Modul 223

Multiuser - Apps

Unterrichtsziel heute

- Erklären Architektur React-Frontend
- Implementieren einfaches React-Frontend mit REST-API Zugriff
- ~~Erklären Architektur Multiuser React-Frontend~~
- ~~Implementieren Multiuser - Frontend~~

Warmup Backend

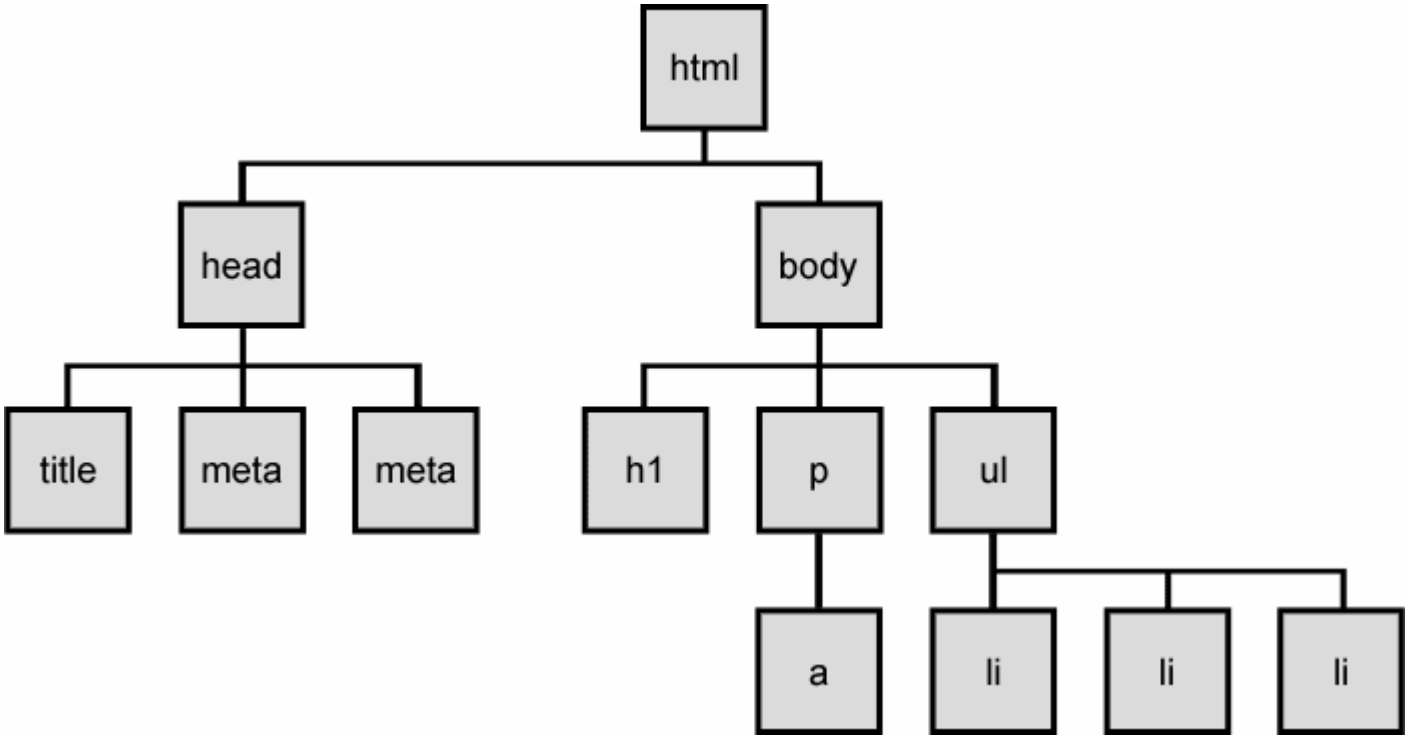
- In welcher Reihenfolge startet man die Komponenten Ihrer Webapp?
- Was hat die Salami-Technik mit dem Testen und der Fehlersuche in Web-Apps zu tun?
- Das backend startet nicht. Wie geht man bei der Fehlersuche vor?
- Das backend startet, aber antwortet nicht auf Requests. Wie geht man bei der Fehlersuche vor?
- Wie geht man beim Testen vor?

Warmup Frontend

- Wie legt man ein React-Frontend Projekt an?
- Wie startet man das React-Frontend?
- Wie werden GUI-Komponenten mit React erstellt?
- Wie wird im Frontend zwischen verschiedenen Views navigiert?
- Was hat es mit useState() und useEffect() auf sich?

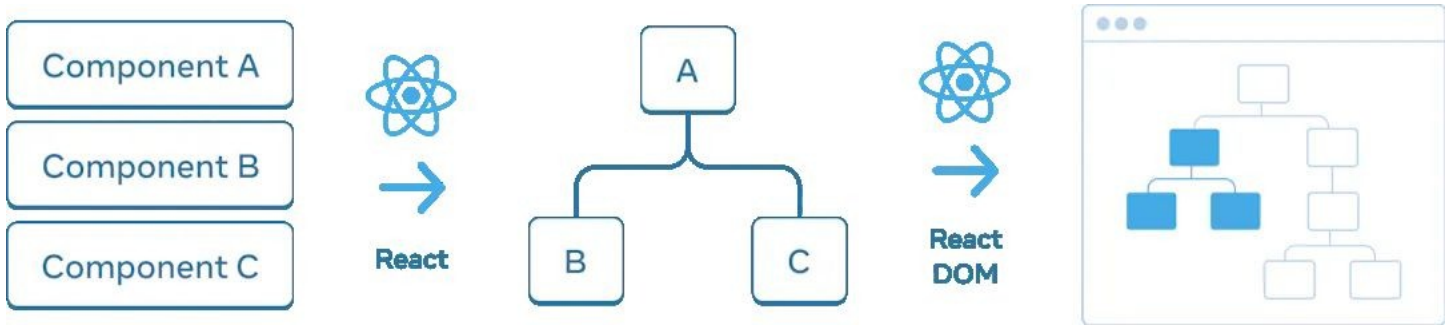
HTML DOM

- Was versteht man unter dem HTML – DOM?
- Welche Eigenschaften haben HTML-Elemente?



React & the DOM

https://react.dev/learn/understanding-your-ui-as-a-tree



App.js FancyText.js InspirationGenerator.js Copyright.js quotes.js

Reset Fork

```
1  import FancyText from './FancyText';
2  import InspirationGenerator from './InspirationGenerator
3  import Copyright from './Copyright';
4
5  export default function App() {
6    return (
7      <>
8        <FancyText title text="Get Inspired App" />
9        <InspirationGenerator>
10         <Copyright year={2004} />
11       </InspirationGenerator>
12     </>
13   );
14 }
15
16
```

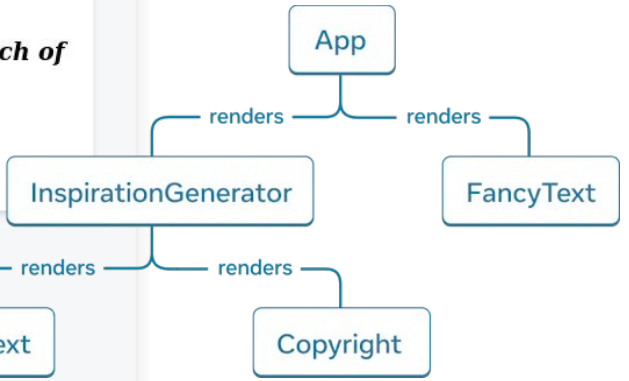
Get Inspired App

Your inspirational quote is:

Don't let yesterday take up too much of today." — Will Rogers

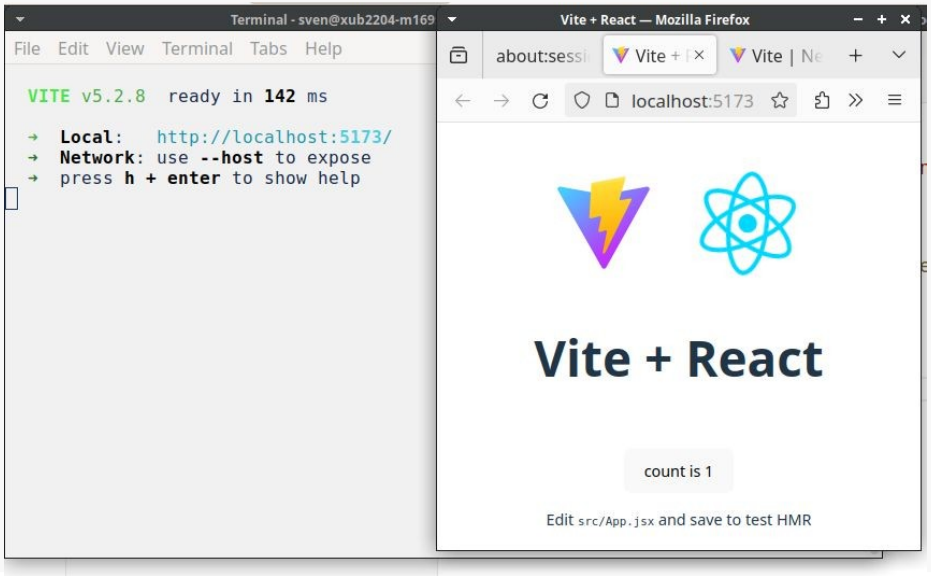
Inspire me again

© 2004



React: Get started

1) npm create vite@latest M223_Frontend_Basic -- --template react
2) oder besser gleich mit Routing: git clone https://github.com/greenorca/M223_Frontend_Basic.git
3) cd M223_Frontend_Basic
4) npm install
5) npm run dev



Think in React

https://react.dev/learn/thinking-in-react

Debugging mit Browser: npm install react-devtools

```jsx
# src/main.jsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import Person from './Person.jsx'
import './index.css'

const data = {
  name: "Max Muster",
  hobbies: ["surf", "eat", "sleep"]
}
const root = document.getElementById("root")
ReactDOM.createRoot(root).render(
  <React.StrictMode>
    <Person person={ data } />
    <App />
  </React.StrictMode>,
)
```

```jsx
# src/Person.jsx
const Liste = ({liste}) => {
  const items = liste.map(item => {
    return <li>{ item }</li>
  })
  return <ul>{ items }</ul>
}

const Person = ({person}) => {
 return <>
   <h1>{ person.name }</h1>
   <Liste liste={ person.hobbies } />
  </>;
};

export default Person;
```

Router mit Route Elementen (deklarativ)

https://reactrouter.com/en/main/start/tutorial#tutorial

- src/App.jsx

```jsx
function App() {
 return (
  <>
   <Routes>
    <Route path="/" element={<Layout />}>
     <Route index element={<Home />} />
     <Route path="new" element={<NewForm />} />
     <Route path="about" element={<About />} />
     <Route path="*" element={<NoPage />} />
    </Route>
   </Routes>
  </>
 )
}
```

- *npm install react-router-dom*
- *Route definiert Endpunkte* (URLs im Frontend) und dazugehörigen *Render-Componenten*

Router mit Route Elementen (deklarativ)

https://reactrouter.com/en/main/start/tutorial#tutorial

- src/modules/Layout.jsx

```
import { Outlet } from "react-router-dom"
import Navigation from "./Navigation"

export default function Layout(){
  return(
    <div>
      <Navigation />
      <Outlet />
    </div>
  )
}
```

- Wurzel-Element innerhalb der definierten <Route>
- enthält Navigation und Outlet zum Anzeigen der verschachtelten Routes

Fetching Data using axios

and React hooks (useState, useEffect)
(zum Installieren des packages: *npm install axios*)

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

const baseURL = "http://localhost:8080"

const Public = () => {
  const [items, setItems] = useState(null);

  useEffect(() => {
    axios.get(baseURL+"/items")
        .then((response) => {
            setItems(response.data);
         });
  }, []);
```

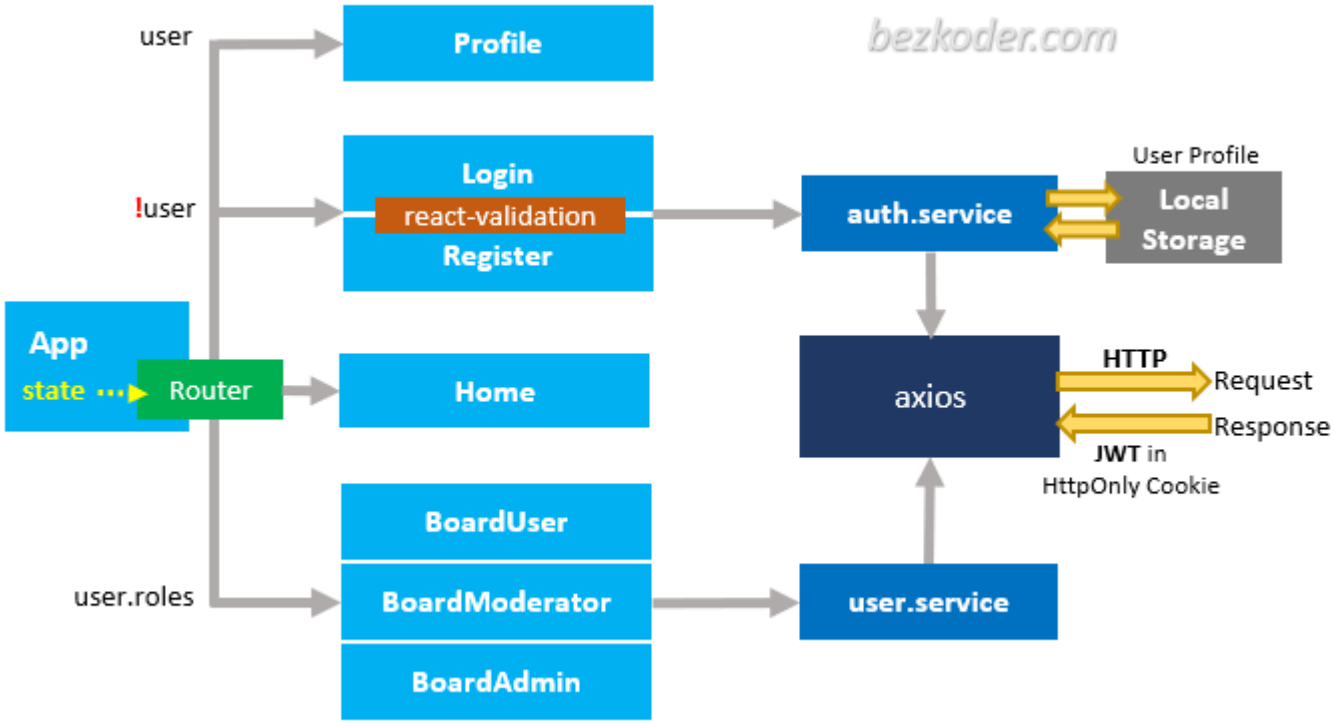
```
if (!items) {
  return <p>no data fetched</p>;
}
const liste = items.map(item =>
  <li id={item.id}>
   {item.name}
  </li>);

return (
  <div>
    <h1>Public Area Data</h1>
    <div>
     { liste }
    </div>
  </div>
 )
};
```

Abläufe im React-Frontend

https://www.bezkoder.com/spring-boot-react-jwt-auth/

https://www.bezkoder.com/react-login-example-jwt-hooks/



auth.service.js

uses Axios for HTTP requests and Local Storage for user information & JWT.

```
const login = (username, password) => {
  return axios
    .post(API_URL + "login",
      { username, password, })
    .then((response) => {
      if (response.data.username) {
        localStorage.setItem("user",
           JSON.stringify(response.data));
        return response.data;
      }
    }).catch((error) => {
      console.log(error);
      throw error;
    })
};

/* By doing throw error inside .catch(),
you're allowing the caller to still catch
the error. Otherwise, the promise will
resolve with an error object, which is
unexpected behavior for most consumers.
*/
```

```
const logout = () => {
  localStorage.removeItem("user");
};

const getCurrentUser = () => {
  return JSON.parse(localStorage.getItem("user"));
};

const getJwtHeader = () => {
  let tok = JSON.parse(localStorage.getItem("user")).token
  return {headers: { Authorization: `Bearer ${tok}`}}
}

const AuthService = {
  login,
  logout,
  getJwtHeader,
  getCurrentUser,
}

export default AuthService;
```

login.jsx

Funktion rendert Formular, nutzt Zustandsvariablen und *useState* Hooks; beim Submit wird die *login*-Funktion des AuthService aufgerufen

```
import { useNavigate } from 'react-router-dom';
import AuthService from '../services/auth.service';
import React, { useState } from 'react';

export default function Login() {
  const redirect = useNavigate();
  const [entries, setEntries] = useState({ username: "", password: "" })
  function store(e){ setEntries({...entries,
          [e.target.name]: e.target.value });
  }

  const handleSubmit = (event) => {
    event.preventDefault();
    try {
      AuthService.login(entries.username, entries.password)
      .then((res) => {
        if (res.username) {
          redirect(0);  // that actually does the redirect correctly
        }
      })
      .catch((err) => {
        if (err.status === 401) {
          alert("Wrong username or password");
        }
      });
    } catch (err) { console.log(err); }
};
```

```
  return (
    <div className="login-container">
      <form onSubmit={handleSubmit}>
        <h2>Login</h2>
        <div className="form-group">
          <label htmlFor="username">Username:</label>
          <input type="text" id="username"
            name="username" value={entries.username}
            onChange={store} required
          />
        </div>
        <div className="form-group">
          <label htmlFor="password">Password:</label>
          <input type="password" id="password"
            name="password" value={entries.password}
            onChange={store} required
          />
        </div>
        <button type="submit">Login</button>
      </form>
    </div>
  );
}
```

Requests mit JWT - Header

JWT-Token muss für authorisierte Requests im Request-Header hinzugefügt werden, e.g.

```
let data = [];
const token =  JSON.parse(localStorage.getItem("user")).accessToken;
const header = { headers: {"Authorization": `Bearer ${token}`} };
await client.get("/someSecuredPath", header)
   .then((response)=> {
     data = response.data;
   });
```

Your turn

in eigene App integrieren!
GlobalNavigation als Funktion?
Login-Form mit Validierung und MUI?