Women Go Tech
Introduction to React

Maurice de Beijer - @mauricedb

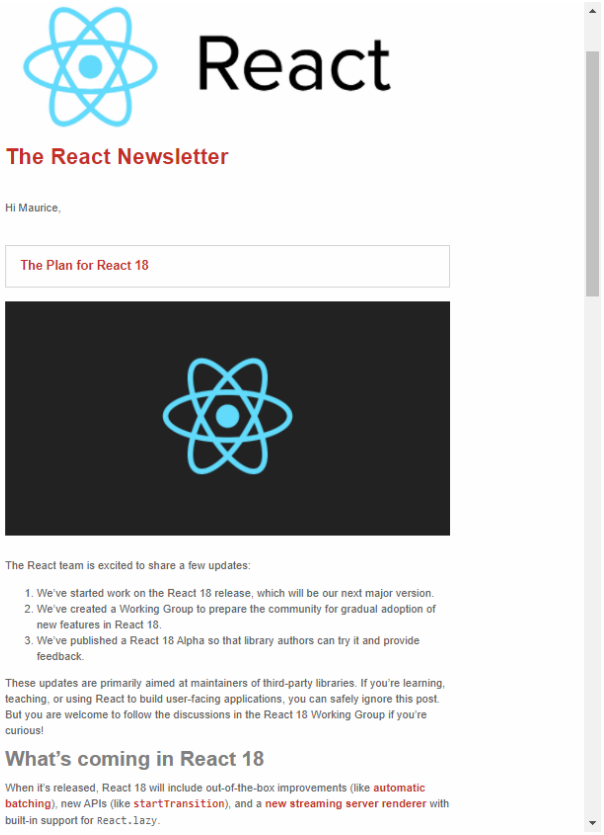
Workshop goal

Workshop goal

- Learn about:
  - What is React?
  - Create React App
  - React Components and JSX
  - Using component Props and State
  - Styling a React component
  - Side Effects and AJAX requests
  - Conditional rendering of components

- Maurice de Beijer
- The Problem Solver
- Microsoft MVP
- Freelance lead/developer/instructor
- Twitter: @mauricedb
- Web: http://www.TheProblemSolver.nl
- E-mail: maurice.de.beijer@gmail.com

The React Newsletter



React

The React Newsletter

Hi Maurice,

The Plan for React 18

The React team is excited to share a few updates:

1. We've started work on the React 18 release, which will be our next major version.
2. We've created a Working Group to prepare the community for gradual adoption of new features in React 18.
3. We've published a React 18 Alpha so that library authors can try it and provide feedback.

These updates are primarily aimed at maintainers of third-party libraries. If you're learning, teaching, or using React to build user-facing applications, you can safely ignore this post. But you are welcome to follow the discussions in the React 18 Working Group if you're curious!

What's coming in React 18

When it's released, React 18 will include out-of-the-box improvements (like **automatic batching**), new APIs (like `startTransition`), and a **new streaming server renderer** with built-in support for `React.lazy`.

Type it out by hand?

"Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now!"

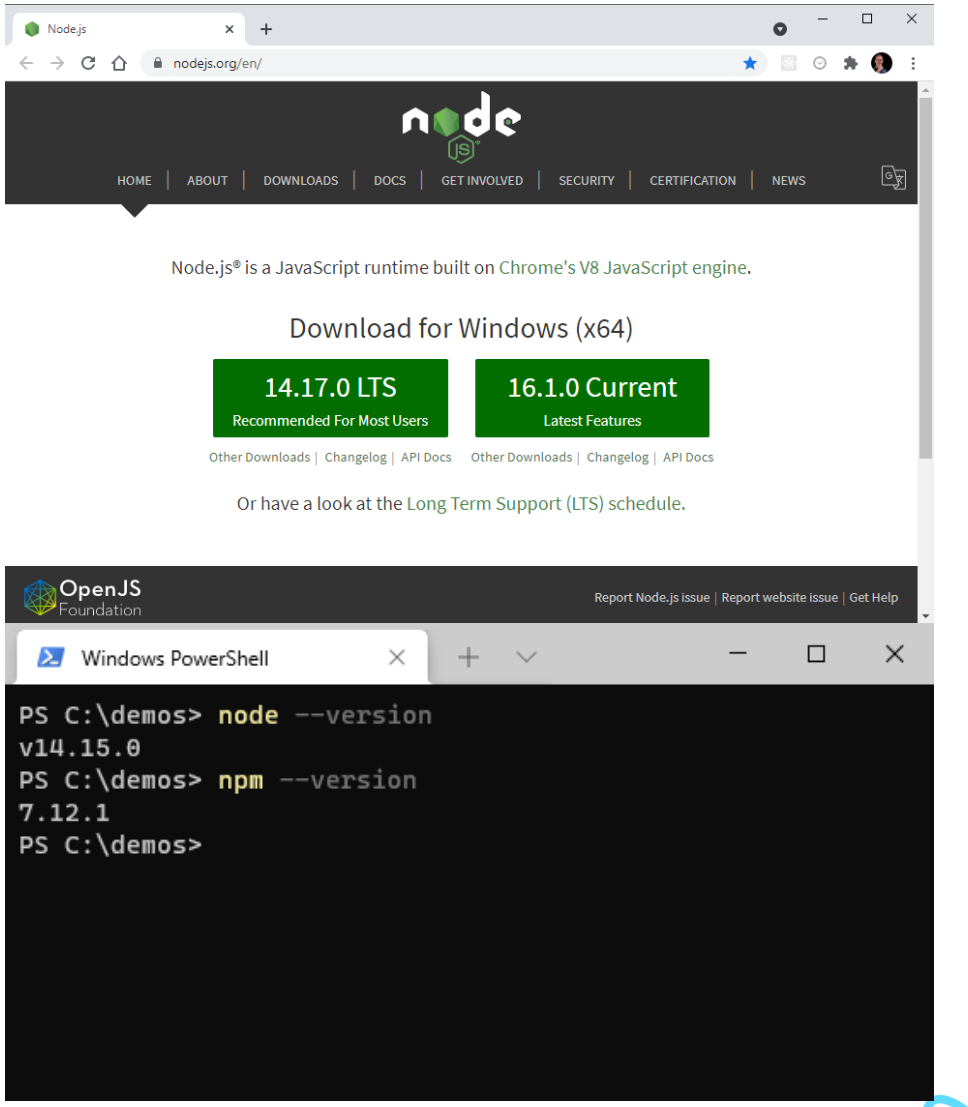
Prerequisites

Install Node & NPM

Install the GitHub repository

Install Node.js & NPM

- Minimal:
  - Node version 10
  - NPM version 6



Following Along

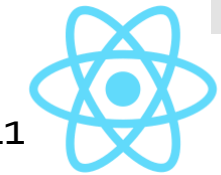

```
import "./App.css";

import { Movie } from "./components/Movie";

function App() {
  return (
    <div className="App">
      <Movie />
    </div>
  );
}
```

- Repository: https://github.com/mauricedb/women-go-tech-2021/
- Slides: http://theproblemsolver.nl/women-go-tech-2021.pdf

What is React?

What is React?

- A JavaScript library for building **user interfaces**
- Build at **Facebook** and Instagram
  - Open source and maintained on GitHub
  - React is MIT licensed
- Work against a **virtual DOM**
  - No differences between browsers
  - Can also be rendered on the server when needed
- Uses a **unidirectional data flow**
  - No two way data binding



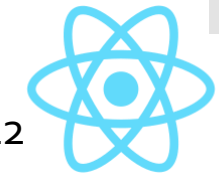
React Trade-offs

Advantages

- Large community
- Small runtime library
- Fast
- Stable
- Simple and predictable API
- Supports server side rendering
- Open source
- Dedicated team from Facebook

Disadvantages

- JavaScript focused
- Markup inside JavaScript
- Increases memory pressure
- Driven by Facebooks needs
- Not very opinionated

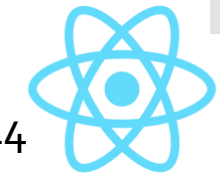


Create React App

Create a React application with no build configuration

Create React App

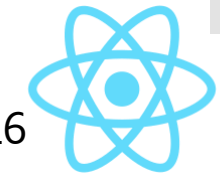
- Create React apps with **no build configuration**
- The **react-scripts** package does most of the work
  - Uses Babel, Webpack and ESLint under the hood.
- npx create-react-app my-app
- 👉 Requires Node 10 or later



Getting started


```
// To create a new application
npx create-react-app movies-app
cd ./movies-app
npm start
Open browser at http://localhost:3000
```

What is JSX?



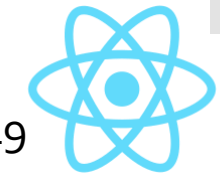
What is JSX?

- JSX is an **XHTML syntax** used by React to define components
  - Extends JavaScript
- JSX needs to be **transpiled** to standard JavaScript
  - Using Babel

What is JSX?

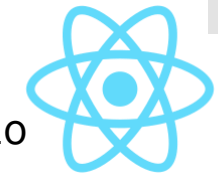
```jsx
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>Edit <code>src/App.js</code> and save to reload.</p>
        <a
          className="App-link" href="https://reactjs.org"
          target="_blank" rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

React Components

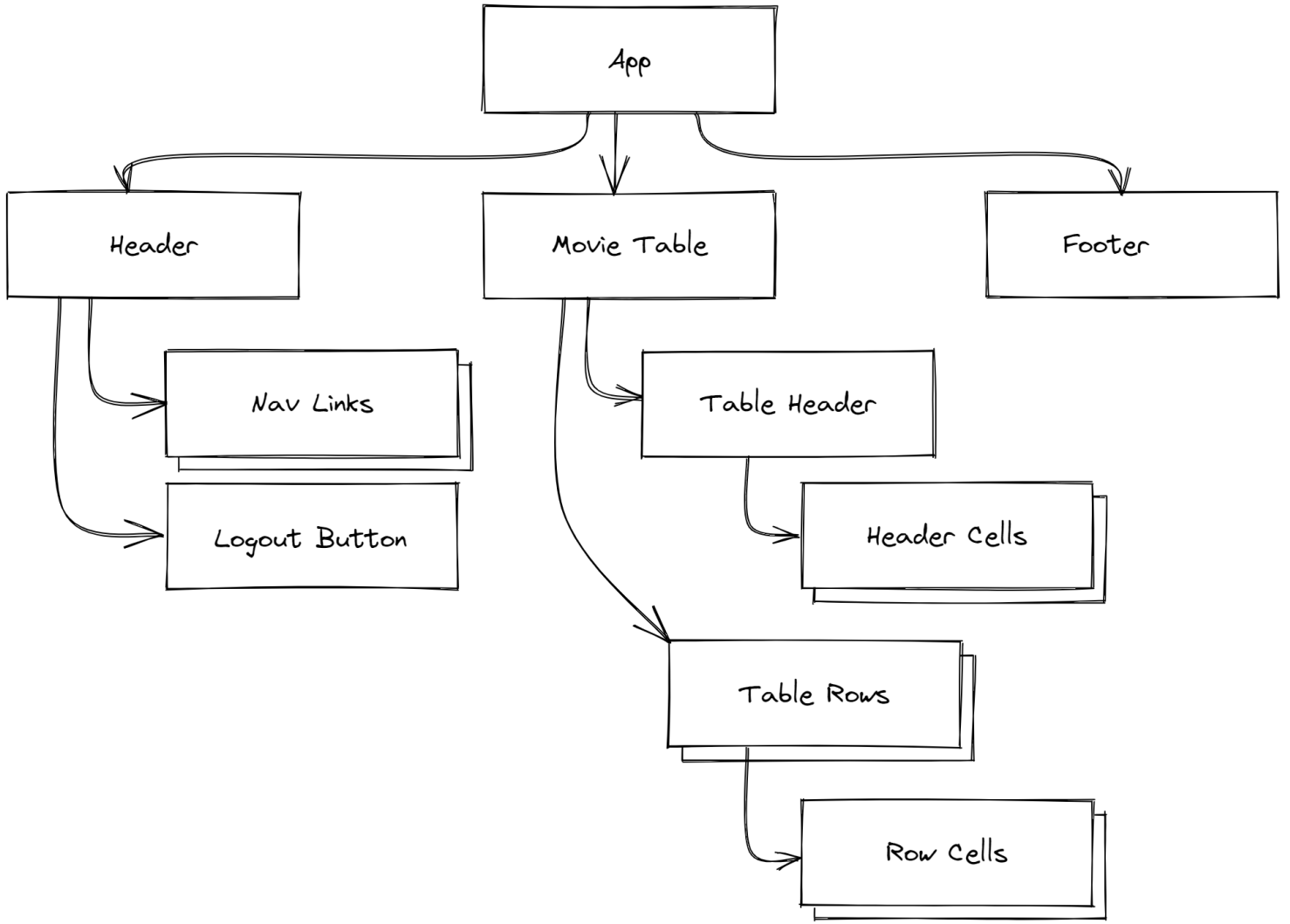


React Components

- A React application consists of one or more **components**
- Components are the **view** with embedded view logic
  - Results in HTML elements with associated behavior
- Each component can contain other **nested components** if needed
  - Split large components into smaller subcomponents
  - Use the single responsibility pattern

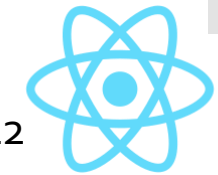


Component Tree



React Components

- Components are usually **written as functions**
  - There is also an older class based syntax



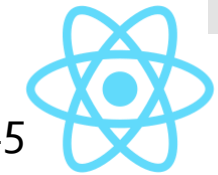
Movie.js

```
const movie19404 = {
  id: 19404,
  release_date: "1995-10-20",
  title: "Dilwale Dulhania Le Jayenge",
  vote_average: 8.7,
};

export function Movie() {
  return (
    <form>
      <div>
        <label>Title</label>
        <p>{movie19404.title} </p>
      </div>

      <div>
        <label>Vote average</label>
        <p>{movie19404.vote_average} </p>
      </div>

      <div>
        <label>Release date</label>
        <p>{movie19404.release_date} </p>
      </div>
    </form>
  );
}
```

App.js

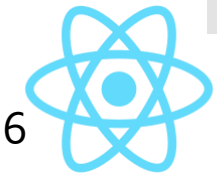

```
import "./App.css";

import { Movie } from "./components/Movie";

function App() {
  return (
    <div className="App">
      <Movie />
    </div>
  );
}
```

Component Props



Component Props

- **Props** are used to pass arguments from parent to child components
  - Both regular data like string, number or objects
  - As well as callback functions
- Props should be considered **immutable**
  - You should never assign a value to them
  - Even if they are objects or arrays
- Some prop names are **special names** because of JavaScript keywords
  - An HTML `class` attribute is passed as `className`
  - An HTML `for` attribute is passed as `htmlFor`

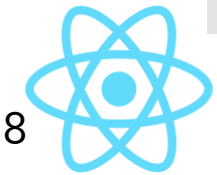


Props & Data types

- **String props** can be passed using " "
- **Other types** need to be passed as an expression using {}
- Just specifying the prop name passes *true*

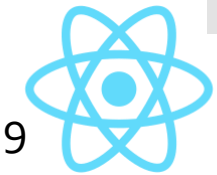
The key prop

- Required when creating **arrays of child components**
  - The values should be unique
- Key properties should have **stable values**
  - A chaining key will unmount and remount a component subtree



Movie.js

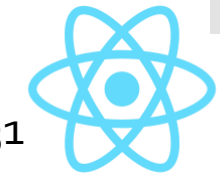
```jsx
export function Movie({ movie }) {
  return (
    <form>
      <div>
        <label>Title</label>
        <p>{movie.title} </p>
      </div>

      <div>
        <label>Vote average</label>
        <p>{movie.vote_average} </p>
      </div>

      <div>
        <label>Release date</label>
        <p>{movie.release_date} </p>
      </div>
    </form>
  );
```

App.js

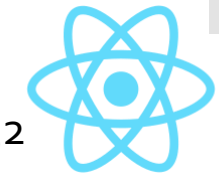

```
import './App.css';

import { Movie } from "./components/Movie";

const movie19404 = {
  id: 19404,
  release_date: "1995-10-20",
  title: "Dilwale Dulhania Le Jayenge",
  vote_average: 8.7,
};

function App() {
  return (
    <div className="App">
      <Movie movie={movie19404} />
    </div>
  );
}
```

State Management



State Management

- State is **data** in a component that can change over it's lifetime
  - Owned by the components
  - Just like local variables in a function
- Functional components can use the ***useState()* hook**
  - Returns a tuple with the current state and an update function
- Updating a component state will **re-render** that component
  - Also re-renders child components
- **Never mutate** state directly
  - Always use the update function
- Use **multiple *useState()* hooks** if needed



Movie.js


```jsx
import { useState } from "react";

export function Movie({ movie }) {
  const [thisMovie, setThisMovie] = useState(movie);

  return (
    <form>
      <div>
        <label>Title</label>
        <input
          value={thisMovie.title}
          onChange={(e) => setThisMovie({ ...thisMovie, title: e.target.value })}
        />
      </div>

      <div>
        <label>Vote average</label>
        <input
          value={thisMovie.vote_average}
          onChange={(e) => setThisMovie({ ...thisMovie, vote_average: e.target.value })}
        />
      </div>

      <div>
        <label>Release date</label>
        <input
          value={thisMovie.release_date}
          onChange={(e) => setThisMovie({ ...thisMovie, release_date: e.target.value })}
        />
      </div>
    </form>
  );
```

Submitting the movie form

Submitting the movie form

- Add a **submit button** to the data entry form
- Handle the <form> **onSubmit event**
  - Make sure to prevent the default browser action

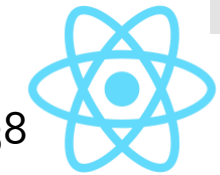
Movie.js


```jsx
import { useState } from "react";

export function Movie({ movie }) {
  const [thisMovie, setThisMovie] = useState(movie);

  return (
    <form onSubmit={(e) => {
        e.preventDefault();
        alert(JSON.stringify(thisMovie, null, 2));
      }}>
      <div>
        <label>Title</label>
        <input
          value={thisMovie.title}
          onChange={(e) => setThisMovie({ ...thisMovie, title: e.target.value })}
        />
      </div>

      <div>
        <label>Vote average</label>
        <input
          value={thisMovie.vote_average}
          onChange={(e) => setThisMovie({ ...thisMovie, vote_average: e.target.value })}
        />
      </div>

      <div>
        <label>Release date</label>
        <input
          value={thisMovie.release_date}
          onChange={(e) => setThisMovie({ ...thisMovie, release_date: e.target.value })}
        />
      </div>
      <div>
        <button>Submit</button>
      </div>
    </form>
  );
```

Styling with Bootstrap

Styling with Bootstrap

- Use NPM to **install the bootstrap package**
  - Using: `npm install bootstrap`
  - And import the CSS file
- React **uses the *className*** prop
  - Instead of the *class* attribute
- **Inline styles** can be done with a *style* prop
  - This takes an object



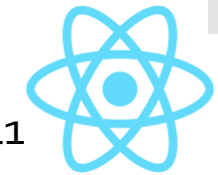
App.js

```
import "bootstrap/dist/css/bootstrap.css";
import "./App.css";

import { Movie } from "./components/Movie";

const movie19404 = {
  id: 19404,
  release_date: "1995-10-20",
  title: "Dilwale Dulhania Le Jayenge",
  vote_average: 8.7,
};

function App() {
  return (
    <div className="container">
      <Movie movie={movie19404} />
    </div>
  );
}
```

Movie.js

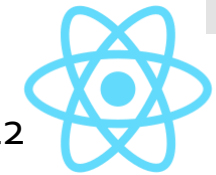

```jsx
export function Movie({ movie }) {
  const [thisMovie, setThisMovie] = useState(movie);

  return (
    <form
      onSubmit={(e) => {
        e.preventDefault();
        alert(JSON.stringify(thisMovie, null, 2));
      }}
    >
      <div className="mb-3">
        <label className="form-label">Title</label>
        <input
          className="form-control"
          value={thisMovie.title}
          onChange={(e) => setThisMovie({ ...thisMovie, title: e.target.value })}
        />
      </div>

      <div className="mb-3">
        <label>Vote average</label>
        <input
          className="form-control"
          value={thisMovie.vote_average}
          onChange={(e) => setThisMovie({ ...thisMovie, vote_average: e.target.value })}
        />
      </div>

      <div className="mb-3">
        <label>Release date</label>
        <input
          className="form-control"
          value={thisMovie.release_date}
          onChange={(e) => setThisMovie({ ...thisMovie, release_date: e.target.value })}
        />
      </div>
      <div className="mb-3">
        <button className="btn btn-primary">Submit</button>
      </div>
    </form>
  );
}
```

Doing an AJAX request with useEffect



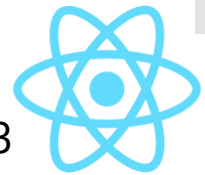
Side Effects

- Functional components use **hooks** for lifecycle management
  - The useEffect hook is the main hook used
- As the name suggests it is intended **for side effects**
- Optionally return a **cleanup function** from a useEffect hook
  - How often the effect hook is called depends on the second parameter with dependencies



Movies.js

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

export function Movies() {
  const [movies, setMovies] = useState([]);

  useEffect(() => {
    async function loadMovies() {
      const response = await fetch(
        "https://the-problem-solver-sample-data.azurewebsites.net/top-rated-movies"
      );

      if (response.ok) {
        const data = await response.json();
        setMovies(data);
      }
    }

    loadMovies();
  }, []);

  return (
    <div>
      <h1>Movies</h1>
    </div>
  );
}
```



App.js


```
import "bootstrap/dist/css/bootstrap.css";
import "./App.css";

import { Movies } from "./components/Movies";
import { Movie } from "./components/Movie";

function App() {
  return (
    <div className="container">
      <Movies />
    </div>
  );
}
```

HTML table with movies

HTML table

- Use the JavaScript **array.map()** function
  - Map an array of objects into an array of components
- Each mapped component needs a **key prop** to identify it
  - Normally mapped to some object property
  - The key is a special reserved prop

Movies.js


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

export function Movies() {
  // Code as before

  return (
    <div>
      <h1>Movies</h1>


      <table className="table table-striped table-bordered">
        <thead>
          <tr>
            <th>Title</th>
            <th>Edit</th>
          </tr>
        </thead>
        <tbody>
          {movies.map((movie) => (
            <tr key={movie.id}>
              <td>{movie.title}</td>
              <td>
                <button className="btn btn-primary" onClick={() => {}}>
                  Edit
                </button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
}
```

Navigating

From table to edit form

Navigating

- **Render different components** based on a stateful **condition**
  - Has a movie been selected or not?
- Pass **callback functions as props** to the child component
  - If needed to affect the stateful condition

App.js

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

import "bootstrap/dist/css/bootstrap.css";
import "./App.css";

import { Movies } from "./components/Movies";
import { Movie } from "./components/Movie";

function App() {
  const [currentMovie, setCurrentMovie] = useState(null);

  return (
    <div className="container">
      {currentMovie ? (
        <Movie
          movie={currentMovie}
          clearCurrentMovie={() => setCurrentMovie(null)}
        />
      ) : (
        <Movies setCurrentMovie={setCurrentMovie} />
      )}
    </div>
  );
}

export default App;
```

Movies.js

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

export function Movies({ setCurrentMovie }) {
  // Code as before

  return (
    <div>
      <h1>Movies</h1>


      <table className="table table-striped table-bordered">
        <thead>
          <tr><th>Title</th><th>Edit</th></tr>
        </thead>
        <tbody>
          {movies.map((movie) => (
            <tr key={movie.id}>
              <td>{movie.title}</td>
              <td>
                <button className="btn btn-primary" onClick={() => {
                    setCurrentMovie(movie);
                  }}>
                  Edit
                </button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
}
```

Movie.js


```jsx
export function Movie({ movie, clearCurrentMovie }) {
  const [thisMovie, setThisMovie] = useState(movie);

  return (
    <form
      onSubmit={(e) => {
        e.preventDefault();
        alert(JSON.stringify(thisMovie, null, 2));
        clearCurrentMovie();
      }}
    >
      { /* JSX as before */}

      <div className="mb-3">
        <button className="btn btn-primary">Submit</button>
      </div>
    </form>
  );
}
```

Conclusion

- React is a very capable and popular UI library for web applications
- Use Create React App to get started quickly
- Split you application into a tree of components
- Style components just like HTML tags
- Use read-only props to pass data to components
- Use the useState() hook for data that needs to be updated
- Use the useEffect() hook for side effect like AJAX requests

Maurice de Beijer

@mauricedb

maurice.de.beijer@gmail.com