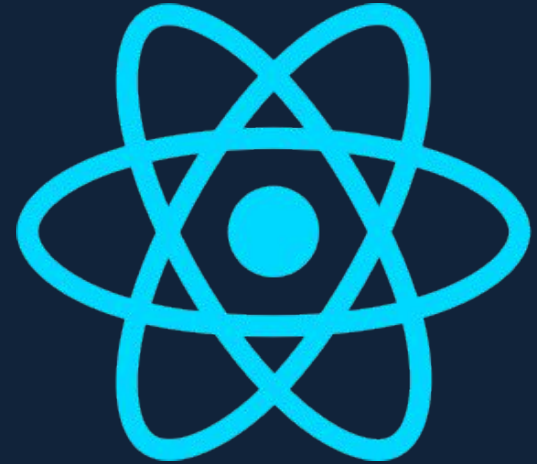




Workshop

React Router



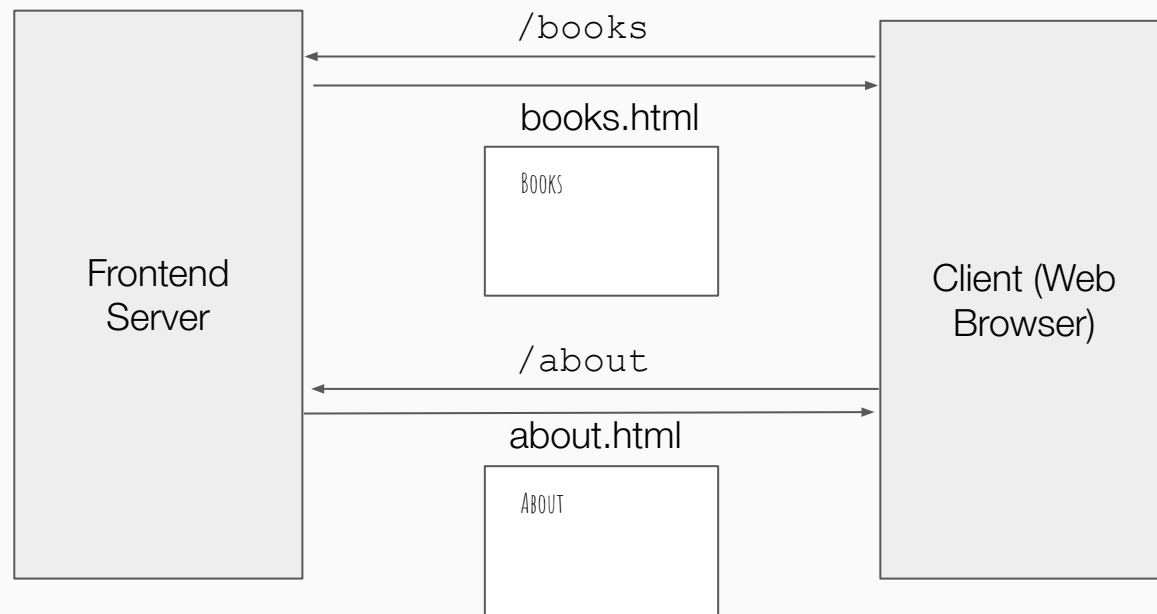
Why / What you'll learn



- How to define Routes in a declarative way
- Using `react-router-dom` to build a more complex application

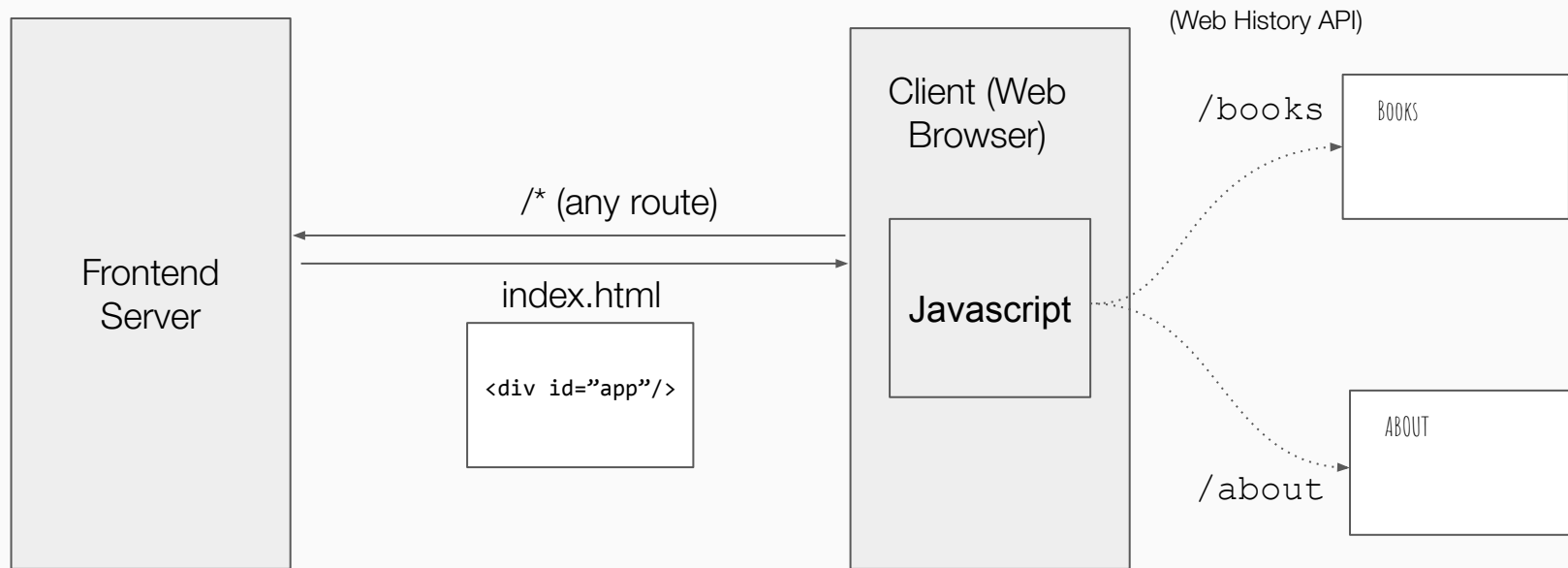
Server-Side Routing

Every url path returns a distinct html file



Client-Side Routing

Server returns the same (almost empty) index.html for any url
Javascript looks at the url path and renders matching content



A **S**ingle **P**age **A**pplication handles routing in
most cases by itself.

A router library helps with managing the routing logic/state.

The router maps URLs to components / screens

`/books` → `<BookList />`

React Router

React Router is a routing library for React.

It's “just” React and follows the principles we've learned so far.

The React Router Module

- **Declarative** routing for React
- Since v6.4 React Router introduces and recommends new Data API
- Functions & Components of React Router in the `react-router-dom` module:

- `npm install react-router-dom`
- `import { createBrowserRouter, RouterProvider } from 'react-router-dom'`

```
const router = createBrowserRouter([...])
```

Create a router with your route definitions

```
<RouterProvider router={router} />
```

Primary component of React Router. It renders the element that matches the current url.

Router In Action

<code>

Create a router and render a RouterProvider

```
import { createBrowserRouter, RouterProvider } from "react-router-dom";

const router = createBrowserRouter([
  {
    path: "/",
    element: <App />,
  },
]);

root.render(<RouterProvider router={router} />);
```

Router In Action

<code>

Create a router and render a RouterProvider

```
import { createBrowserRouter, RouterProvider } from "react-router-dom";

const router = createBrowserRouter([
  {
    path: "/",
    element: <App />,
  },
]);

root.render(<RouterProvider router={router} />);
```

Here you define which url should render which element

Router In Action

<code>

Create a router and render a RouterProvider

```
import { createBrowserRouter, RouterProvider } from "react-router-dom";
```

```
const router = createBrowserRouter([  
  {  
    path: "/",  
    element: <App />,  
  },  
]);
```

This is where the <App/ > element will be rendered

```
root.render(<RouterProvider router={router} />);
```


Defining Routes

Defining Routes: Multiple siblings

<code>

```
const router = createBrowserRouter([
  {
    path: "/",
    element: <App />,
  },
  {
    path: "/contact",
    element: <ContactScreen />,
  },
  {
    path: "/about",
    element: <AboutScreen title="About"/>,
  },
  ...
]);
```

setting props is allowed



Defining Routes: Extra Properties

<code>

```
const router = createBrowserRouter([
  {
    path: "/",
    element: <App />,
    errorElement: <ErrorScreen />,
    loader: () => { return "Hello"; },
    children: [
      {
        path: "",
        element: <BooksScreen />,
      },
    ],
  },
]);
```

Optional

Defining Routes: Extra Properties

<code>

```
const router = createBrowserRouter([
  {
    path: "/",
    element: <App />,
    errorElement: <ErrorScreen />,
    loader: () => { return "Hello"; },
    children: [
      {
        path: "",
        element: <BooksScreen />,
      },
    ],
  },
]);
```

React Element to render when
there's an error (uncaught
exception, routing error)

Defining Routes: Extra Properties

<code>

```
const router = createBrowserRouter([
  {
    path: "/",
    element: <App />,
    errorElement: <ErrorScreen />,
    loader: () => { return "Hello"; },
    children: [
      {
        path: "",
        element: <BooksScreen />,
      },
    ],
  },
]);
```

Function, which will run when the route matches.

Access the return value in the component with `useLoaderData()`

Route Definition: Extra Properties

<code>

```
const router = createBrowserRouter([
  {
    path: "/",
    element: <App />,
    errorElement: <ErrorScreen />,
    loader: () => { return "Hello"; },
    children: [
      {
        path: "",
        element: <BooksScreen />,
      },
    ],
  },
]);
```

Children routes. We'll look at this in a bit...

Task

Install React Router and add a fallback error screen



Children Routes

Problem

By default React Router will exchange the **entire** content of the page on a route switch.

But there are often elements in the UI that should stay between route switches (Navigation, Footer, etc...)

Solution

Add children to your main route definition.

Use `<Outlet />` to tell React router where to render the children within your main component.

Children Routes

<code>

<App /> will be rendered as **frame** for matched child route

```
{
  path: "/",
  element: <App />,
  children: [
    {
      path: "",
      element: <HomeScreen />,           matches: /
    },
    {
      path: "about",
      element: <AboutScreen />,         matches: /about
    },
  ],
},
```

Children Routes

<code>

Also works on non root routes (nested routes), e.g. /admin

```
{
  path: "/admin",
  element: <AdminLayout />,
  children: [
    {
      path: "",
      element: <AdminDashboard />,      matches: /admin
    },
    {
      path: "settings",
      element: <AdminSettings />,      matches: /admin/settings
    },
  ],
},
```

The Outlet Component

<code>

AppHeader and AppFooter will be visible for all children routes

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

export const App = () => {
  return (
    <div className="App">
      <AppHeader />
      <Outlet />
      <AppFooter />
    </div>
  );
};
```

This is where the matching child route will be rendered

Task

Add a Books and About Screen



Links

The primary way to allow users to navigate
around your application.

The Link Component

- `<Link />` will render a fully accessible anchor tag with the proper href.
 - Property: `to="/my/route"`
- `<NavLink />` adds `.active` css class to anchor when link is active

```
import {  
  Link, NavLink  
} from 'react-router-dom'
```

Using NavLink

<code>

<NavLink> will render an <a> tag, that doesn't cause a page refresh when clicked

```
<div>
  <ul>
    <li><NavLink to="/home">Home</NavLink></li>
    <li><NavLink to="/about">About</NavLink></li>
  </ul>
</div>
```


Task

**Add navigation links to the Books
and About Screen**



Dynamic Route Params

Problem

So far all the routes we defined had a static path.

→ **Finite** number of urls our app can handle.


What if we want to use the isbn of a book in the url?

Detail of a book

A detailed View of a book including the Abstract, Number of Pages, Publisher and ISBN.

Book details should be available under
`/books/:isbn`

[BookMonkey](#) [Home](#) [Books](#) [Login](#)



Design Patterns

Elements of Reusable Object-Oriented Software von
Erich Gamma / Richard Helm / Ralph E. Johnson / John
Vlissides

Capturing a wealth of experience about the design of object-oriented software, four top-notch designers present a catalog of simple and succinct solutions to commonly occurring design problems. Previously undocumented, these 23 patterns allow designers to create more flexible, elegant, and ultimately reusable designs without having to rediscover the design solutions themselves.

Das Buch hat 395 Seiten und wurde bei [Addison-Wesley](#) veröffentlicht

ISBN: 978-0-20163-361-0

[Buch bearbeiten](#)

Dynamic Route Params

<code>

Every path segment that starts with `:` becomes a dynamic param

```
{  
  path: "/books/:isbn",  
  element: <BookDetailScreen />,  
}
```


Read Params In A Component

<code>

Read the params via the useParams hook

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

const BookDetailsScreen = () => {
  const { isbn } = useParams<{ isbn: string }>();
  // use the isbn to load your data
  return <p>ISBN: {isbn}</p>;
};
```



Type for expected
params, following the
URL (/books/:isbn).

Task

Add a BookDetailScreen





We teach.

workshops.de