COACHING PROGRAM Week 4
React.js

Bagian 3: Routing

Pengertian Routing

Routing dalam pengembangan aplikasi web adalah proses yang mengatur navigasi dan perpindahan antara berbagai halaman atau tampilan tanpa perlu memuat ulang seluruh halaman web. Dalam pengembangan web dengan React, Anda menggunakan library seperti **react-router** untuk mengelola rute-rute yang menghubungkan URL dengan komponen React tertentu. Misalnya, ketika pengguna mengakses URL "/about," React Router memastikan bahwa komponen "About" dimuat dan ditampilkan. Ini menciptakan pengalaman pengguna yang responsif dan interaktif, memungkinkan pengguna berpindah antara berbagai konten dalam aplikasi dengan mudah.

Instalasi React-Router

Library **react-router** memiliki beberapa ekstensi, salah satunya react-douter-dom yang dirancang khusus untuk kebutuhan aplikasi React yang berjalan di web browser. Instalasi react-router-dom cukup simpel, cukup menjalankan perintah `npm i react-router-dom` melalui terminal pada folder yang sudah terinstall project react (Vite.js yang ada pada materi pre-day 2).

```
(base) PS C:\Users\Fanes\test\UKM Programming> npm i react-router-dom

added 3 packages, and audited 272 packages in 2s

97 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Pada file **app.jsx**, kita dapat melakukan setup sebagai menu untuk melakukan routing, sehingga komponen dipisahkan ke file yang berbeda, dan dipanggil seperti ini:

app.jsx

```jsx
import { BrowserRouter, Route, Routes } from "react-router-dom"
import Home from './pages/Home'

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path='/' element={<Home />} />
      </Routes>
    </BrowserRouter>
  )
}
export default App
```

code

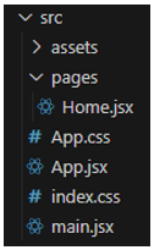
COACHING PROGRAM Week 4
React.js

home.jsx

```jsx
const Home = () => {

  return (
    <div>
      Ini homepage
    </div>
  )
}
export default Home
```

Struktur folder src:



Dari kode diatas, file app.jsx di setup sebagai menu utama untuk dasar dari routing. Diawali dengan tag <BrowserRouter> dengan children <Routes>, dan diikuti dengan <Route> sebagai list route apa aja yang ingin kalian buat.

Pada tag Route, terdapat property path dan element:

Path: property **path='/'** memiliki arti bahwa komponen akan di panggil jika mengakses root page (misal http://localhost:5173/). Jika ingin merubah seperti **path='/home'**, maka komponen akan di panggil dengan cara mengakses http://localhost:5173/home.

Element: property **element={<Home />}** memiliki arti bahwa saat route di akses, maka komponen inilah yang akan ditampilkan kepada user. Komponen **<Home />** berasal dari file **Home.jsx** yang berisi tag div dengan tulisan “Ini homepage”, sehingga saat mengakses route ini, akan menampilkan tulisan tersebut. Karena komponen Home ini terletak pada file yang berbeda, dibutuhkannyalah kode untuk melakukan import `import Home from './pages/Home'`. Yang artinya melakukan import terhadap komponen **Home** yang berasal dari direktori /pages/Home.jsx.

Kalian bisa menerapkan materi ini untuk membuat file Pokemon.jsx yang berisikan kode pada pertemuan sebelumnya terkait fetching data dari API.

Bagian 4: Advance API Filtering

API yang kompleks, biasanya memiliki banyak array/object yang bersarang (nested array/object). Untuk mendapatkan data yang diinginkan, biasanya developer menggunakan berbagai macam sintaks-sintaks dari javascript untuk memilah data yang diperoleh dari API tersebut.

Array Filtering

Ada beberapa sintaks yang digunakan untuk melakukan fitering terhadap array, seperti:

COACHING PROGRAM Week 4
React.js

Array.filter()

```
const fruits = ["apple", "banana", "cherry", "avocado", "pear"]

console.log(
  fruits.filter(fruit => fruit.length == 4)
); // output: ["pear"]

console.log(
  fruits.filter(fruit => fruit.includes("c"))
); // output: ["cherry", "avocado"]

console.log(
  fruits.filter(fruit => fruit.endsWith("e"))
); // output: ["apple"]

console.log(
  fruits.filter(fruit => !fruit.startsWith("a"))
); // output: ["banana", "cherry", "pear"]
```

Array filter bertugas melakukan filterasi data terhadap array yang ingin dikelola. Contoh diatas salah satunya `fruits.filter(fruit => fruit.length == 4)`. Disini akan mengembalikan nilai buah dengan kata sebanyak 4 huruf, yaitu “pear”.

Array.find()

```
const fruits = ["apple", "banana", "cherry", "avocado", "pear"]

console.log(
  fruits.find(fruit => fruit.includes("c"))
); // output: "cherry"

console.log(
  fruits.find(fruit => fruit.startsWith("a"))
); // output: "apple"
```

Array find bertugas melakukan pencarian data terhadap array yang ingin dikelola. Contoh diatas yaitu `fruits.find(fruit => fruit.includes(“c”))`. Disini akan mengembalikan nilai cherry, karena buah cherry adalah buah yang memiliki huruf c didalamnya dan di posisi index terendah. Karena jika dilihat, kata “avocado” juga mengandung huruf c, namun tidak ikut menjadi output, karena fungsi **find** hanya mengembalikan 1 element dengan index terendah.

Selain dua filterasi diatas, ada beberapa sintaks yang lumayan sering digunakan dalam mengelola sebuah array, seperti pop(), shift(), splice(), slice(), map(), dll yang bisa kalian eksplorasi sendiri.

COACHING PROGRAM Week 4
React.js

Array of Object Filtering

Ada beberapa sintaks yang digunakan untuk melakukan fitering terhadap object, berikut adalah inisialisasi object yang akan digunakan sebagai demo:

```
const freshMart = {
  name: "Fresh Mart",
  products: [
    {
      name: "Apple",
      price: 5000,
      stock: 10
    },
    {
      name: "Orange",
      price: 3000,
      stock: 10
    },
    {
      name: "Banana",
      price: 4000,
      stock: 10
    },
    {
      name: "Mango",
      price: 7000,
      stock: 10
    },
  ]
}
```

Object.filter()

```
console.log(
  freshMart.products.filter(
    product => product.price > 5000
  )
); // output: [ { name: 'Mango', price: 7000, stock: 10 } ]

console.log(
  freshMart.products.filter(
    product => product.name == "Apple"
  )
); // output: [ { name: 'Apple', price: 5000, stock: 10 } ]
```

Mirip dengan filterasi terhadap array, fungsi filter() pada object juga dapat melakukan filter data terhadap property setiap product diatas. Bedanya dengan array filtering, object filtering lebih spesifik mengakses property seperti price, name, dll.

Object.find()

Konsepnya mirip dengan Array.find() sebelumnya, namun seperti yang sudah dijelaskan pada Object.filter() diatas, terdapat beberapa perbedaan dengan mengharuskan untuk mengakses property tiap data.

COACHING PROGRAM Week 4
React.js

Object.map()

```
console.log(
  freshMart.products.map(product => product.name)
); // ["Apple", "Orange", "Banana", "Mango"]

console.log(
  freshMart.products.map(product => product.price)
); // [5000, 3000, 4000, 7000]
```

Fungsi map() adalah fungsi yang lumayan sering digunakan dalam project react.js. Biasanya digunakan untuk menampilkan banyak data seperti list product sehingga code tidak ditulis secara berulang. Contoh:

```
freshMart.products.map(product => (
  <div className="myCard">
    <div>Name: {product.name}</div>
    <div>Price: {product.price}</div>
    <div>Stock: {product.stock}</div>
  </div>
))
```

Jadi komponen card diatas hanya ditulis 1x dan dapat menampilkan banyak card berdasarkan banyak data products.

Kesimpulan

Selain filtering diatas, ada beberapa fungsi yang dapat digunakan dalam mengelola suatu object walau hanya untuk case-case tertentu. Kalian dapat eksplorasi mandiri untuk fungsi-fungsi Object.keys(), Object.value(), Object.entries(), Object.hasOwnProperty(), dll.