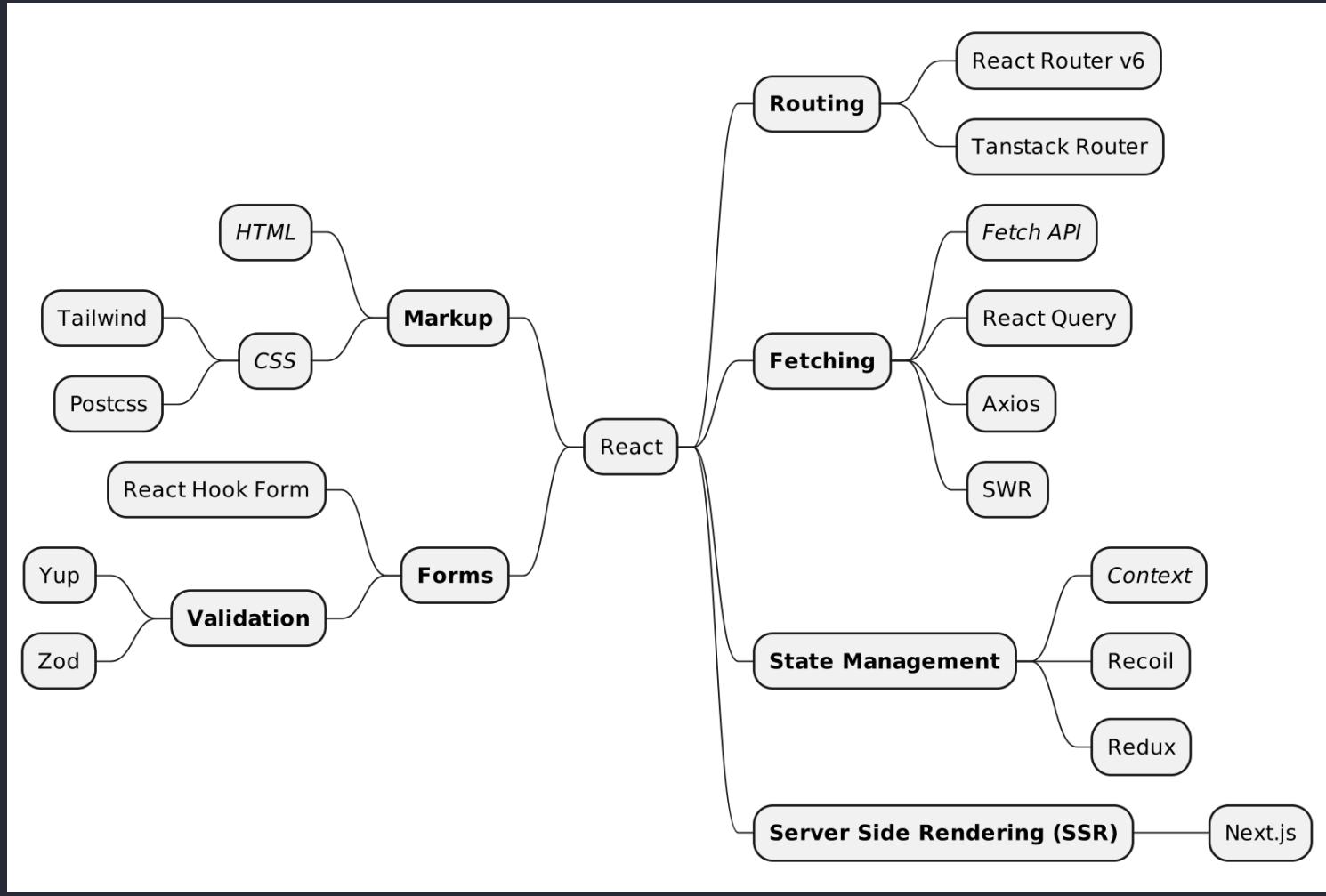
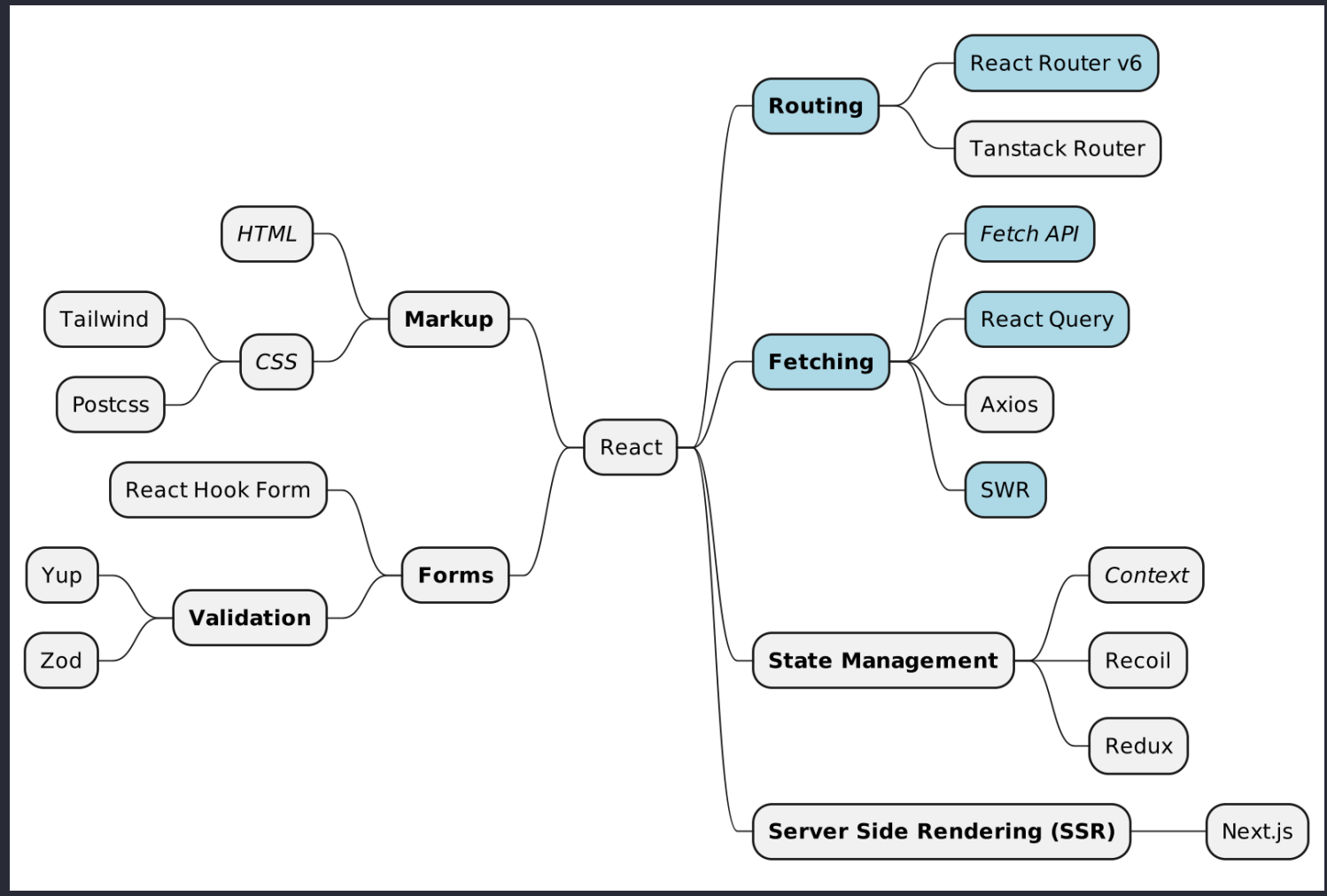
PB138 Moderní značkovací jazyky

Týden 11 - Routing and Fetching

Checkpoint

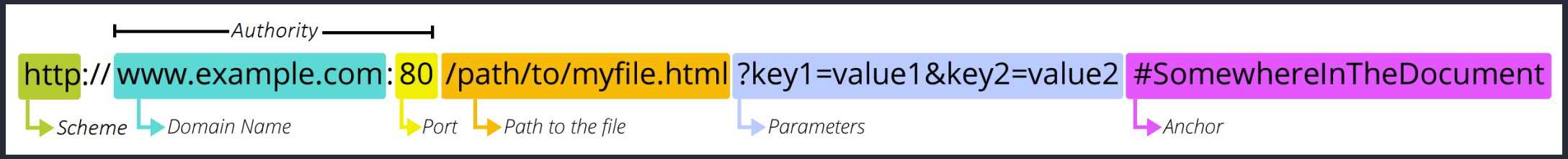

React
Routing
React Router v6
Tanstack Router
Fetching
Fetch API
React Query
Axios
SWR
State Management
Context
Recoil
Redux
Server Side Rendering (SSR)
Next.js
Markup
HTML
CSS
Tailwind
Postcss
Forms
React Hook Form
Validation
Yup
Zod

Checkpoint


React
Routing
React Router v6
Tanstack Router
Fetching
Fetch API
React Query
Axios
SWR
State Management
Context
Recoil
Redux
Server Side Rendering (SSR)
Next.js
Markup
HTML
CSS
Tailwind
Postcss
Forms
React Hook Form
Validation
Yup
Zod

Routing

URL


Authority
http://
www.example.com
:
80
/path/to/myfile.html
?key1=value1&key2=value2
#SomewhereInTheDocument
Scheme
Domain Name
Port
Path to the file
Parameters
Anchor

Web API's

https://developer.mozilla.org/en-US/docs/Web/API

History

```
// `history` is a global object
history.pushState({page: 2}, "title 2", "?page=2")
history.replaceState({page: 3}, "title 3", "?page=3")
// These 2 are equivalent:
history.back()
history.go(-1)
// These 2 are equivalent:
history.forward()
history.go(1)
// Refresh the current page
history.go()
```

Location

```
// The current location is:
// `https://developer.mozilla.org:8080/en-US/search?q=URL#search-results-close-container`
const loc = document.location;

console.log(loc.href);      // the whole URL
console.log(loc.protocol);  // "https:"
console.log(loc.host);      // "developer.mozilla.org:8080"
console.log(loc.hostname);  // "developer.mozilla.org"
console.log(loc.port);      // "8080"
console.log(loc.pathname);  // "/en-US/search"
console.log(loc.search);    // "?q=URL"
console.log(loc.hash);      // "#search-results-close-container"
console.log(loc.origin);    // "https://developer.mozilla.org:8080"
```

Client-side routing

> Routing = provázání cesty v URL k určité komponentě

Na dnešním cviku si ukážeme knihovnu `React Router v6`:

```
npm install react-router-dom
```

Při ~~kopírování kódu~~ inspirování se kódem z internetu pozor na starší a dost odlišné verze `v4` či `v5`.

Více type-safe alternativa na vzestupu: `Tanstack Router`.

Wrapper v `main.tsx`

Obalením celé naší aplikace do `<BrowserRouter>` k němu získáme všude přístup.

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import {BrowserRouter} from "react-router-dom";

import App from './App'

ReactDOM
    .createRoot(document.getElementById('root') as HTMLElement)
    .render(
      <React.StrictMode>
          <BrowserRouter>
              <App />
          </BrowserRouter>
      </React.StrictMode>,
);
```

Routování v `App.tsx`

```tsx
import React from "react";
import { Routes, Route } from "react-router-dom";

export const App = () => {
    return (
        <Routes>
            <Route path="/" element={<Layout />}>
                <Route index element={<Home />} />
                <Route path="about" element={<About />} />
                <Route path="dashboard" element={<Dashboard />} />

                <Route path="*" element={<NoMatch />} />
            </Route>
        </Routes>
    );
}
```

Spoločný vzhled v `Layout.tsx`

```tsx
import {NavLink, Outlet} from "react-router-dom";

export const Layout = () => {
    return (
        <div>
            <nav>
                <ul>
                    <li>
                        <Link to="/">Home</Link>
                    </li>
                    <li>
                        <Link to="/about">About</Link>
                    </li>
                    <li>
                        <Link to="/dashboard">Dashboard</Link>
                    </li>
                </ul>
            </nav>

            <main>
                <Outlet />
            </main>
        </div>
  );
}
```

Zpracování dynamické URL

```
export const App = () => {
  return (
    <div>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route path="/product/:id" element={<ProductView />} />
        </Route>
      </Routes>
    </div>
  );
}

const ProductView = () => {
  let { id } = useParams();
  return (
    <div>
      <h3>Product {id}</h3>
    </div>
  );
}
```

Zpracování query parametrů

```
import {useSearchParams} from "react-router-dom";

export const UseSearchParams = () => {
    const [searchParams, setSearchParams] = useSearchParams();
    const filter = searchParams.get("contains") ?? "";

    const data = ["Air","Bat","Cot","Doll","Eye","Frog","Grass","Hut","Igloo","..."]

    return (
        <>
            <h3>Nouns</h3>
            <ul>
                {data
                    .filter(noun => noun.toLowerCase().includes(filter))
                    .map(noun => <li key={noun}><small>{noun}</small></li>)}
            </ul>
        </>
    );
};
```

Alternativní routování

Z jednoho pole tak můžeme chtít generovat `<nav>` menu i routování.

```
import React from "react";
import ReactDOM from "react-dom/client";
import { createBrowserRouter, RouterProvider} from "react-router-dom";

export const pages = [
    {
        path: "/",
        element: <Layout />,
    },
    // ...
];
const router = createBrowserRouter(pages);

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <RouterProvider router={router} />
  </React.StrictMode>
);
```

Fetching

`fetch` funkce

Nahradila starší `XMLHttpRequest` objekt.

```
fetch('https://jsonplaceholder.typicode.com/users')
    .then(response => response.json())
    .then(data => console.log(data))
    .catch((error) => console.error('Error:', error));
```

`fetch` v komponentě

```tsx
import {useParams} from "react-router-dom";
import {useEffect, useState} from "react";

export const Fetch = () => {
    let { id } = useParams();
    const [data, setData] = useState<User | null>(null);

    useEffect(() => {
        const fetchData = async () => {
            const response = await fetch(`https://jsonplaceholder.typicode.com/users/${id}/`);
            const newData = await response.json();
            setData(newData);
        };

        fetchData();
    }, [id]);

    return (
        <>
            <h3>User page</h3>

            {data ? <article>{data.name}</article> : <span>Loading...</span>}
        </>
    );
};
```

V dnešní době ***nechceme*** psát manuálně.

Stale-While-Revalidate koncept

Lightweight knihovna obalující `fetch`.

```
import useSWR from 'swr'

const fetcher = url => fetch(url).then(r => r.json())

const Profile = () => {
  const { data, isLoading, error } = useSWR('/api/user', fetcher)

  if (error) {
      return <div>failed to load</div>
  }
  if (!data) {
      return <div>loading...</div>
  }

  return <div>hello {data.name}!</div>
}
```

React Query

Feature-rich knihovna.

```
const queryClient = new QueryClient()

const App = () => {
  return (
    <QueryClientProvider client={queryClient}>
      <App />
    </QueryClientProvider>
  )
}
```

React Query

```
import { useQuery } from '@tanstack/react-query';
import { AnimalsApi } from '../services';

const AnimalsPage: FC = () => {
    const { data: animals } = useQuery({
        queryKey: ['animals'],
        queryFn: () => AnimalsApi.getAll()
    })

    return <div>
        <h2>Animals</h2>
        { animals?.data.map(animal => <AnimalCard key={animal.id} animal={animal} />) || <></> }
    </div>
}
```

Otázky?

- URL
- Web API's
- Routing
- Fetching

Odpovědník

Week 11 - Routing & Fetching

Dnešní úkol

V dnešním úkolu budete mít k dispozici hotový backend s databází a rozpracovaný frontend.

Nezapomeňte pro **oba** projekty nainstalovat závislosti (`npm install`) a spustit je (`npm run start` / `npm run dev`). Vhodně vytvořte `.env` soubor v `backendu` .

Použijte `React Router v6` a `React Query` na dokončení `frontend` u:

1. Přidejte routování do `App.tsx` dle TODO komentáře.
2. Přidejte fetchování do stránek v `src/pages` dle TODO komentářů.