    const data = await response.json();
    setSelectedMovie(data);
  } catch (error) {
    console.error('Fout bij het ophalen van filmgegevens:', error);
  }
};
```

2. **Open details bij klikken op een film:**
   - Pas de `movies.map`-functie aan om een knop toe te voegen die de details opent:

```
{movies.map((movie) => (
  <li key={movie.imdbID}>
    <h2>{movie.Title}</h2>
    <p>{movie.Year}</p>
    <img src={movie.Poster} alt={movie.Title} />
    <button onClick={() => fetchMovieDetails(movie.imdbID)}>Bekijk
details</button>
  </li>
))}
```

3. **Toon de filmgegevens:**
   - Voeg een sectie toe om de details weer te geven:

```
{selectedMovie && (
  <div className="movie-details">
    <h2>{selectedMovie.Title}</h2>
    <p><strong>Jaar:</strong> {selectedMovie.Year}</p>
    <p><strong>Genre:</strong> {selectedMovie.Genre}</p>
```

```
    <p><strong>Plot:</strong> {selectedMovie.Plot}</p>
    <img src={selectedMovie.Poster} alt={selectedMovie.Title} />
    <button onClick={() => setSelectedMovie(null)}>Sluiten</button>
  </div>
)}
```

4. **Styling voor de details-popup:**
   - Voeg styling toe in een nieuw bestand `App.css` of in-line:

```
.movie-details {
  position: fixed;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  background-color: white;
  padding: 20px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
  z-index: 1000;
}

.movie-details button {
  margin-top: 10px;
  padding: 10px;
  background-color: #007BFF;
  color: white;
  border: none;
  cursor: pointer;
}

.movie-details button:hover {
  background-color: #0056b3;
}
```

Stap 2: Responsief ontwerp toevoegen

1. **Voeg algemene styling toe:**
   - Zorg ervoor dat de pagina een nette indeling heeft:

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  text-align: center;
}

ul {
  display: flex;
  flex-wrap: wrap;
  list-style-type: none;
  padding: 0;
  margin: 0;
}

li {
  flex: 1 1 calc(33.333% - 20px);
  margin: 10px;
  box-sizing: border-box;
}

@media (max-width: 768px) {
  li {
    flex: 1 1 calc(50% - 20px);
  }
}

@media (max-width: 480px) {
  li {
    flex: 1 1 100%;
  }
}
```

2. **Voeg mobiele ondersteuning toe:**
   - Gebruik media queries om de app goed te laten werken op mobiele apparaten.
   - Zorg ervoor dat de film-details zich aanpassen aan kleinere schermen:

```
.movie-details {
  width: 90%;
  max-width: 500px;
}
```

Stap 3: Test de applicatie

1. Start de ontwikkelserver:

   ```
   npm run dev
   ```

2. Zoek naar een film, klik op "Bekijk details" en controleer of de details worden weergegeven.
3. Test de app op verschillende schermformaten (desktop, tablet, mobiel).

Extra uitdaging (optioneel)

- Voeg een "Loading..."-indicator toe tijdens het ophalen van de filmgegevens.
- Voeg animaties toe aan de popup voor een betere gebruikerservaring.
- Laat gebruikers filteren op genre of jaar.

Samenvatting van de les

- Studenten hebben geleerd hoe ze een gedetailleerde weergave van films kunnen implementeren.
- Ze hebben een responsief ontwerp toegevoegd om de app gebruiksvriendelijk te maken op verschillende apparaten.

Voorbereiding voor toekomstige lessen

- Denk na over extra functionaliteiten, zoals filteren of paginering.

- Verken hoe je React Router kunt gebruiken om echte pagina's te maken in plaats van popups.

Revision #8
Created 2025-01-10 12:34:24 UTC by yildiz
Updated 2025-01-13 17:03:47 UTC by yildiz