Frond End 2

- React 2 (React Movie Explorer - concept 13/1)
- React 2 (React To-Do Master - concept 13/1)
- React 2

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.

React 2 (React To-Do Master - concept 13/1)

ChatGPT: Hier is een ander leuk projectidee dat geschikt is voor beginnende studenten om Vite + React te leren:

Projectnaam: "React To-Do Master"

Beschrijving: Studenten maken een eenvoudige maar functionele to-do lijstapplicatie waarin gebruikers taken kunnen toevoegen, bewerken, verwijderen en als voltooid kunnen markeren. Dit project introduceert de basisprincipes van React en laat studenten werken met state, props en eenvoudige interacties.

Functionaliteiten:

1. **Taken toevoegen:**
   - Gebruikers kunnen een nieuwe taak toevoegen via een invoerveld.
   - Elke taak heeft een titel en een optionele beschrijving.
2. **Taken weergeven:**
   - Een lijst van taken wordt weergegeven.
   - Taken die voltooid zijn, worden visueel anders getoond (bijvoorbeeld door een streep of grijze kleur).
3. **Taken beheren:**
   - Gebruikers kunnen een taak als voltooid markeren.
   - Gebruikers kunnen taken verwijderen.
   - Optioneel: gebruikers kunnen de taaknaam en beschrijving bewerken.
4. **LocalStorage-integratie:**
   - De takenlijst wordt opgeslagen in `localStorage`, zodat taken blijven bestaan na een herstart van de applicatie.
5. **Responsief design:**
   - De app werkt zowel op desktop als mobiel.

Extra Opties voor Gevorderde Studenten:

- **Categorisatie:** Voeg categorieën toe (bijvoorbeeld "School", "Persoonlijk", "Werk").
- **Filteren:** Voeg filters toe om alleen voltooide, onvoltooide of alle taken te zien.
- **Deadline-functionaliteit:** Voeg een deadline-optie toe en markeer taken die bijna verlopen.
- **Drag-and-Drop:** Laat gebruikers taken herschikken.

Benodigde tools en kennis:

1. **Tools:**
   - Vite: Voor het opzetten van het project.
   - React: Voor het bouwen van componenten.
   - CSS of een framework zoals Tailwind voor styling.
2. **Kennis:**
   - Basis JavaScript (ES6+) en React-beginselen (`useState`, props).
   - HTML/CSS voor eenvoudige opmaak.

Projectopbouw:

1. **Les 1:** Introductie en opzet
   - Installeer Vite en maak een nieuw project.
   - Begrijp de folderstructuur en bouw een "Hello World" component.
   - Maak een basisonderdeel voor de to-do lijst (bijvoorbeeld een leeg invoerveld en een knop).
2. **Les 2:** Taken toevoegen en weergeven
   - Gebruik `useState` om een takenlijst in de state op te slaan.
   - Bouw een eenvoudige lijstweergave waarin ingevoerde taken worden getoond.
3. **Les 3:** Taken beheren
   - Voeg functionaliteit toe om taken te verwijderen en als voltooid te markeren.

- Gebruik conditionele rendering om voltooide taken visueel anders weer te geven.

4. **Les 4:** Styling en afronding
   - Voeg eenvoudige CSS-styling toe om de app aantrekkelijk te maken.
   - Integreer `localStorage` zodat taken behouden blijven na een herstart.

Waarom dit project?

- Het project is eenvoudig en overzichtelijk.
- Studenten leren direct werken met state en props, twee fundamentele concepten in React.
- Het kan stapsgewijs worden uitgebreid voor gevorderde functionaliteiten.
- Het resultaat is praktisch en bevredigend om te gebruiken.

Laat me weten of je hulp nodig hebt met het uitschrijven van een lesplan of codevoorbeelden! □□

React 2

01 - React

React is een tool die je helpt om websites te bouwen die snel en interactief zijn.
Stel je voor dat je een website maakt zoals Instagram of YouTube, waar je op knoppen kunt klikken, reacties kunt plaatsen, of video's kunt bekijken zonder dat de hele pagina opnieuw geladen hoeft te worden.
React zorgt ervoor dat dit soepel en makkelijk werkt.

In plaats van dat je alles steeds opnieuw bouwt, kun je met React kleine bouwstenen maken (we noemen die componenten). Bijvoorbeeld: een knop, een zoekbalk of een bericht. Je kunt die bouwstenen hergebruiken, net als LEGO-blokjes, om snel een complete website te maken.

React is dus als een gereedschap voor moderne webbouwers om gave, interactieve websites te maken!

Opdracht

> Ga naar linkedin.com/learning React Essential Training via SPL
> >> Neem door: hoofdstuk 1 en 2

Inleveren:

Beantwoord de volgende vragen in eigen woorden:
- Wat is React?
- Wat is JSX?
- Noem 5 bedrijven die gebruik maken van React?

1. Noteer je vragen met antwoorden in de tekstvenster en
   lever een schreenshot van de Training waarbij vinkjes staan bij hoofdstuk 1 en 2.

afbeelding.png

---

02 - Versie beheer

Wat is versiebeheer met GitHub?
Stel je voor dat je aan een schoolproject werkt, bijvoorbeeld een presentatie of een website, en je wilt elke stap opslaan.
Zo kun je altijd teruggaan naar een eerdere versie als iets fout gaat.
Dat is wat **versiebeheer** doet: het bewaart alle veranderingen die je maakt.

GitHub is een soort online kluis voor je projecten.
Je kunt er jouw werk opslaan, delen met anderen en samenwerken zonder dat je bestanden kwijtraakt of per ongeluk iets overschrijft. Het werkt samen met een programma genaamd **Git**, dat al je wijzigingen netjes bijhoudt.

> Als je nog geen account hebt op GitHub maak dan een account op GitHub
> Ga naar github.com en maak een gratis account aan. Installeer Git op je computer en koppel het aan VS code, zodat je kan werken met versiebeheer en het kunnen samenwerken. Dat ga je later met jouw team doen.

Opdracht

Zorg ervoor dat git en github werkt.
Bij deze les is geen instructie je gaat zelf op onderzoek hoe het moet.
“Bijvoorbeeld via GitHub-documentatie, de instructies van VS Code of op https://www.linkedin.com/learning/learning-github”

Inleveren:

1. Lever één screenshot in dat je laat zien dat je github werkt.
2. Link naar je github account.

---

03 - Nieuw project met React Vite

Stap voor stap gaan wij een Boter Kaas en Eieren (Tic-Tac-Toe) spel maken in React
Onderstaande handleiding is bedoeld voor studenten om eenvoudig een Boter Kaas en Eieren-spel te maken met React.
We gebruiken Vite als bundler en ontwikkelserver omdat die lekker snel werkt.

> ### Handleiding Tic Tac Toe

Stap 0: Benodigdheden

- Node.js (12+ of hoger) Controleer met **node -v** en **npm -v** in de terminal of je Node en npm hebt.
- Basiskennis van React: kennis van componenten, props en state.

Stap 1: Nieuw React-project opzetten met Vite

```
npm create vite@latest tictactoe -- --template react
cd tictactoe
npm install
npm run dev
```

Stap 2: Projectstructuur begrijpen
Na het creëren van het project heb je ongeveer deze mappen en bestanden:

- `main.jsx`: De React-entree van je app. Hier wordt de `<App/>`-component ingeladen.
- `App.jsx`: Dit is je hoofdbestand voor je applicatie-logica en (in dit geval) voor ons spel.
- `App.css`: Hier komt (of staat) de styling voor je app.

```
tictactoe/
├─ index.html
├─ package.json
├─ vite.config.js
└─ src/
   ├─ App.css
   ├─ App.jsx
   ├─ main.jsx
   └─ ...
```

Opdracht

Voer de bovenstaande stappen uit en maak via github een Commit zodat je weer terug kan als je iets verkeert gaat.
Ruim onnodige bestanden op.

Inleveren:
1. Screenshot van je project
2. Link naar je github account waar het project staat.

04 - Tic Tac Toe

Nu je React project hebt geinstalleerd ga je verder

Stap 1: Styling toevoegen (App.css)

Open src/App.css in je code-editor. Vervang (of voeg toe) onderstaande CSS:
Dit zorgt voor een simpel maar overzichtelijk design van het bord en de knoppen.

```
.app {
  text-align: center;
  font-family: Arial, sans-serif;
}
.board {
  display: grid;
  grid-template-columns: repeat(3, 100px);
  grid-gap: 5px;
  margin: 20px auto;
}
.square {
  width: 100px;
  height: 100px;
  font-size: 24px;
  font-weight: bold;
  background-color: #f0f0f0;
  border: 2px solid #333;
  cursor: pointer;
  display: flex;
  justify-content: center;
  align-items: center;
}
.square:hover {
  background-color: #ddd;
}
.info {
  margin-top: 20px;
  font-size: 18px;
}
.reset-button {
```

```
  margin-top: 20px;
  padding: 10px 20px;
  font-size: 16px;
  background-color: #007BFF;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}
.reset-button:hover {
  background-color: #0056b3;
}
```

Stap 2: De React-code (App.jsx)

- Open `src/App.jsx`. Verwijder de bestaande inhoud en plak de onderstaande code in:

```
import React, { useState } from 'react';
import './App.css';

function App() {
  // 1. State voor het bord: een array van 9 posities (3x3)
  const [board, setBoard] = useState(Array(9).fill(null));

  // 2. State voor wie er aan de beurt is: X begint
  const [isXNext, setIsXNext] = useState(true);

  // 3. Check of er een winnaar is:
  const winner = calculateWinner(board);

  // Functie die afhandelt wat gebeurt bij klik
  const handleClick = (index) => {
    // Als het vakje al gebruikt is of er al een winnaar is, niets doen
    if (board[index] || winner) return;

    // Kopieer het bord en zet X of O in de aangeklikte plek
    const newBoard = [...board];
    newBoard[index] = isXNext ? 'X' : 'O';
    setBoard(newBoard);
```

```
  // Wissel van speler
  setIsXNext(!isXNext);
};

// Functie om het spel opnieuw te starten
const resetGame = () => {
  setBoard(Array(9).fill(null));
  setIsXNext(true);
};

return (
  <div className="app">
    <h1>Boter Kaas en Eieren</h1>

    {/* Bord-weergave */}
    <div className="board">
      {board.map((value, index) => (
        <Square
          key={index}
          value={value}
          onClick={() => handleClick(index)}
        />
      ))}
    </div>

    {/* Info: Winnaar, gelijkspel of volgende speler */}
    <div className="info">
      {winner ? (
        <h2>Winnaar: {winner}</h2>
      ) : board.every((square) => square) ? (
        <h2>Gelijkspel!</h2>
      ) : (
        <h2>Volgende speler: {isXNext ? 'X' : 'O'}</h2>
      )}
    </div>

    {/* Knop om opnieuw te starten */}
    <button onClick={resetGame} className="reset-button">
      Opnieuw Spelen
    </button>
```

```
    </div>
  );
}

// Klein componentje voor één vakje van het bord
function Square({ value, onClick }) {
  return (
    <button className="square" onClick={onClick}>
      {value}
    </button>
  );
}

// Functie die nagaat of iemand gewonnen heeft
function calculateWinner(board) {
  const lines = [
    [0, 1, 2], [3, 4, 5], [6, 7, 8], // Horizontale rijen
    [0, 3, 6], [1, 4, 7], [2, 5, 8], // Verticale kolommen
    [0, 4, 8], [2, 4, 6], // Diagonalen
  ];
  for (let [a, b, c] of lines) {
    // board[a], board[b], board[c] zijn gelijk en niet null -> winnaar
    if (board[a] && board[a] === board[b] && board[a] === board[c]) {
      return board[a]; // 'X' of 'O'
    }
  }
  return null; // Geen winnaar
}

export default App;
```

Stap 3: Applicatie starten en testen

Zorg dat je ontwikkelserver draait:

```
npm run dev
```

Open (of herlaad) in je browser de link die in de terminal staat, meestal http://127.0.0.1:5173/.
Je ziet nu een 3x3-bord. Klik op een vakje om een X te plaatsen, klik vervolgens op een ander vakje voor O, enzovoorts.

Je ziet nu een 3x3-bord. Klik op een vakje om een X te plaatsen, klik vervolgens op een ander vakje voor O, enzovoorts.

Werkt alles?

1. Zie je “Volgende speler: X” of “Volgende speler: O”?
2. Komt er “Winnaar: X (of O)” in beeld als er 3 op een rij is?
3. Krijg je “Gelijkspel” als alle vakjes vol zijn maar niemand heeft gewonnen?
4. Werkt de knop “Opnieuw Spelen”?

> Als het werkt, heb je een volledig React boter kaas en eieren spel gebouwd!

Opdracht

Voer de bovenstaande stappen uit en zorg ervoor dat je een versie hebt opgeslagen github.

Inleveren:

1. link naar je project op github
2. screencast (filmpje) van je werkende spel. (let op: max 5mb)

---

05 - Samenwerken in een team.

Je hebt een basisversie van Boter Kaas en Eieren (Tic-Tac-Toe) in React gebouwd. Deze werkt, maar kan nog veel spannender worden! In deze opdracht gaan jullie in teams samenwerken om het spel uit te breiden met nieuwe functionaliteiten en styling.

Doel

1. Samenwerken in een groep van 3 of 4 studenten.
2. Werken met **Git/GitHub** om code samen te voegen.
3. Leren hoe je React-uitbreidingen maakt: scorebord, AI, highlight van de winnende rij, of een ander idee.
4. Presenteren van jullie eindresultaat aan de klas/docent.

Te ondernemen stappen

1. **Vorm een team**
   - Vraag aan de docent met wie je samenwerkt.
   - Maak onderling afspraken over communicatie en taakverdeling.
2. **Verdeel de uitbreidingen**
   Ieder kiest een uitbreiding of verzint zelf iets. Bijvoorbeeld:
   1. **Scorebord:** Houd bij hoe vaak X en O winnen, en toon dit in de UI.
   2. **Winnende rij highlighten:** Licht de winnende vakjes extra op.
   3. **AI-tegenstander:** Laat een eenvoudige computer ook meespelen.
   4. **App versie** voor mobile telefoon
   5. **Eigen idee:** Bedenk zelf een toffe functie (zoals een timer, statistieken, grotere spelborden, etc.).
3. **Afspraken maken**
   - **Wie doet wat?** Noteer dit kort.
   - **Hoe delen jullie de code?** Gebruik bijv. Git en maak voor elk onderdeel een eigen branch.
4. **Implementeren**
   - Pas de React-code aan om jouw gekozen feature te bouwen.
   - Werk tegelijkertijd aan styling en layout (CSS).
   - Test alles grondig in je groep (check of alle functies goed samenwerken).
5. **Samenvoegen en testen**
   - Maak Pull Requests (PR’s) of merge van ieders branch naar de hoofdbranch.
   - Test als team het geheel: Werkt het scorebord? Komt de highlight goed in beeld? Doet de AI een zet?
6. **Eindpresentatie**
   - Demonstreer jullie spel. Speel een potje, laat zien hoe je feature(s) werken.
   - Licht kort jullie rol en uitdagingen toe.

Opdracht

Voer bovenstaande stappen uit

Inleveren:

1. presentatie. (na presentatie is deze opdracht voldaan)

---

06 - Ieder een eigen game

Met je team werk je samen aan één nieuw spel, gebouwd in React (met Vite).
Samen ga je onderzoeken wat de mogelijkheden zijn met React.

Welk spel ga je bouwen?

- Dat mag een **bordspel**, **kaartspel**, **quiz**, **platformer** of ander genre zijn.
- Enkele voorbeelden ter inspiratie:
  - **Memory** (kaarten omdraaien en matchen)
  - **Ganzebord** (pionnen, dobbelsteen, speciale vakken)
  - **Pictionary**-achtige app (tekenen, raden)
  - **Quiz-/Trivia-spel** met meerkeuzevragen
  - Of een **eigen** origineel concept!

2. **Spelregels en Doel**
   - Beschrijf kort en bondig de regels en het doel van je spel.
   - Wat maakt het spel “leuk” of uitdagend voor de speler?
3. **Layout en Styling**
   - Maak een (schets)ontwerp van de interface.
   - Werk met componenten (bijv. `Board`, `Card`, `Player`, `Question`, `Controls`, etc.).
   - Zorg voor een herkenbare en aantrekkelijke styling (CSS).
4. **Functionaliteit**
   - Minimaal:
     - **State management** met React Hooks (bijv. `useState`, `useEffect`).
     - **Interactie**: De gebruiker kan klikken, invoeren of anderszins interacteren.
     - **Logica**: Het spel reageert op de acties volgens de afgesproken spelregels.
   - Optioneel (maar aanbevolen):
     - **Scoren** of bijhouden van voortgang.

- **Speciaal effect** of **animaties** bij winst/verlies of bij interactie.
- **LocalStorage** of **session** om (een deel van) de voortgang op te slaan.

---

2. Samenwerkingsafspraken

1. **Teamrollen**
   - Verdeel onderling wie het game concept uitwerkt, wie de frontend vormgeeft, wie de logica programmeert, enzovoort.
   - Minimaal één keer per week overleg (live of online).
2. **Versiebeheer**
   - Gebruik een Git-repository (GitHub, GitLab, Bitbucket).
   - Maak voor elk onderdeel (feature) een aparte branch; doe Pull Requests en code reviews.
3. **Planning**
   - Stel een globale tijdslijn op: ontwerp → implementatie basis → testen → uitbreiden → presentatie.
   - Zorg dat iedereen tussentijds kan laten zien hoe ver hij/zij is.

Inleveren

1. **Werkende React-app**
   - Gemaakt met Vite.
   - Startbaar via `npm install` en `npm run dev`
2. **Git-repository**
   - Gedeelde repository met duidelijke commit-geschiedenis van alle teamleden.
3. **Documentatie**
   - **Korte handleiding**: hoe werkt het spel? Wat zijn de regels?
   - **Korte technische uitleg**: hoe hebben jullie de logica opgezet? Welke componenten zijn er?
   - **Afgesproken taken**: wie heeft wat gebouwd?
   - **Problemen en oplossingen**: noem eventuele uitdagingen en hoe jullie die oplosten.

4. **Presentatie** (als onderdeel van de les)
    - Demo van het spel (kort potje spelen, of laten zien hoe het werkt).
    - Toelichting op jullie samenwerking (verdeling, processen).
    - Vragen van de docent/medestudenten beantwoorden.