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
image not found or type unknown

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.

Revision #19
Created 19 January 2025 12:51:40 by yildiz
Updated 7 February 2025 15:48:50 by yildiz