

Webinar React.js

Outline

Introduction to ReactJS

Create a New ReactJS Project with Vite

JSX

Functional Component

Props and State

React

sebuah library JavaScript yang dibuat oleh Facebook untuk membangun user interface.

Dengan menggunakan ReactJS, kita dapat membuat aplikasi web menggunakan **component** dan dapat digunakan kembali di berbagai tempat.

Why React JS

- Menggunakan **Virtual DOM**, sehingga proses rendering menjadi lebih cepat.
- Menggunakan **JSX**, sehingga dapat menulis kode HTML di dalam JavaScript.
- Menggunakan konsep **component**, sehingga dapat direuse
- Menggunakan konsep state dan props, memungkinkan untuk mengatur data secara dinamis.

Getting started

- `npm create vite@latest my-app -- --template react`
- `cd my-app`
- `npm install`
- `npm run dev`

JSX

JavaScript XML

JSX adalah sebuah sintaksis yang ditambahkan pada JavaScript. JSX memungkinkan kita untuk menulis kode HTML di dalam JavaScript.

```
function Component()
{
  return(
    <div>
      <h3>Hello World</h3>
      <ul>
        <li>
          <a href="#">Hello World</a>
        </li>
      </ul>
    </div>
  );
}
```

JSX Rules

Return hanya satu elemen

Jika kita ingin mengembalikan lebih dari satu elemen, maka kita harus mengelompokkannya dalam satu elemen. Contohnya seperti ini:

```
function Component()
{
  return(
    <div>
      <h3>Hello World</h3>
      <ul>
        <li>
          <a href="#">Hello World</a>
        </li>
      </ul>
    </div>
  );
}
```

```
function Component()
{
  return(
    <div>
      <h3>Hello World</h3>
      <ul>
        <li>
          <a href="#">Hello World</a>
        </li>
      </ul>
    </div>
    <div></div>
  );
}
```

Use camelCase for HTML Attribute

Jika kita ingin menambahkan atribut pada elemen, maka kita harus menuliskannya dengan format camelCase.

```
<button onClick={alert("Button Clicked!")}>Click Me!</button>
```

className Instead of Class

Jika kita ingin menambahkan class pada elemen, maka kita harus menggunakan atribut className.

```
<button className="btn btn-primary">Click Me!</button>
```

Close Every Element

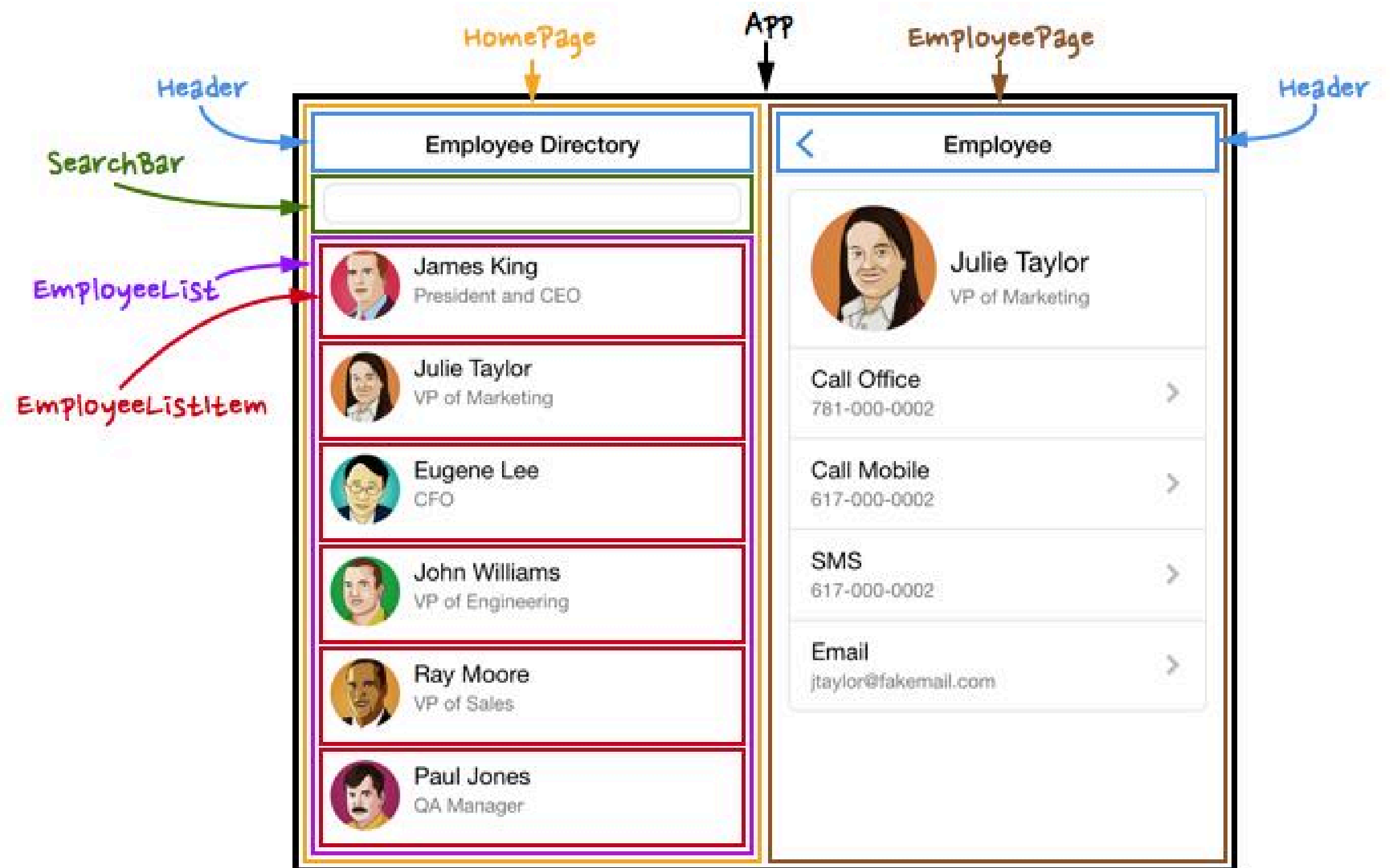
In HTML almost all the tags have starting and closing tags, rather than a few e.g. ``, `<input>`, `
`.

These few tags have no closing tags. We use it simply as the way they are defined in the example above. In React JSX every tag must be close even those which have no closing tags e.g.

```
<img src = " " alt = " " />
```

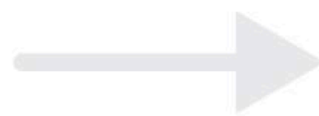
Component

Jika kita ingin membuat sebuah website atau aplikasi, maka kita bisa membaginya menjadi beberapa bagian kecil. Setiap bagian kecil tersebut disebut dengan component

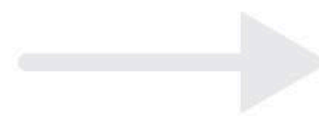




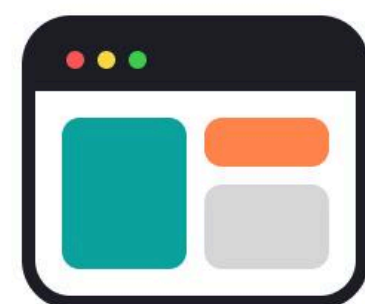
Atoms



Molecules



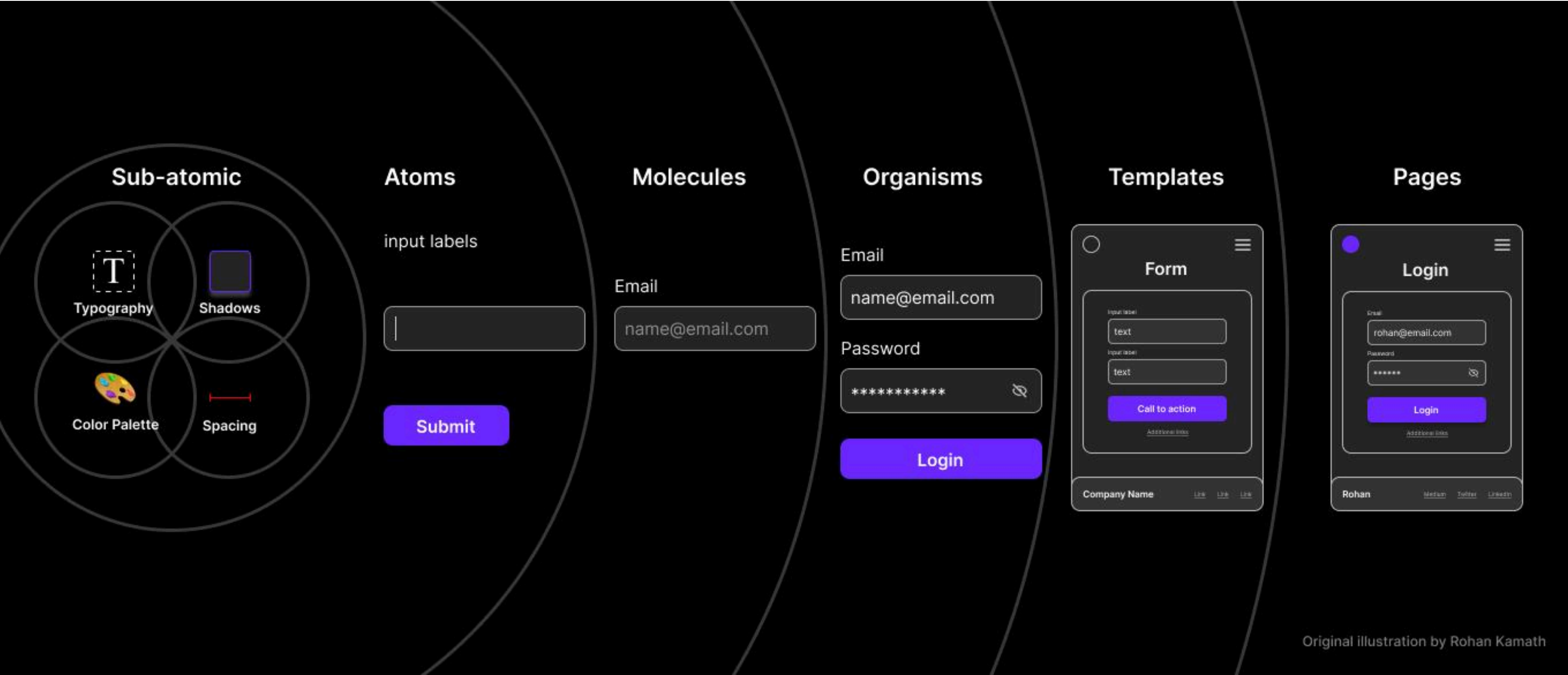
Organisms



Pages



Templates



Props

Props

Props adalah sebuah object yang berisi data yang akan digunakan oleh component. Props ini akan di kirimkan ke component melalui parameter.

```
function Example(props) {  
  return <h1>Hello {props.name}</h1>;  
}  
  
function App() {  
  return <Example name="John Doe" />;  
}
```

Why use props?

Props ini berguna untuk membuat component reusable. Jika kita ingin membuat sebuah component yang bisa digunakan di berbagai bagian website atau aplikasi, maka kita bisa membuat props untuk menerima data dari luar.

```
function Example(props) {  
  return <h1>Hello {props.name}</h1>;  
}  
  
function App() {  
  return <Example name="John Doe" />;  
}
```

Event Handling in ReactJS

Event di React adalah interaksi pengguna dengan elemen, misalnya klik tombol atau ketik di input. React pakai sintaks `onClick`, `onChange`, dll., buat handle event

Event handling itu kayak bel pintu. Kalau ada tamu datang (klik bel), bel bakal bunyi (aksi).

Kamu tinggal bikin fungsi “apa yang terjadi kalau bel dipencet.”

```
<button onClick={handleClick}>Klik Saya</button>
```



List and Keys

List di React dipakai buat menampilkan data yang banyak, contohnya daftar produk atau daftar nama.

```
const products = [  
  {  
    id: 1,  
    name: "Apple",  
    price: 10000,  
  },  
  {  
    id: 2,  
    name: "Orange",  
    price: 5000,  
  },  
  {  
    id: 3,  
    name: "Banana",  
    price: 3000,  
  },  
  {  
    id: 4,  
    name: "Mango",  
    price: 20000,  
  },  
];
```

```
const users = [  
  {  
    id: 1,  
    name: "John Doe",  
    email: "johndoe@mail.com",  
  },  
  {  
    id: 2,  
    name: "Jane Doe",  
    email: "janedoe@mail.com",  
  },  
  {  
    id: 3,  
    name: "Bob Doe",  
    email: "bobdoe@mail.com",  
  },  
  {  
    id: 4,  
    name: "Alice Doe",  
    email: "alicedoe@mail.com",  
  },  
];
```

sekumpulan data yang biasanya dalam bentuk array. Array ini bisa saja didapat dari data yang diambil dari API, atau bisa juga dari data yang kita buat sendiri.

Render List in ReactJS

Untuk mengelola list data, kita bisa menggunakan function **map()** pada array.

function map() ini akan mengembalikan array baru yang sudah diubah sesuai dengan function yang kita buat.

Render List in ReactJS

```
const users = [  
  {  
    id: 1,  
    name: "John Doe",  
    email: "johndoe@mail.com",  
  },  
  {  
    id: 2,  
    name: "Jane Doe",  
    email: "janedoe@mail.com",  
  },  
  {  
    id: 3,  
    name: "Bob Doe",  
    email: "bobdoe@mail.com",  
  },  
  {  
    id: 4,  
    name: "Alice Doe",  
    email: "alicedoe@mail.com",  
  },  
];
```

```
const UserList = () => {  
  return (  
    <div>  
      {users.map((user) => (  
        <div key={user.id}>  
          <h3>{user.name}</h3>  
          <p>{user.email}</p>  
        </div>  
      ))}  
    </div>  
  );  
};
```


Data Fetching

Data fetching digunakan untuk ambil data dari API atau server. React biasanya pakai `useEffect` buat ambil data ketika komponen pertama kali di-load.

React router dom

package npm yang digunakan untuk membuat sebuah single-page application (SPA), atau aplikasi yang memiliki banyak halaman namun perpindahan halamannya tidak akan melakukan refresh (reload browser)

Route

merupakan sebuah komponen yang digunakan untuk **mendefinisikan** dan **merender komponen berdasarkan path** yang ditentukan.

Installation & Configuration

```
// menggunakan pnpm  
$ pnpm install react-router-dom  
  
// menggunakan yarn  
$ yarn add react-router-dom
```

```
npm i react-router-dom
```


<BrowserRouter />, <Routes />, <Route />, <Link />, dan <Outlet />

Untuk menyiapkan router,

App perlu dibungkus **BrowserRouter** di index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import { BrowserRouter } from "react-router-dom";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```



Define Routes

define routes di **App.js**

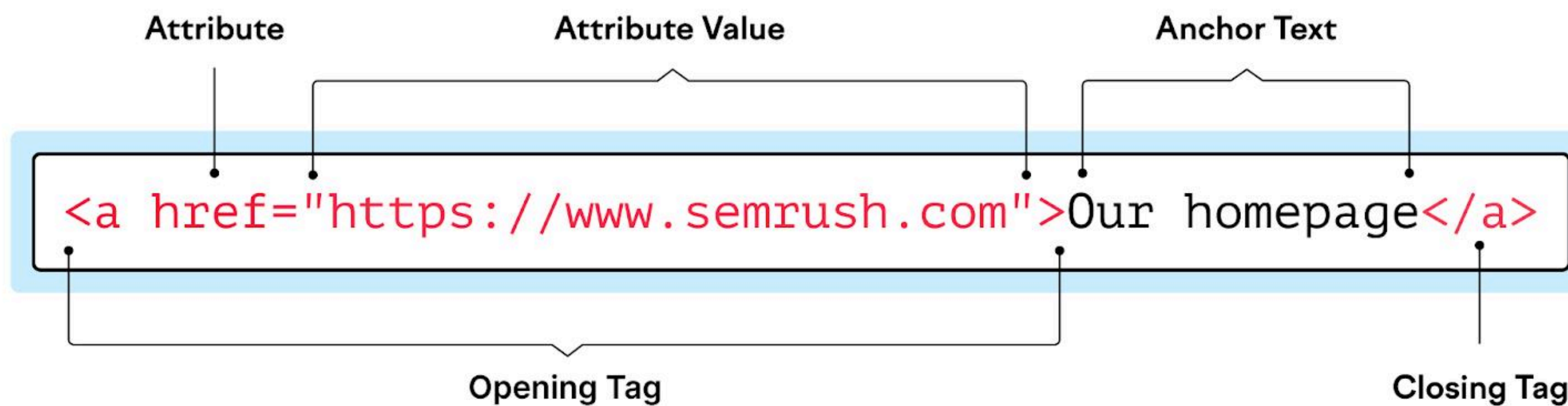
menentukan element mana yang akan selalu di-render, mana yang akan dynamically berubah sesuai Route yang diakses

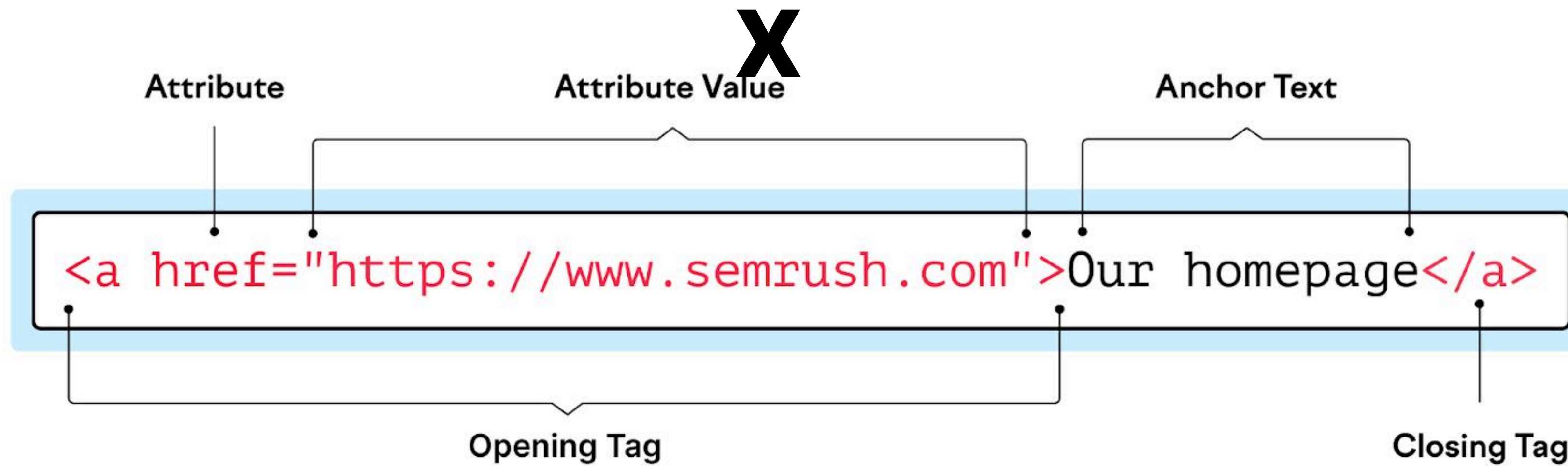
```
// src/App.js
import React from "react";
import { Routes, Route } from "react-router-dom";
import "./App.css";

function App() {
  return (
    <div className="App">
      <h1>Welcome to Ruangguru!</h1>
      <Routes>
        <Route path="/" element={<div>Home</div>} />
        <Route path="about" element={<div>About</div>} />
      </Routes>
    </div>
  );
}
```



Navigasi antar routes





```
function Home(props) {  
  return (  
    <div className="container">  
      <h1>Home Page</h1>  
      <h3>  
        <Link className="btn btn-primary" to="/profile">  
          Profile  
        </Link>  
      </h3>  
    </div>  
  );  
}  
  
export default Home;
```

