

Materi Pelatihan ReactJs

Pertemuan 1-5
LNSW 2023

ReactJs

- ReactJS adalah sebuah library JavaScript untuk membangun antarmuka yang dikembangkan oleh Facebook.
- ReactJs fokus pada pembuatan-pembuatan komponen UI
- Menggunakan Virtual-DOM yang membuat UI menjadi lebih efisien dan cepat

Kenapa ReactJs Populer?

- Tidak lebih kompleks dari beberapa alternatif (Angular & EmberJs)
- Dikembangkan oleh Facebook, sehingga menjadi sebuah ReactJs menjadi library yang mungkin baik untuk kedepannya

Mudah untuk dipelajari?

Meskipun dikenal tidak lebih kompleks dari beberapa library yang lainnya, tidak menjadikan ReactJs mudah, jika digabungkan menggunakan Redux dll.

Namun, React sendiri hanyalah struktur API yang sangat simple. Pada dasarnya Anda hanya perlu mengenal dan mengetahui beberapa konsep untuk memulainya. (Components, JSX, State dan Props)

Reactjs === HTML?

Tidak, ReactJS dan HTML bukanlah hal yang sama.
ReactJs adalah sebuah library JavaScript, kedua teknologi ini dapat digunakan Bersama-sama untuk membuat web yang interaktif.

Dengan memanfaatkan syntax HTML yang disebut JSX. JSX memungkinkan Anda menulis komponen React seolah-olah mereka adalah elemen HTML

ReactJs Installation & Configurations




- Node ([Click to install](#))
- React Library menggunakan create-react-app
 - npm install -g create-react-app
 - Npx create-react-app my-app
- Atau dapat menggunakan editor pada browser pada [StackBlitz](#)

HTML FlashBack

- Perbedaan utama antara HTML dan React adalah React menggunakan JavaScript untuk membuat komponen-komponen seperti tag HTML, sehingga lebih dinamis dan memiliki lebih banyak fitur dibandingkan HTML biasa. Namun, syntax-nya seperti HTML, sehingga mudah dipahami bagi yang sudah familiar dengan HTML.

HTML:


css

 Copy code

```
<h1>Hello World</h1>
<p>Ini adalah paragraf HTML</p>
```

React:

javascript

 Copy code

```
import React from 'react';

function App() {
  return (
    <div>
      <h1>Hello World</h1>
      <p>Ini adalah paragraf React</p>
    </div>
  );
}

export default App;
```

Component

- Terdapat 2 cara untuk mendefinisikan komponen pada React Js yaitu menggunakan Class dan Function Components

```
javascript Copy code  
  
import React, { Component } from 'react';  
  
class Example extends Component {  
  constructor(props) {  
    super(props);  
    this.state = {  
      message: 'Hello World'  
    };  
  }  
  
  render() {  
    return (  
      <div>  
        <h1>{this.state.message}</h1>  
      </div>  
    );  
  }  
}  
  
export default Example;
```

```
javascript Copy code  
  
import React from 'react';  
  
const Example = (props) => {  
  return (  
    <div>  
      <h1>Hello World</h1>  
    </div>  
  );  
};  
  
export default Example;
```

What is Component?

- Class Komponen adalah sebuah komponen yang dibuat dengan menggunakan syntax class dalam React. Class component memiliki akses ke metode seperti constructor, render, dan lifecycle method seperti componentDidMount. Class Komponen harus memiliki metode render yang akan menentukan apa yang akan ditampilkan pada layar.
- Sedangkan Function Komponen adalah sebuah Komponen yang dibuat menggunakan sebuah function. Function Komponen tidak memiliki akses ke metode seperti constructor dan lifecycle method, namun Function Komponen bisa mengakses props dan state melalui parameter function

Conditional Component

- Kondisional komponen pada ReactJS adalah komponen yang membuat render komponen ditentukan oleh kondisi tertentu. Ini memungkinkan aplikasi ReactJS untuk membuat render komponen hanya jika kondisi tertentu terpenuhi

Example Conditional Component

Contoh 1

```
app.tsx •
1  import * as React from 'react';
2
3  const Example = () => {
4    let kondisi = 1
5    return (
6      <div>
7        {
8          kondisi === 1 ? (
9            <p><button>Button 1</button></p>
10           ) : kondisi === 2 ? (
11             <p><button>Button 2</button></p>
12           ) : <div>Tidak ada yang sesuai</div>
13         }
14       </div>
15     )
16   }
17
18   export default Example;
```

Contoh 2

```
1  import * as React from 'react';
2
3  const Example = () => {
4    const kondisi = true
5
6    const Content = () => {
7      let content = ''
8      if (kondisi) {
9        content = 'Benar'
10      } else {
11        content = 'Salah'
12      }
13      return(
14        <button>Button {content}</button>
15      )
16    }
17    return (
18      <Content />
19    )
20  }
21  export default Example;
```

States & Props



State dan Props adalah dua konsep utama dalam React.js

- State adalah bagian dari komponen yang menyimpan data yang berubah-ubah dan mempengaruhi tampilan komponen. State digunakan untuk mengatur perilaku dan tampilan dari komponen React.
- Props adalah argumen atau properti yang diteruskan ke komponen dari komponen lain. Props memungkinkan komponen untuk menerima data dari komponen lain dan digunakan untuk mengontrol tampilan komponen

Contoh States & Props

States:

javascript

Copy code

```
class Counter extends React.Component {
  state = {
    count: 0
  };

  increment = () => {
    this.setState({
      count: this.state.count + 1
    });
  };

  render() {
    return (
      <div>
        <p>Counter: {this.state.count}</p>
        <button onClick={this.increment}>Increment</button>
      </div>
    );
  }
}
```

Props:

javascript

Copy code

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

ReactDOM.render(
  <Welcome name="Sara" />,
  document.getElementById('root')
);
```

Event Handling

Event handling pada ReactJS adalah proses menangani aksi atau perubahan yang terjadi pada aplikasi ReactJS, seperti klik tombol, submit form, dan lain-lain. Event handling bertujuan untuk menjalankan suatu aksi atau memproses suatu data setiap kali terjadi perubahan pada elemen

```
javascript Copy code

import React, { useState } from 'react';

const Example = () => {
  const [count, setCount] = useState(0);

  const handleClick = () => {
    console.log(count);
    setCount(count + 1);
  };

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={handleClick}>
        Increment
      </button>
    </div>
  );
};

export default Example;
```

Event Handling. (onFocus, onBlur, & onChange)



onFocus, onBlur, & onChange adalah bagian dari event handling pada Reactjs

- onFocus: ini adalah event yang terjadi saat sebuah elemen memperoleh fokus (seperti saat kursor berpindah ke dalam sebuah form input).
- onBlur: ini adalah event yang terjadi saat sebuah elemen kehilangan fokus (seperti saat kursor berpindah dari sebuah form input).
- onChange: ini adalah event yang terjadi saat isi dari sebuah elemen berubah

Example Event Handling

```
App.tsx •  
1  import * as React from 'react';  
2  import './style.css';  
3  import * as Yup from 'yup';  
4  
5  export default function App() {  
6    return (  
7      <div>  
8        <h1>Hello StackBlitz!</h1>  
9        <input onFocus={() => console.log('onFocus')}></input>  
10       <input onBlur={() => console.log('onBlur')}></input>  
11       <input onChange={() => console.log('onChange')}></input>  
12     </div>  
13   );  
14 }  
15
```

Form Input & Validation



Dalam ReactJS, form input dapat dibuat dengan menggunakan tag HTML seperti input, textarea, dan select. Validasi dapat dilakukan dengan memanfaatkan state dan props.

Untuk melakukan validasi form input di ReactJS, kita dapat membuat logika validasi pada event handling seperti onSubmit atau onChange.

Example: Form & Validation

```
App.tsx •
1  import * as React from 'react';
2  import './style.css';
3  import { BrowserRouter, Route, Link, Routes } from 'react-router-dom';
4
5  function SignupForm() {
6    const [email, setEmail] = React.useState("");
7    const [error, setError] = React.useState("");
8
9    const handleSubmit = (e) => {
10     e.preventDefault();
11     if (!email.includes("@")) {
12       setError("Invalid email address");
13     } else {
14       setError("");
15       // handle success submission
16     }
17   };
18   return (
19     <form onSubmit={handleSubmit}>
20       <input
21         type="email"
22         placeholder="Enter email"
23         value={email}
24         onChange={e => setEmail(e.target.value)}
25       />
26       <button type="submit">Submit</button>
27     </form>
28   );
29 }
30
31 export default SignupForm;
```

React Router



React Router adalah library JavaScript untuk membantu membuat routing di aplikasi React. Routing memungkinkan Anda membuat halaman yang berbeda dan membuat perpindahan antar halaman dengan mudah. React Router membantu menangani perpindahan antar halaman tanpa memuat ulang halaman seluruhnya, sehingga memberikan pengalaman yang lebih halus dan cepat bagi pengguna. Library ini sangat populer dan menjadi standar untuk routing di aplikasi React

Example: React Router

App.tsx

```
1 import * as React from 'react';
2 import './style.css';
3 import { BrowserRouter, Route, Link, Routes } from 'react-router-dom';
4
5 function Home() {
6   return <h2>Home</h2>;
7 }
8
9 function About() {
10   return (
11     <div>
12       <h2>About</h2>
13     </div>
14   );
15 }
16
```

```
17 function App() {
18   return (
19     <BrowserRouter>
20       <nav>
21         <ul>
22           <li>
23             <Link to="/">Home</Link>
24           </li>
25           <li>
26             <Link to="/about">About</Link>
27           </li>
28         </ul>
29       </nav>
30
31       <Routes>
32         <Route path="/" exact element={<Home />} />
33         <Route path="/about" element={<About />} />
34       </Routes>
35     </BrowserRouter>
36   );
37 }
38
39 export default App;
```

React Hooks



React Hook adalah fitur baru dalam React yang memungkinkan pengembangan aplikasi React tanpa menggunakan class component. Hook memungkinkan Anda membagikan state dan logika antara komponen. Ada beberapa jenis hook seperti useState, useEffect, useContext, useReducer, dll

Example: React Hook

javascript

 Copy code

```
import React, { useState } from 'react';

function Example() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

useReducer

React Hook adalah fitur baru dalam React yang memungkinkan pengembangan aplikasi React tanpa menggunakan class component. Hook memungkinkan Anda membagikan state dan logika antara komponen. Ada beberapa jenis hook seperti useState, useEffect, useContext, useReducer, dll

```
import React, { useReducer } from 'react';

const initialState = {
  count: 0
};

function reducer(state, action) {
  switch (action.type) {
    case 'increment':
      return { count: state.count + 1 };
    case 'decrement':
      return { count: state.count - 1 };
    default:
      throw new Error();
  }
}
```

```
function Counter() {
  const [state, dispatch] = useReducer(reducer, initialState);

  return (
    <div>
      Count: {state.count}
      <button onClick={() => dispatch({ type: 'increment' })}>+</button>
      <button onClick={() => dispatch({ type: 'decrement' })}>-</button>
    </div>
  );
}

export default Counter;
```

useEffect

UseEffect adalah sebuah hook React yang memungkinkan kita menjalankan efek bersampingan seperti memuat data atau memperbarui DOM, setelah render komponen. useEffect memungkinkan kita untuk mengikat efek pada suatu komponen dan memastikan bahwa efek tersebut dijalankan setiap kali komponen tersebut dirender. useEffect sangat berguna untuk memastikan aplikasi kita memperbarui dengan benar, misalnya memuat data atau mengubah state ketika sebuah event terjadi.

Example: useEffect

javascript

 Copy code

```
import React, { useState, useEffect } from 'react';

function Example() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    document.title = `You clicked ${count} times`;
  });

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}

export default Example;
```


Fetch data with Axios and Hook

Fetching data API adalah proses pengambilan data dari suatu sumber data melalui API (Application Programming Interface) oleh aplikasi. Dalam hal ini, ReactJS dapat digunakan untuk mengambil data dari API dan menampilkannya dalam aplikasi. Proses ini bisa dilakukan dengan bantuan library seperti Axios atau Fetch API. Kode yang ditulis untuk mengambil data dari API harus berjalan pada lifecycle method atau menggunakan React Hook seperti useEffect. Setelah data berhasil diambil, data tersebut dapat diolah dan ditampilkan dalam bentuk yang diinginkan

Instalasi:

- Npm -i axios@0.24.0

GET, POST, PUT, and DELETE

- GET adalah salah satu method dalam HTTP (HyperText Transfer Protocol) yang digunakan untuk mendapatkan / mengambil data ke server..
- POST adalah salah satu method dalam HTTP (HyperText Transfer Protocol) yang digunakan untuk mengirim data ke server. Dalam pemrograman web, POST sering digunakan untuk mengirim data ke server dan menyimpan data baru pada aplikasi.
- PUT adalah salah satu method yang digunakan untuk mengubah data yang telah tersimpan pada server / database.
- DELETE adalah salah satu method yang digunakan untuk menghapus data yang telah tersimpan pada server / database.

Example Fetching Data (GET)

```
1  import React, { useState, useEffect } from 'react';
2  import axios from 'axios';
3
4  function App() {
5    const [data, setData] = useState([]);
6
7    useEffect(() => {
8      const fetchData = async () => {
9        const result = await axios.get('https://jsonplaceholder.typicode.com/posts');
10       setData(result.data);
11     };
12
13     fetchData();
14   }, []);
15
16   return (
17     <ul>
18       {data.map(item => (
19         <li key={item.id}>
20           {item.title}
21         </li>
22       ))}
23     </ul>
24   );
25 }
26
27 export default App;
```

Example Fetching Data (POST)

```
1  import * as React from 'react';
2  import axios from "axios";
3
4  const App = () => {
5    const [post, setPost] = React.useState({
6      title: "",
7      body: "",
8    });
9
10   const handleChange = (e) => {
11     setPost({
12       ...post,
13       [e.target.name]: e.target.value,
14     });
15   };
16
17   const handleSubmit = (e) => {
18     e.preventDefault();
19     axios
20       .post("https://jsonplaceholder.typicode.com/posts", {
21         title: post.title,
22         body: post.body,
23       })
24       .then((res) => console.log(res))
25       .catch((err) => console.error(err));
26   };
27 }
```

```
28   return (
29     <form onSubmit={handleSubmit}>
30       <input
31         type="text"
32         name="title"
33         placeholder="Title"
34         value={post.title}
35         onChange={handleChange}
36       />
37       <textarea
38         name="body"
39         placeholder="Body"
40         value={post.body}
41         onChange={handleChange}
42       />
43       <button type="submit">Submit</button>
44     </form>
45   );
46 };
47
48 export default App;
```

Example Fetching Data (PUT)

```
1  import * as React from 'react';
2  import axios from 'axios';
3
4  const App = () => {
5    const [data, setData] = React.useState({});
6
7    const handleSubmit = (e) => {
8      e.preventDefault();
9      axios
10       .put(`https://jsonplaceholder.typicode.com/posts/1`, data)
11       .then((res) => {
12         console.log(res.data);
13       });
14    };
15
16    return (
17      <form onSubmit={handleSubmit}>
18        <input
19          type="text"
20          name="title"
21          onChange={(e) => setData({ ...data, title: e.target.value })}
22        />
23        <input
24          type="text"
25          name="body"
26          onChange={(e) => setData({ ...data, body: e.target.value })}
27        />
28        <button type="submit">Submit</button>
29      </form>
30    );
31  };
32
33  export default App;
```

Example Fetching Data (DELETE)

```
1  import * as React from 'react';
2  import axios from 'axios';
3
4  const Example = () => {
5    const handleDelete = (id) => {
6      axios.delete(`https://jsonplaceholder.typicode.com/posts/${id}`)
7        .then(res => {
8          console.log(res.data, 'berhasil dihapus');
9        })
10   }
11
12   return (
13     <button onClick={() => handleDelete(1)}>Delete</button>
14   )
15 }
16
17 export default Example;
```

React Admin Template (Core-UI)

Core UI React adalah salah satu library komponen UI (User Interface) untuk React.js. Ini menyediakan komponen UI yang siap pakai dan dapat dengan mudah digunakan untuk membangun aplikasi web React. Core UI React juga mencakup template halaman, form input, tabel, modal, ikon, dan banyak lagi. Hal ini mempermudah pengembangan aplikasi web dengan menghemat waktu dan usaha pengembang.

Unduh Core UI:

- [Core UI Free Admin Template](#)
- Ekstrak file zip
- lalu npm install pada directory project tersebut
- Npm start

Adding Page

- Buat halaman baru dengan membuat file baru, misalnya Page.js atau halaman lainnya yang sesuai dengan kebutuhan Anda.
- Impor komponen React pada file baru tersebut
- Buat class component atau function component yang mengandung kode HTML, CSS, dan JavaScript yang diperlukan untuk membuat halaman baru.
- Dalam file src/routes.js, tambahkan halaman baru dengan memasukkan kode berikut

```
{ path: '/dashboard', name: 'Dashboard', element: Dashboard },
```

- Lalu import dengan ReactLazy

```
const Dashboard = React.lazy(() => import('./views/dashboard/Dashboard'))
```


Using Charts

```
<CCard className="mb-4">
  <CCardHeader>Line Chart</CCardHeader>
  <CCardBody>
    <CChartLine
      data={{
        labels: ['January', 'February', 'March', 'April', 'May', 'June', 'July'],
        datasets: [
          {
            label: 'My First dataset',
            backgroundColor: 'rgba(220, 220, 220, 0.2)',
            borderColor: 'rgba(220, 220, 220, 1)',
            pointBackgroundColor: 'rgba(220, 220, 220, 1)',
            pointBorderColor: '#fff',
            data: [random(), random(), random(), random(), random(), random(), random()],
          },
          {
            label: 'My Second dataset',
            backgroundColor: 'rgba(151, 187, 205, 0.2)',
            borderColor: 'rgba(151, 187, 205, 1)',
            pointBackgroundColor: 'rgba(151, 187, 205, 1)',
            pointBorderColor: '#fff',
            data: [random(), random(), random(), random(), random(), random(), random()],
          },
        ],
      }}
    />
  </CCardBody>
</CCard>
```

Create Table

Berikut contoh membuat table dengan CoreUI table:

```
<DocsExample href="components/table">
  <CTable>
    <CTableHead>
      <CTableRow>
        <CTableHeaderCell scope="col">#</CTableHeaderCell>
        <CTableHeaderCell scope="col">Class</CTableHeaderCell>
        <CTableHeaderCell scope="col">Heading</CTableHeaderCell>
        <CTableHeaderCell scope="col">Heading</CTableHeaderCell>
      </CTableRow>
    </CTableHead>
    <CTableBody>
      <CTableRow>
        <CTableHeaderCell scope="row">1</CTableHeaderCell>
        <CTableDataCell>Mark</CTableDataCell>
        <CTableDataCell>Otto</CTableDataCell>
        <CTableDataCell>@mdo</CTableDataCell>
      </CTableRow>
      <CTableRow>
        <CTableHeaderCell scope="row">2</CTableHeaderCell>
        <CTableDataCell>Jacob</CTableDataCell>
        <CTableDataCell>Thornton</CTableDataCell>
        <CTableDataCell>@fat</CTableDataCell>
      </CTableRow>
      <CTableRow>
        <CTableHeaderCell scope="row">3</CTableHeaderCell>
        <CTableDataCell colSpan="2">Larry the Bird</CTableDataCell>
        <CTableDataCell>@twitter</CTableDataCell>
      </CTableRow>
    </CTableBody>
  </CTable>
</DocsExample>
```

```
<DocsExample href="components/table">
  <CTable>
    <CTableHead>
      <CTableRow>
        <CTableHeaderCell scope="col">#</CTableHeaderCell>
        <CTableHeaderCell scope="col">Class</CTableHeaderCell>
        <CTableHeaderCell scope="col">Heading</CTableHeaderCell>
        <CTableHeaderCell scope="col">Heading</CTableHeaderCell>
      </CTableRow>
    </CTableHead>
    <CTableBody>
      {data.map((el, index) => {
        return (
          <CTableRow key={index}>
            <CTableHeaderCell scope="row">{el.id}</CTableHeaderCell>
            <CTableDataCell>{el.names}</CTableDataCell>
            <CTableDataCell>{el.names}</CTableDataCell>
            <CTableDataCell>{el.names}</CTableDataCell>
            <CTableDataCell>{el.names}</CTableDataCell>
          </CTableRow>
        )
      })}
    </CTableBody>
  </CTable>
</DocsExample>
```

Build React & Apps Deployment

Ketika proyek sudah selesai dibuat atau dikembangkan proyek tersebut haruslah melewati proses Build, sehingga proyek tersebut menjadi aplikasi yang dapat dijalankan pada web server.

Perintah yang dibutuhkan untuk melakukan build proyek reactJs adalah dengan mengetikkan:

```
#npm run build
```

Proses ini akan mem-bundle seluruh kode sumber dan menyimpannya ke dalam folder dist, sehingga dapat digunakan untuk deployment. Kode dalam folder ini sudah dalam bentuk yang optimal dan siap untuk dipublikasikan.

- Menggunakan ES6 dan fitur-fitur terbaru JavaScript: Gunakan fitur-fitur terbaru dari ES6 seperti Arrow Function, Spread Operator, dan Destructuring untuk membuat kode lebih bersih dan efisien.
- Menggunakan komponen state dan props dengan benar: Sesuaikan state dan props dengan tepat agar mudah dipahami dan mempermudah dalam pengembangan aplikasi.
- Menggunakan hook dan komponen berbasis function: Komponen berbasis function memiliki performa yang lebih baik dan lebih mudah digunakan daripada class component. Gunakan juga hook untuk mempermudah pengelolaan state dan efek.
- Memperhatikan performa: Gunakan taktik seperti memoization dan lazily-loaded component untuk meningkatkan performa aplikasi.
- Penggunaan modul dan pustaka eksternal dengan bijak: Gunakan modul dan pustaka eksternal yang diterima dan dikenal baik dalam komunitas.
- Menggunakan tools dan teknik pengembangan yang tepat: Gunakan tools seperti Webpack dan Babel untuk membuat pengembangan aplikasi lebih mudah dan efisien.
- Testing dan debugging: Pastikan melakukan tes dan debugging secara teratur selama pengembangan aplikasi.
- Menggunakan best practice dalam penulisan kode: Gunakan best practice seperti penggunaan nama variable yang sesuai dan konsisten, memisahkan logika dan presentasi, dan membuat kode yang terbaca dengan baik.

Prospect ReactJs in The Future



ReactJS merupakan teknologi yang cukup populer dan inovatif untuk pengembangan aplikasi web. ReactJS dikenal dengan mudah dipahami, dapat membantu pengembangan aplikasi web yang cepat, responsif dan mudah di-debug. Ini membuat ReactJS menjadi teknologi yang sangat dicari oleh developer dan perusahaan. Kelebihan lainnya seperti dukungan yang kuat dari komunitas, ketersediaan banyak sumber daya dan bantuan, dan integrasi dengan teknologi lain seperti Node.js dan GraphQL, membuat ReactJS menjadi teknologi yang sangat menarik untuk digunakan dalam proyek-proyek web modern. Oleh karena itu, prospek ReactJS sebagai teknologi front-end sangat baik dan memiliki potensi untuk terus berkembang dan menjadi lebih populer di masa depan.