

ReactJS Router



Eko Kurniawan Khannedy

Eko Kurniawan Khannedy

- Technical architect at one of the biggest ecommerce company in Indonesia
- 14+ years experiences
- www.programmerzamannow.com
- youtube.com/c/ProgrammerZamanNow



Eko Kurniawan Khannedy

- Telegram : [@khannedy](#)
- Linkedin : <https://www.linkedin.com/company/programmer-zaman-now/>
- Facebook : fb.com/ProgrammerZamanNow
- Instagram : instagram.com/programmerzamannow
- Youtube : youtube.com/c/ProgrammerZamanNow
- Telegram Channel : t.me/ProgrammerZamanNow
- Tiktok : <https://tiktok.com/@programmerzamannow>
- Email : echo.khannedy@gmail.com

Sebelum Belajar

- ReactJS Dasar

Pengenalan Single Page Application

Client atau Server Side

- Saat kita membuat Web, terdapat dua pilihan yang bisa kita pilih, CSR (Client Side Render) atau SSR (Server Side Render)
- Saat kita menggunakan Frontend Framework seperti React, biasanya pilihan pertama adalah menggunakan CSR (Client Side Render)
- Hal ini menjadikan bagian Server tidak menangani halaman Web lagi dan tidak perlu membuat halaman HTML lagi, semua dikerjakan di Client oleh React
- Secara otomatis CSR akan semakin cepat dibandingkan SSR, karena komunikasi ke Server bisa dibuat minimal dan data yang dikirim/diterima tidak terlalu besar

Diagram Server Side Render

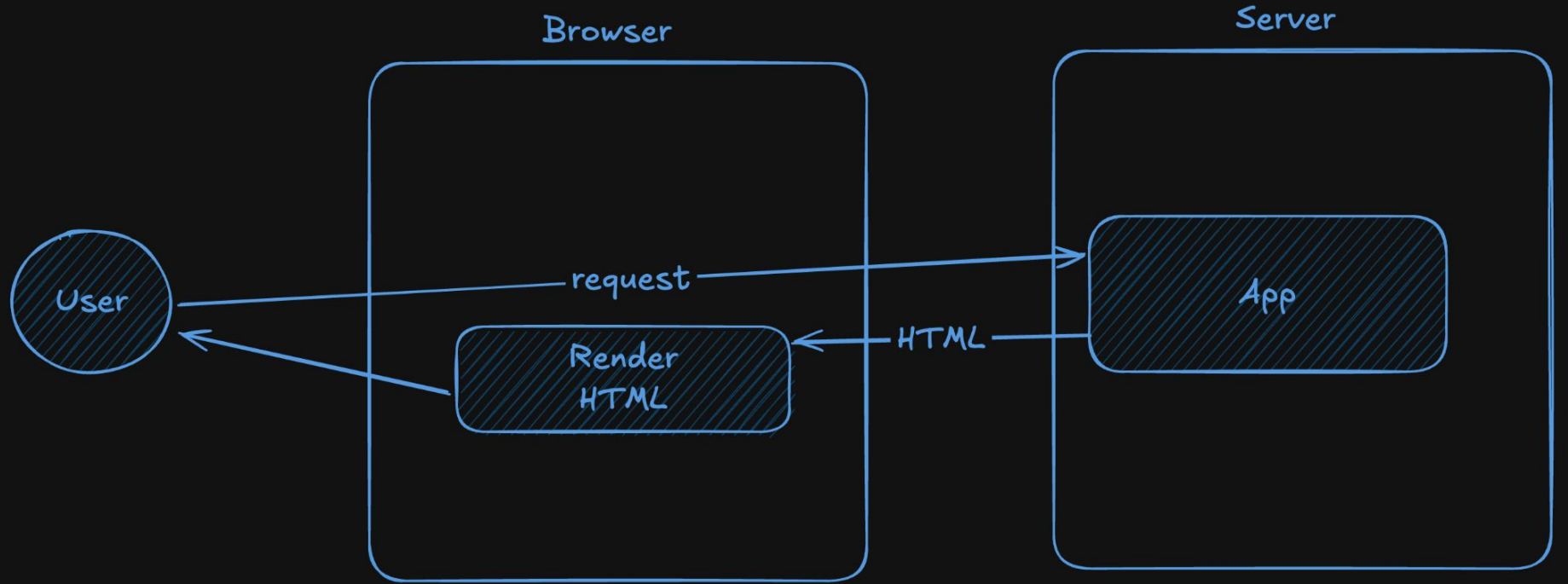
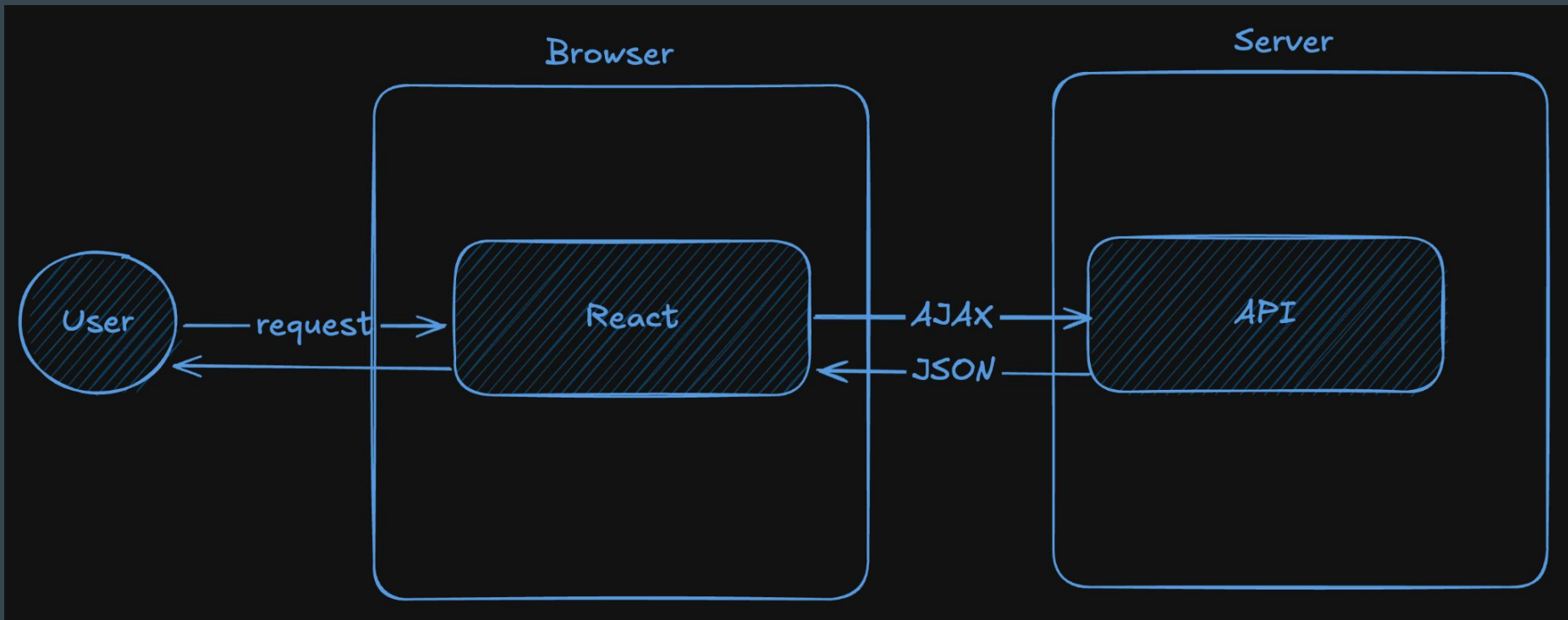


Diagram Client Side Render



Multi Page Application

- Sebelumnya, ketika belajar React Dasar, kita membuat banyak sekali halaman HTML, dan tiap halaman HTML menampilkan component React
- Hal ini artinya walaupun aplikasi yang sudah kita buat adalah CSR, namun tetap Multi Page
- Problem dengan MPA (Multi Page Application) adalah, setiap berganti halaman, maka Browser akan menjalankan ulang program React nya dari awal lagi
- Hal ini menyebabkan walaupun sudah menggunakan CSR, namun kemungkinan tidak optimal karena harus di load ulang semuanya ketika berpindah halaman

Single Page Application

- Kebalikan dari MPA, adalah SPA (Single Page Application), dimana kita hanya membuat satu halaman HTML
- Nanti halaman itu yang akan menerima semua request, dan isi halamannya akan berubah sesuai dengan request yang diminta oleh User atau sesuai response dari Server
- Keuntungan dari SPA adalah perpindahan halaman lebih cepat, karena hanya mengubah isi content saja, tidak harus me-load ulang seluruh halaman lagi
- Namun sayangnya, secara default, React tidak mendukung SPA dengan baik
- Kita harus lakukan semuanya secara manual jika ingin mengganti-ganti isi dari halaman, dan itu sangat rumit dilakukan

Router Library

Router Library

- Salah satu hal yang bisa kita lakukan dibanding mengganti-ganti halaman secara manual, adalah menggunakan library khusus untuk itu, yaitu Router Library
- Router Library adalah library yang digunakan agar Component yang ditampilkan bisa berbeda-beda tergantung dari URL yang diakses
- Dengan begitu, kita bisa fokus membuat Component untuk halaman, tidak perlu memikirkan lagi kompleksitas bagaimana cara melakukan routing (menentukan Component mana yang ditampilkan untuk URL yang diakses)
- Untungnya, banyak sekali Router Library yang tersedia untuk React

React Router

- Ada banyak sekali Router Library yang bisa kita gunakan untuk React
- <https://react.libhunt.com/libs/router/react>
- Namun pada materi ini, kita akan menggunakan salah satu Router Library yang populer, yaitu adalah React Router
- <https://reactrouter.com/>
- React Router adalah salah satu library untuk routing yang opensource dan saat ini sangat populer digunakan di ekosistem React
- <https://github.com/remix-run/react-router>

Membuat Project

Membuat Project

- `npm create vite@latest belajar-reactjs-router -- --template react`
- Upgrade versi React ke versi 19

Menambah Library React Router

- `npm install react-router`

Setup

Setup

- Berbeda Saat dengan MPA, ketika membuat SPA, kita cukup membuat satu halaman html sebagai entry point (titik masuk) semua request
- Nanti, penentuan komponen mana yang akan ditampilkan, diurus oleh React Router

BrowserRouter

- Ketika menggunakan React Router, saat membuat melakukan render komponen menggunakan React, kita bisa bungkus aplikasi React kita menggunakan komponen BrowserRouter
- https://api.reactrouter.com/v7/functions/react_router.BrowserRouter.html

Kode : src/main.jsx

main.jsx x

```
1 import {StrictMode} from 'react'
2 import {createRoot} from 'react-dom/client'
3 import './index.css'
4 import App from './App.jsx'
5 import {BrowserRouter} from "react-router";
6
7 createRoot(document.getElementById('root')).render(
8   <StrictMode>
9     <BrowserRouter>
10       <App/>
11     </BrowserRouter>
12   </StrictMode>,
13 )
```

Routing

Routing

- Routing adalah pemetaan antara URL path dan komponen yang akan ditampilkan
- Untuk melakukan Routing, kita bisa menggunakan komponen `<Routes>`
- https://api.reactrouter.com/v7/functions/react_router.Routes.html
- Dan untuk pemetaan tiap Routing nya, kita bisa gunakan komponen `<Route>`
- https://api.reactrouter.com/v7/functions/react_router.Route.html

Tugas

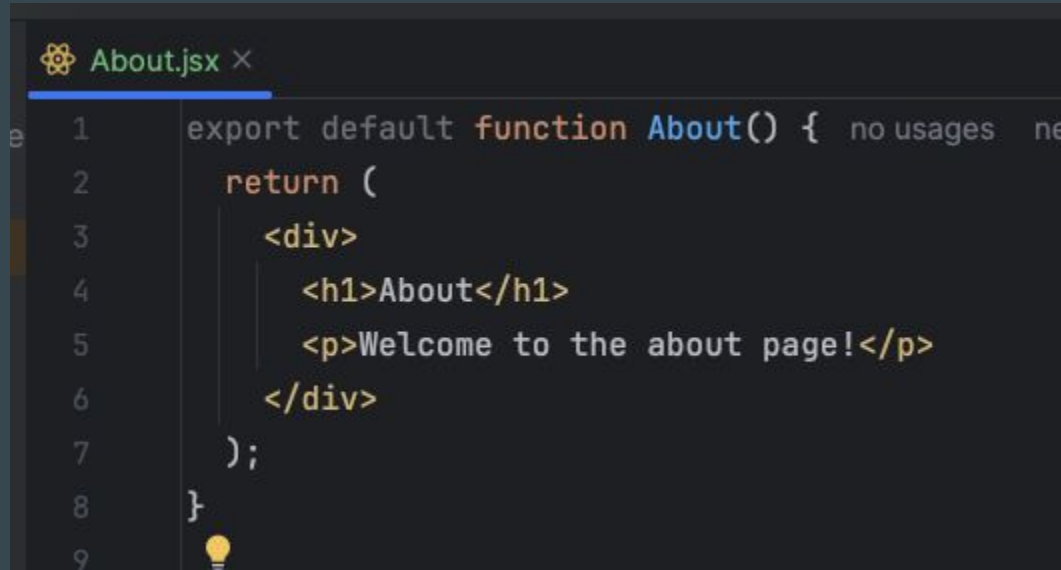
- Sekarang kita akan coba membuat Routing sederhana, kita akan tambahkan 2 buah komponen Home dan About
- Kita akan lakukan routing untuk path / ke Home
- Dan path /about ke About

Kode : src/Home.jsx

Home.jsx X

```
1 export default function Home() { no usages new *
2   return (
3     <div>
4       <h1>Home</h1>
5       <p>Welcome to the home page!</p>
6     </div>
7   );
8 }
```


Kode : src/About.jsx



```
1 export default function About() { no usages ne
2   return (
3     <div>
4       <h1>About</h1>
5       <p>Welcome to the about page!</p>
6     </div>
7   );
8 }
9
```

Kode : src/main.jsx

main.jsx ×

```
1 import {StrictMode} from 'react'
2 import {createRoot} from 'react-dom/client'
3 import {BrowserRouter, Route, Routes} from "react-router";
4 import Home from "./Home.jsx";
5 import About from "./About.jsx";
6
7 createRoot(document.getElementById('root')).render(
8   <StrictMode>
9     <BrowserRouter>
10       <Routes>
11         <Route path="/" element={<Home/>}/>
12         <Route path="/about" element={<About/>}/>
13       </Routes>
14     </BrowserRouter>
15   </StrictMode>,
16 )
```

Nested Route

- Saat nanti kita membuat routing, kadang terdapat routing yang memiliki prefix yang sama
- Jika kita harus buat satu per satu, maka akan tidak efektif ketika pembuatan kodanya
- Untungnya, React Router mendukung Nested Route
- Kita bisa membuat komponen Route di dalam komponen Route, dan secara otomatis Route path di atasnya akan digunakan sebagai prefix path untuk child Route nya

Tugas

- Sekarang kita akan coba buat komponen Product, Customer dan Seller
- Kita akan buat 3 route
- Route path /data/products ke Product
- Route path /data/customers ke Customer
- Route path /data/sellers ke Seller
- Kita akan buat menggunakan Nested Route

Kode : src/main.jsx

```
createRoot(document.getElementById('root')).render(  
  <StrictMode>  
    <BrowserRouter>  
      <Routes>  
        <Route path="/" element={<Home/>}/>  
        <Route path="/about" element={<About/>}/>  
        <Route path="/data">  
          <Route path="products" element={<Product/>}/>  
          <Route path="customers" element={<Customer/>}/>  
          <Route path="sellers" element={<Seller/>}/>  
        </Route>  
      </Routes>  
    </BrowserRouter>  
  </StrictMode>,  
  </>  
)
```

Index Route

- Komponen Route memiliki atribut spesial bernama index
- Atribut ini digunakan sebagai default halaman index
- Ini biasanya digunakan untuk path awal, misal /, atau untuk nested path /data/...
- Contoh sebelumnya, saat kita membuka /data, maka tidak ada komponen yang ditampilkan, kita bisa tampilkan komponen yang kita mau dengan menggunakan Route index
- Misal, sekarang kita akan buat komponen Data

Kode : src/main.jsx

```
11 createRoot(document.getElementById('root')).render(  
12   <StrictMode>  
13     <BrowserRouter>  
14       <Routes>  
15         <Route index element={<Home/>}/>  
16         <Route path="/about" element={<About/>}/>  
17         <Route path="/data">  
18           <Route index element={<Data/>}/>  
19           <Route path="products" element={<Product/>}/>  
20           <Route path="customers" element={<Customer/>}/>  
21           <Route path="sellers" element={<Seller/>}/>  
22         </Route>  
23       </Routes>  
24     </BrowserRouter>  
25   </StrictMode>,  
26 )
```

Outlet

Outlet

- Saat kita membuat halaman web, kadang beberapa halaman memiliki Layout yang sama. Misal header dan footer nya sama, namun isi content nya berbeda
- Sebenarnya kita bisa menggunakan JSX untuk melakukan ini dengan menggunakan children
- Namun, React Router memiliki cara yang lebih mudah, kita bisa membuat Route dengan component yang digunakan sebagai Layout
- Lalu kita bisa tambahkan Route di dalamnya yang digunakan sebagai children
- Namun kita tidak menggunakan children Props lagi, melainkan menggunakan Component Outlet
- https://api.reactrouter.com/v7/functions/react_router.Outlet.html

Tugas

- Sekarang kita akan coba buat komponen yang digunakan sebagai Layout dengan nama `DataLayout`.
- Layout ini kita gunakan sebagai Layout untuk semua halaman Data

Code src/DataLayout.jsx

DataLayout.jsx ×

```
1 import {Outlet} from "react-router";
2
3 export default function DataLayout() { no us
4   return (
5     <>
6       <div>
7         <h1>This is Header</h1>
8       </div>
9       <div>
10        <Outlet/>
11      </div>
12      <div>
13        <p>This is footer</p>
14      </div>
15    </>
16  )
```

Kode : src/main.jsx

```
11
12 createRoot(document.getElementById('root')).render(
13   <StrictMode>
14     <BrowserRouter>
15       <Routes>
16         <Route index element={<Home/>}/>
17         <Route path="/about" element={<About/>}/>
18         <Route path="/data" element={<DataLayout/>}>
19           <Route index element={<Data/>}/>
20           <Route path="products" element={<Product/>}/>
21           <Route path="customers" element={<Customer/>}/>
22           <Route path="sellers" element={<Seller/>}/>
23         </Route>
24       </Routes>
25     </BrowserRouter>
26   </StrictMode>,
27 )
```

Route Param

Route Param

- Saat kita membuat URL Path, kadang kita tidak membuat Path yang statis, kadang kita membuat Path yang dinamis
- Misal, kita ingin menambahkan id / kode pada Path, misal `/products/123`, dimana 123 adalah id dari Product
- Di React Router, kita juga bisa lakukan itu dengan menambah tanda `:` (titik dua) diikuti dengan nama param, misal `/products/:id`, id dianggap sebagai parameter dinamis
- Jika kita menggunakan `/products/123`, maka 123 akan dianggap sebagai data parameter id

Tugas

- Kita akan menambahkan komponen ProductDetail, dimana komponen tersebut akan kita gunakan pada route path /data/products/:id

Kode : src/ProductDetail.jsx

 ProductDetail.jsx ×

```
1 export default function ProductDetail() {  
2     return (  
3         <>  
4             <h1>Product Detail</h1>  
5             <p>product detail page</p>  
6         </>  
7     )  
8 }
```


Kode : src/main.jsx

```
12  
13 createRoot(document.getElementById('root')).render(  
14   <StrictMode>  
15     <BrowserRouter>  
16       <Routes>  
17         <Route index element={<Home/>}/>  
18         <Route path="/about" element={<About/>}/>  
19         <Route path="/data" element={<DataLayout/>}>  
20           <Route index element={<Data/>}/>  
21           <Route path="products" element={<Product/>}/>  
22           <Route path="products/:id" element={<ProductDetail/>}/>  
23           <Route path="customers" element={<Customer/>}/>  
24           <Route path="sellers" element={<Seller/>}/>  
25         </Route>  
26       </Routes>
```

useParams()

- Saat kita membuat Path yang dinamis, biasanya kita ingin mendapatkan informasi Path yang dinamis tersebut
- Seluruh dynamic param yang terdapat di URL Path secara otomatis akan disimpan dalam object, dan kita bisa mengambilnya menggunakan function useParams()
- https://api.reactrouter.com/v7/functions/react_router.useParams.html
- Misal, sekarang kita akan coba tambahkan useParams() pada komponen ProductDetail

Kode : src/ProductDetail.jsx

ProductDetail.jsx ×

```
1 import {useParams} from "react-router";
2
3 export default function ProductDetail() {
4   const params = useParams();
5
6   return (
7     <>
8       <h1>Product Detail {params.id}</h1>
9       <p>product detail page</p>
10    </>
11  )
12 }
```

Multiple Route Param

- Kita bisa menambahkan dynamic param pada Path lebih dari satu
- Yang penting kita gunakan nama dynamic param yang berbeda, misal
- `/users/:userId/addresses/:addressId`
- Maka kita bisa dapatkan informasi `userId` dan `addressId` di Object hasil dari `useParams()`

Star Segment

Star Segment

- Jika route path diakhiri dengan `/*` (star segment), maka itu akan cocok dengan karakter apapun, termasuk karakter `/` itu sendiri
- Misal, ketika kita memiliki path `/files/*`, maka itu akan cocok dengan `/files/contoh` dan `/files/lagi/contoh`
- Star segment juga bisa diambil nilainya pada `useParams()`

Tugas

- Misal sekarang kita akan coba membuat komponen Image dengan Path /images/*

Kode : src/Image.jsx

```
Image.jsx ×  
1 import {useParams} from "react-router";  
2  
3 export default function Image() {  
4   const params = useParams();  
5  
6   return (  
7     <>  
8       <h1>Image</h1>  
9       <p>Image Page : {params['*']}</p>  
10    </>  
11  )  
12
```


Kode : src/main.jsx

```
14 createRoot(document.getElementById('root')).render(  
15   <StrictMode>  
16     <BrowserRouter>  
17       <Routes>  
18         <Route index element={<Home/>}/>  
19         <Route path="/about" element={<About/>}/>  
20         <Route path="/images/*" element={<Image/>}/>  
21         <Route path="/data" element={<DataLayout/>}>  
22           <Route index element={<Data/>}/>  
23           <Route path="products" element={<Product/>}/>  
24           <Route path="products/:id" element={<ProductDetail/>}/>  
25           <Route path="customers" element={<Customer/>}/>  
26           <Route path="sellers" element={<Seller/>}/>  
27         </Routes>
```

Star Segment sebagai Not Found Page

- Apa yang terjadi jika kita mengakses URL Path yang tidak tersedia di Router?
- Kita akan mendapatkan error bahwa tidak ada route yang cocok dengan path yang kita akses
- Salah satu yang biasa dilakukan di React Router adalah, kita bisa menambah star segment untuk path `/*` pada bagian bawah sebagai handler untuk halaman Not Found
- Sekarang kita akan coba buat komponen `NotFound`, dan registrasikan ke Route path `/*`

Kode : src/NotFound.jsx

🔗 NotFound.jsx ×

```
1 import { useParams } from "react-router";
2
3 export default function NotFound() {
4   const params = useParams();
5   return (
6     <>
7       <h1>Not Found : {params['*']}</h1>
8       <p>page is not found</p>
9     </>
10   )
11 }
```

Kode : src/main.jsx

```
15 createRoot(document.getElementById('root')).render(  
16   <StrictMode>  
17     <BrowserRouter>  
18       <Routes>  
19         <Route index element={<Home/>}/>  
20         <Route path="/about" element={<About/>}/>  
21         <Route path="/images/*" element={<Image/>}/>  
22         <Route path="/data" element={<DataLayout/>}>  
23           <Route index element={<Data/>}/>  
24           <Route path="products" element={<Product/>}/>  
25           <Route path="products/:id" element={<ProductDetail/>}/>  
26           <Route path="customers" element={<Customer/>}/>  
27           <Route path="sellers" element={<Seller/>}/>  
28         </Route>  
29         <Route path="/*" element={<NotFound/>}/>  
30       </Routes>
```

Navigation

Navigation

- Salah satu yang biasa kita lakukan saat membuat web, adalah berpindah dari satu halaman ke halaman yang lainnya
- Biasanya untuk perpindahan halaman, kita biasanya akan menggunakan Anchor Element, yaitu menggunakan tag `<a>`
- Namun permasalahannya adalah, ketika menggunakan Anchor Element, maka browser akan melakukan reload ke halaman baru, yang artinya halaman akan dimuat ulang, termasuk object React juga akan dibuat ulang semuanya
- Hal ini mungkin akan memperlambat proses perpindahan halaman

Tugas

- Kita akan ubah komponen `DataLayout` agar di bagian atas terdapat menu menuju semua halaman data

Kode : src/DataLayout.jsx

```
DataLayout.jsx ×  
1 import {Outlet} from "react-router";  
2  
3 export default function DataLayout() {  
4   return (  
5     <>  
6       <div>  
7         <h1>This is Header</h1>  
8       </div>  
9       <div>  
10        <ul>  
11          <li><a href="/data/products">Products</a></li>  
12          <li><a href="/data/sellers">Sellers</a></li>  
13          <li><a href="/data/customers">Customer</a></li>  
14        </ul>  
    )  
  }  
}
```


Link Element

- React Router menyediakan komponen Link
- Komponen Link ini digunakan sebagai pengganti Anchor Element
- Berbeda dengan Anchor Element, komponen Link ini tidak akan melakukan reload halaman
- https://api.reactrouter.com/v7/functions/react_router.Link.html

Kode : src/DataLayout.jsx

DataLayout.jsx ×

```
1 import {Link, Outlet} from "react-router";
2
3 export default function DataLayout() {
4   return (
5     <>
6       <div>
7         <h1>This is Header</h1>
8       </div>
9       <div>
10        <ul>
11          <li><Link to="/data/products">Products</Link></li>
12          <li><Link to="/data/sellers">Sellers</Link></li>
13          <li><Link to="/data/customers">Customers</Link></li>
14        </ul>
15      </div>
```

Link To

- Attribute to di komponen Link bisa menerima parameter Object, dimana terdapat attribute pathname, search dan hash
- Attribute pathname untuk lokasi path
- Attribute search untuk query parameter
- Attribute hash untuk #fragmen

Kode : src/DataLayout.jsx

```
8      </div>
9      <div>
10         <ul>
11             <li><Link to="/data/products">Products</Link></li>
12             <li><Link to="/data/sellers">Sellers</Link></li>
13             <li><Link to="/data/customers">Customers</Link></li>
14             <li><Link to={{
15                 pathname: "/data/products",
16                 search: "?category=shoes",
17                 hash: "#top"
18             }}>Products</Link></li>
19         </ul>
20     </div>
21     <div>
```

NavLink

- Salah satu yang biasa kita lakukan ketika membuat Link adalah, membedakan style Link yang sedang aktif dan yang tidak aktif
- Biasanya, kita lakukan pengecekan, jika URL path yang sedang dikunjungi sama dengan tujuan Link, maka kita ubah style nya menjadi aktif
- React Router menyediakan komponen khusus untuk mempermudah hal ini, cara penggunaannya sama seperti komponen Link, namun bedanya jika URL path saat ini sama dengan tujuan dari NavLink, maka secara otomatis NavLink akan menggunakan style `a.active`
- https://api.reactrouter.com/v7/functions/react_router.NavLink.html

Kode : src/data.css

 data.css ×

```
1 a.active {  
2   color: red;  
3 }  
4 
```

Kode : src/DataLayout.jsx

DataLayout.jsx ×

```
1 import {NavLink, Outlet} from "react-router";
2 import "./data.css";
3
4 export default function DataLayout() {
5   return (
6     <>
7       <div>
8         <h1>This is Header</h1>
9       </div>
10      <div>
11        <ul>
12          <li><NavLink to="/data/products">Products</NavLink></li>
13          <li><NavLink to="/data/sellers">Sellers</NavLink></li>
14          <li><NavLink to="/data/customers">Customers</NavLink></li>
15          <li><NavLink to={{
16            pathname: "/data/products",
17            search: "?category=shoes",
18            hash: "#top"
19          }}>Products</NavLink></li>
```

Use Navigate

Use Navigate

- Pada kasus tertentu, mungkin kita ingin melakukan navigasi dari satu halaman ke halaman lain menggunakan JavaScript
- Untuk melakukan ini, React Router menyediakan Hooks `useNavigate()`
- https://api.reactrouter.com/v7/functions/react_router.useNavigate.html
- `useNavigate()` akan mengembalikan function yang bisa kita gunakan untuk berpindah ke halaman lain
- https://api.reactrouter.com/v7/interfaces/react_router.NavigateFunction.html
- Kita bisa gunakan parameter `Path`
- https://api.reactrouter.com/v7/interfaces/react_router.Path.html
- Atau `number` (untuk maju/mundur)

Tugas

- Kita akan tambahkan tombol di halaman Home untuk menuju halaman Data

Kode : src/Home.jsx

Home.jsx ×

```
1 import {useNavigate} from "react-router";
2
3 export default function Home() {
4
5     const navigate = useNavigate();
6
7     function handleClick() {
8         navigate({
9             pathname: "/data",
10         });
11     }
12
13     return (
14         <div>
15             <h1>Home</h1>
16             <p>Welcome to the home page!</p>
17             <button onClick={handleClick}>Go to Data</button>
18         </div>
```

Use Search Param

Use Search Param

- Saat menggunakan URL, kadang kita akan memanfaatkan Query Parameter untuk mengirim data
- React Router menyediakan Hook `useSearchParams()` yang bisa kita gunakan untuk mendapatkan data Query Parameter
- https://api.reactrouter.com/v7/functions/react_router.useSearchParams.html
- `useSearchParams()` akan mengembalikan Array yang berisi object `URLSearchParams` dan setter function nya
- <https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams>

Tugas

- Kita akan coba membuat halaman pencarian Product, data pencariannya akan kita kirim lewat Query Parameter
- Kita akan buat komponen untuk halaman `/data/products/search`

Kode : src/ProductSearch.jsx (1)

ProductSearch.jsx ×

```
1  import {useNavigate, useSearchParams} from "react-router";
2  import {useState} from "react";
3
4  export default function ProductSearch() {
5    const [searchParams] = useSearchParams();
6    const navigate = useNavigate();
7
8    const [search, setSearch] = useState(searchParams.get("search") || "");
9
10   function handleSearch() {
11     if (search) {
12       navigate({
13         pathname: "/data/products/search",
14         search: `?search=${search}`
15       });
16     }
17   }
18 }
```

Kode : src/ProductSearch.jsx (2)

```
return (  
  <>  
    <h1>Search Product</h1>  
    <input type="text" value={search} onChange={e => setSearch(e.target.value)}>  
    <button onClick={handleSearch}>Search</button>  
    <p>  
      Kamu mencari: {searchParams.get("search")}  
    </p>  
  </>  
)
```



Kode : src/main.jsx

```
13 import NotFound from "../NotFound.jsx";
14 import ProductSearch from "../ProductSearch.jsx";
15
16 createRoot(document.getElementById('root')).render(
17   <StrictMode>
18     <BrowserRouter>
19       <Routes>
20         <Route index element={<Home/>}/>
21         <Route path="/about" element={<About/>}/>
22         <Route path="/images/*" element={<Image/>}/>
23         <Route path="/data" element={<DataLayout/>}>
24           <Route index element={<Data/>}/>
25           <Route path="products" element={<Product/>}/>
26           <Route path="products/search" element={<ProductSearch/>}/>
27           <Route path="products/:id" element={<ProductDetail/>}/>
28           <Route path="customers" element={<Customer/>}/>
```

Use Location

Use Location

- Untuk mendapatkan URL saat ini, sebenarnya kita bisa menggunakan `window.location.href`
- Tapi React Router menyediakan Hook `useLocation()`, yang bisa digunakan untuk mendapatkan lokasi path saat ini
- https://api.reactrouter.com/v7/functions/react_router.useLocation.html
- Hasil return dari `useLocation()` adalah object `Location`, sehingga kita bisa dapat informasi lengkap dari location, seperti path, query sampai ke `#fragment`
- https://api.reactrouter.com/v7/interfaces/react_router.Location.html

Tugas

- Kita akan coba tambahkan di footer komponen `DataLayout`, informasi location saat ini
- Sehingga setiap berpindah lokasi, kita bisa melihat lokasinya di footer

Kode : src/DataLayout.jsx

DataLayout.jsx ×

```
1 import {NavLink, Outlet, useLocation} from "react-router";
2 import "./data.css";
3
4 export default function DataLayout() {
5   const location = useLocation();
6
7   return (
8     <>
9       <div>
10         <h1>This is Header</h1>
11       </div>
12       <div...>
13         <div>
14           <Outlet/>
15         </div>
16         <div>
17           <p>This is footer</p>
18           <p>Location : {location.pathname}{location.search}{location.hash}</p>
19         </div>
20       </div>
21     </>
22   );
23 }
```

Materi Selanjutnya

Materi Selanjutnya

- ReactJS Redux