DEV ACADEMY
TE KURA HANGARAU
O AOTEAROA

Client-side routing

React Router DOM

Week 2, Day 3

Agenda

- What is client-side routing?
- React Router DOM
- Links
- Defining and accessing dynamic segments of URLs (params)
- Nested routes and Outlet

Client-side routing

- Components shown and hidden as required depending on the URL
  - e.g. `http://localhost:3000/about` shows an `< About />` component
  - components are added and removed from the virtual DOM
- React Router DOM integrates with browser functionality;
  - Back and Forward buttons (history)
  - Refresh button
  - url params

React Router DOM

1. Create app routes in `routes.tsx` file

```tsx
import { Route, createBrowserRouter, createRoutesFromElements } from 'react-router-dom'

const routes = createRoutesFromElements([
  <Route path="/" element={<Home />} />,
  <Route path="/about" element={<About />} />,
  <Route path="/contact" element={<Contact />} />
])

export const router = createBrowserRouter(routes)
```

1. Import `router` into `index.tsx` file

```tsx
import { createRoot } from 'react-dom/client'
import { RouterProvider } from 'react-router-dom'

import { router } from './routes.tsx'

const root = createRoot(document.getElementById('app') as HTMLElement)
root.render(
    <RouterProvider router={router} />
)
```

Links

- We can use the `<Link>` element to to navigate between pages, without reloading the WHOLE page

```jsx
import { Link } from 'react-router-dom'

function Home() {
  return (
    <div>
      <h1>Home</h1>
      <Link to="/about">About</Link>
      <Link to="/contact">Contact</Link>
    </div>
  )
}
```

- ❌ Do not use `<a>` tags!
- ✅ Use `<Link></Link>` instead

Defining dynamic segments of URLs (params)

- We can define dynamic segments of URLs using the `:paramName` syntax

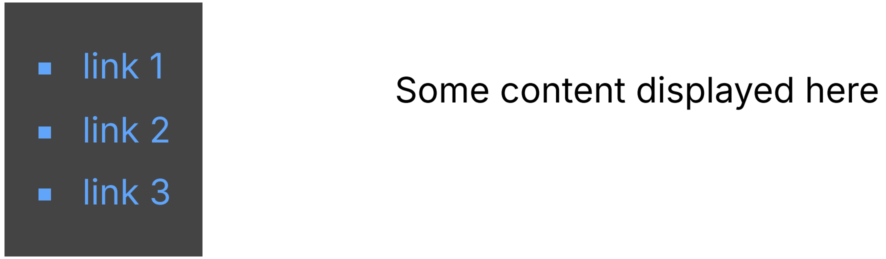
```
import { Route, createBrowserRouter, createRoutesFromElements, } from 'react-router-dom'

const routes = createRoutesFromElements([
  <Route path="/" element={<Home />} />,
  <Route path="/about" element={<About />} />,
  <Route path="/contact" element={<Contact />} />
  <Route path="/products/:id" element={<Products />} />
  // E.g. the browser url could have: `localhost:5173/products/123`
])

export const router = createBrowserRouter(routes)
```

Accessing URL parameters

- We can access URL parameters using the `` `useParams()` `` hook

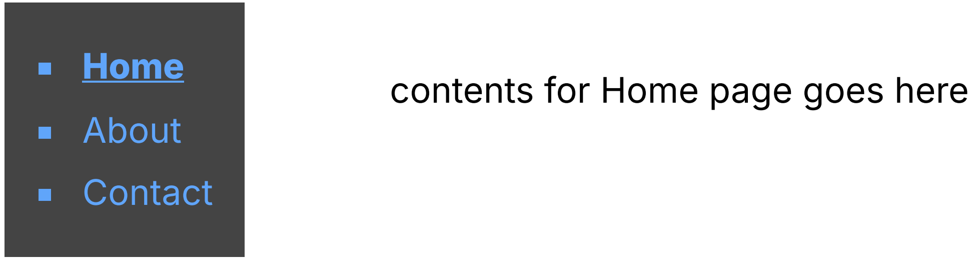
```jsx
import { useParams } from 'react-router-dom'

export default function Products() {

  // Browser url could have:
  // `http://localhost:5173/products/123`

  const { id } = useParams()
  return (
    <div>
      <h1>Product details</h1>
      <p>id: {id}</p>
    </div>
  )
}

```

Building a Nav bar

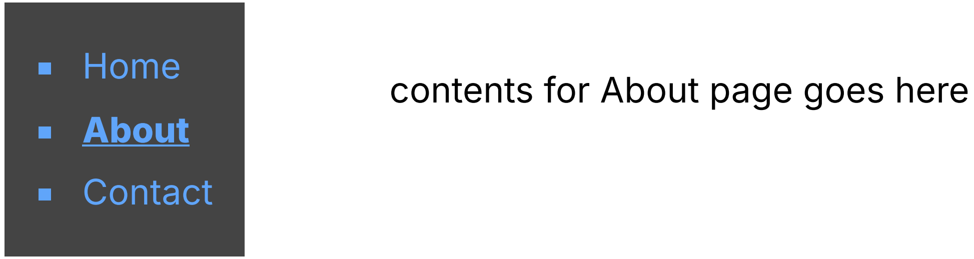


- We want to dynamically display the content of the page based on the link clicked
- Meaning that we want to display different content for each link
- We can do this using:
  - a. Nested routes
  - b. The `` `< Outlet />` `` component from react-router-dom

a. Nested routes - visual



contents for Home page goes here



contents for About page goes here

b. Nested routes - code

```
// BEFORE
const routes = createRoutesFromElements([
  <Route path="/" element={<Home />} />,
  <Route path="/about" element={<About />} />,
  <Route path="/contact" element={<Contact />} />
  <Route path="/products/:id" element={<Products />} />
])

export const router = createBrowserRouter(routes)
```

```
// AFTER
const routes = createRoutesFromElements(
  <Route path="/" element={<Layout />}> // contains nav
    <Route index element={<Home />} /> // index is the default
    <Route path="about" element={<About />} />
    <Route path="contact" element={<Contact />} />
    <Route path="products/:id" element={<Products />} />
  </Route>
)

export const router = createBrowserRouter(routes)
```

b. The `<Outlet/>` component

```
function Layout() {
  return (
    <div>
      <nav>
        <Link to="/">Home</Link>
        <Link to="/about">About</Link>
        <Link to="/contact">Contact</Link>
      </nav>
      <Outlet /> {/* <--- this is where the content will be displayed */}
    </div>
  )
}

export default App
```

- Anything outside of the `<Outlet>` component will be displayed on every page
- In this case, the navigation bar `<nav>`
- Time for a teachable meme moment...

b. The `<Outlet/>` component - visual

Deeply nested routes

```
const routes = createRoutesFromElements(
  <Route path="/" element={<Layout />}> // `<Outlet/>` is here
    <Route index element={<Home />} />
    <Route path="about" element={<About />} />
    <Route path="contact" element={<Contact />} />
    <Route path="products">  // `<Outlet/>` is here
      <Route index path=":id" element={<Product />} />
      <Route path=":id" element={<Product />} />
      <Route path=":id/reviews" element={<Reviews />} />
    </Route>
  </Route>
)

export const router = createBrowserRouter(routes)
```

- We can nest routes as deep as we want

DEMO

converting `` `react-state-kata` `` to React Router DOM!