REACT JS

Pertemuan 5 PWEB –
Robby Nugraha

FRAMEWORK JAVASCRIPT

ReactJS – Pertemuan 5

DEFINISI FRAMEWORK JAVASCRIPT

Framework JavaScript adalah sebuah kerangka untuk mengembangkan website, web app, dan aplikasi dengan bahasa pemrograman JavaScript. Dengan menggunakan framework JavaScript, user tidak perlu menulis kode JavaScript dari nol karena bisa menggunakan kerangka website atau kode dasar yang telah disediakan.

FRAMEWORK JAVASCRIPT

Ada 12 Framework Javascript yaitu:

1. AngularJS
2. ReactJS
3. Vue.js
4. Ember.js
5. Node.js
6. Backbone.js
7. Meteor
8. Mithril
9. Polymer
10. Aurelia
11. Next.js
12. Express S

Selain 12 Framework Javascript, ada framework javascript yang lainnya yaitu:

1. jQuery
2. Midori
3. Ext JS
4. MochiKit
5. Moo Tools
6. Prototype & script.aculo.us
7. Google Web Toolkit
8. Pyjamas
9. SmartClient
10. YUI

FRAMEWORK JAVASCRIPT & JSX

ReactJS – Pertemuan 5

DEFINISI REACTJS

React JS adalah library JavaScript yang biasa digunakan saat membangun UI suatu website atau aplikasi web. Jadi, React JS bisa dianggap seperti perpustakaan yang berisi berbagai kode JavaScript yang sudah tertulis (pre-written). User tinggal mengambil kode yang ingin user gunakan. Sehingga, ini membuat proses coding menjadi lebih efisien dengan framework JavaScript tersebut.

STRUKTUR FOLDER REACTJS

1. node_modules → berisi paket-paket modul Nodejs; semua libaray yang kita install dengan npm akan disimpan di sini.
2. Public → berisi file untuk publik seperti HTML, CSS, icon, dan gambar, dan aset publik lainnya.
    1. Index.html → adalah file HTML yang akan digunakan aplikasi React untuk render komponen.

STRUKTUR FOLDER REACTJS

1. Src → berisi kode dari aplikasi Reactjs, di sinilah kita akan membuat komponen;
    - App.js → berisi kode untuk komponen App komponen utama dari aplikasi.
    - App.test.js → berisi kode untuk testing komponen App
    - Index.js → berisi kode untuk render komponen App ke Real DOM
    - serviceWorker.js → berisi kode untuk service worker, ini yang dibutuhkan nanti saat membuat aplikasi web
    - setTests.js → berisi kode untuk testing aplikasi

STRUKTUR FOLDER REACTJS

1. .gitignore →berisi kode – kode yang akan diabaikan oleh Git
2. Package.json → file JSON yang berisi keterangan proyek dan daftar modul – modul yang dibutukan
3. Yarn.lock → file yang digunakan Yarn untuk mengunci versi – versi modul NodeJS yang digunakan

JSX (JAVASCRIPT XML)

JSX adalah extension syntax JavaScript yang memungkinkan Anda untuk memodifikasi Document Object Model (DOM) dengan kode bergaya HTML.

Sintaks:

```
function Hello() {  return <h1>Hallo Dunia!</h1>}
```

CONTOH JSX

```
const element = <h1>Halo, Dunia!</h1>;
```

```
const name = 'Budi';
const element = <h1>Halo, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

CONTOH JSX

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Budi',
  lastName: 'Mahardika'
};

const element = (
  <h1>
    Halo, {formatName(user)}!
  </h1>
);

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

REACT JS

ReactJS – Pertemuan 5

INSTALL NODE JS & REACTJS

1. Untuk cara install lengkapnya : https://youtu.be/Je0jyEmu5xI
2. Install Node.JS dengan versi LTS
3. Install Git untuk mempermudah install ReactJS
4. Install Text Editor (tidak boleh Notepad++)

KOMPONEN REACTJS

Ada beberapa bagian komponen pada ReactJS yaitu:

1. Class Component
2. Function Component
3. Konstructor
4. Komponen dalam komponen
5. Komponen didalam file

CLASS COMPONENT REACTJS

JavaScript

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  class Car extends React.Component {
5    render() {
6      return <h2>Hi, I am a Car!</h2>;
7    }
8  }
9
10 ReactDOM.render(<Car />,
   document.getElementById('root'));
11
12
```

FUNCTION COMPONENT REACTJS

JavaScript

```
1 function Car() {
2   return <h2>Hi, I am also a Car!</h2>;
3 }
4
```

CONSTRUCTOR COMPONENT REACTJS

JavaScript

```
1 class Car extends React.Component {
2   constructor() {
3     super();
4     this.state = {color: "red"};
5   }
6   render() {
7     return <h2>I am a Car!</h2>;
8   }
9 }
```

COMPONENT DALAM COMPONENT REACTJS

JavaScript

```
import React from 'react';
import ReactDOM from 'react-dom';

class Car extends React.Component {
  render() {
    return <h2>I am a Car!</h2>;
  }
}

class Garage extends React.Component {
  render() {
    return (
      <div>
      <h1>Who lives in my Garage?</h1>
      <Car />
      </div>
    );
  }
}

ReactDOM.render(<Garage />, document.getElementById('root'));
```

COMPONENT FILE REACTJS

App.js

```javascript
import React from 'react';
import ReactDOM from 'react-dom';

class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;


}
}

export default Car;

```

App.js

```javascript
import React from 'react';
import ReactDOM from 'react-dom';

class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}

export default Car;

/*
Notice that you now have three files in your project:
"App.js", "index.js", and "index.html".
*/

```

COMPONENT FILE REACTJS

Index.js

JavaScript

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import Car from './App.js';
4
5 ReactDOM.render(<Car />, document.getElementById('root'));
```

Index.html

HTML

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8" />
5      <meta name="viewport"
6        content="width=device-width, initial-scale=1" />
7      <title>React App</title>
8    </head>
9    <body>
10
11     <div id="root"></div>
12
13   </body>
14 </html>
15
```

STATE & PROPS REACTJS

State dan props adalah objek khusus yang menyimpan data untuk komponen. Kedua objek ini memiliki cara kerja yang berbeda. State adalah objek yang digunakan untuk menyimpan data **di dalam komponen**, sedangkan props adalah obejek yang digunakan untuk menyimpan **data yang diterima dari luar komponen.**

CONTOH STATE REACTJS

```
class Header extends React.Component {
 constructor() {
   super();
   // membuat objek state
   this.state = {
     title: "Belajar Reactjs",
     subTitle: "Panduan step-by-step Reactjs untuk pemula"
   };
 }

 render() {
   return (
     <div>
       <h1>{this.state.title}</h1>
       <h2>{this.state.subTitle}</h2>
     </div>
   );
 }
}

// render komponen ke RealDOM
ReactDOM.render(<Header />, document.getElementById("root"));
```

CONTOH STATE REACTJS

```
class Header extends React.Component {
  constructor() {
    super();
    // membuat objek state
    this.state = {
      title: "Belajar Reactjs",
      subTitle: "Panduan step-by-step Reactjs untuk pemula
    };
  }

  changeTitle = () => {
    this.setState({
      title: "Tutorial Reactjs Petani Kode"
    });
  }

  render() {
    return (
      <div>
        <h1>{this.state.title}</h1>
        <h2>{this.state.subTitle}</h2>
        <button onClick={this.changeTitle}>Ubah Judul</but
      </div>
    );
  }
}

// render komponen ke RealDOM
ReactDOM.render(<Header />, document.getElementById("root"
```

CONTOH STATE REACTJS

```
changeTitle() {
  this.setState({
    title: "Tutorial Reactjs Petani Kode"
  });
}
```

Atau seperti ini:

```
changeTitle = function() {
  this.setState({
    title: "Tutorial Reactjs Petani Kode"
  });
}
```

CONTOH PROPS REACTJS

Props tidak perlu kita buat seperti state, karena ia hanya bertugas untuk menerima data dari luar komponen. Kita tinggal pakai saja.

```
class Message extends React.Component {
  render(){
    return(
      <div>
        <small>{this.props.sender}:</small>
        <p>{this.props.content}</p>
        <hr/>
      </div>
    );
  }
}

// menggunakan komponen
let chat = (
  <div>
  <Message sender="dian" content="Hi, Apa kabar?" />
  <Message sender="petanikode" content="Kabar Baik" />
  </div>
);

// render komponen ke RealDOM
ReactDOM.render(chat, document.getElementById("root"));
```

EVENT REACTJS

Adding Events

React events are written in camelCase syntax:

`onClick` instead of `onclick`.

React event handlers are written inside curly braces:

`onClick={shoot}` instead of `onClick="shoot()"`.

React:

```
<button onClick={shoot}>Take the Shot!</button>
```

HTML:

```
<button onclick="shoot()">Take the Shot!</button>
```

EVENT REACTJS

```
function Football() {
  const shoot = () => {
    alert("Great Shot!");
  }

  return (
    <button onClick={shoot}>Take the shot!</button>
  );
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Football />);
```

STATE HOOK (USESTATE) REACTJS

Import `useState`

To use the `useState` Hook, we first need to `import` it into our component.

Example:

At the top of your component, `import` the `useState` Hook.

```
import { useState } from "react";
```

STATE HOOK (USESTATE) REACTJS

Initialize `useState`

We initialize our state by calling `useState` in our function component.

`useState` accepts an initial state and returns two values:

- The current state.
- A function that updates the state.

Example:

Initialize state at the top of the function component.

```
import { useState } from "react";

function FavoriteColor() {
  const [color, setColor] = useState("");
}
```

STATE HOOK (USESTATE) REACTJS

Read State

We can now include our state anywhere in our component.

Example:

Use the state variable in the rendered component.

```
import { useState } from "react";
import ReactDOM from "react-dom/client";

function FavoriteColor() {
  const [color, setColor] = useState("red");

  return <h1>My favorite color is {color}!</h1>
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<FavoriteColor />);
```

STATE HOOK (USESTATE) REACTJS

Update State

To update our state, we use our state updater function.

We should never directly update state. Ex: `color = "red"` is not allowed.

Example:

Use a button to update the state:

```
import { useState } from "react";
import ReactDOM from "react-dom/client";

function FavoriteColor() {
  const [color, setColor] = useState("red");

  return (
    <>
      <h1>My favorite color is {color}!</h1>
      <button
        type="button"
        onClick={() => setColor("blue")}
      >Blue</button>
    </>
  )
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<FavoriteColor />);
```

STATE HOOK (USESTATE) REACTJS

Example:

Create multiple state Hooks:

```
import { useState } from "react";
import ReactDOM from "react-dom/client";

function Car() {
  const [brand, setBrand] = useState("Ford");
  const [model, setModel] = useState("Mustang");
  const [year, setYear] = useState("1964");
  const [color, setColor] = useState("red");

  return (
    <>
      <h1>My {brand}</h1>
      <p>
        It is a {color} {model} from {year}.
      </p>
    </>
  )
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```

STATE HOOK (USESTATE) REACTJS

Example:

Create a single Hook that holds an object:

```
import { useState } from "react";
import ReactDOM from "react-dom/client";

function Car() {
  const [car, setCar] = useState({
    brand: "Ford",
    model: "Mustang",
    year: "1964",
    color: "red"
  });

  return (
    <>
      <h1>My {car.brand}</h1>
      <p>
        It is a {car.color} {car.model} from {car.year}.
      </p>
    </>
  )
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```

STATE HOOK (USESTATE) REACTJS

Example:

Use the JavaScript spread operator to update only the color of the car:

```
import { useState } from "react";
import ReactDOM from "react-dom/client";

function Car() {
  const [car, setCar] = useState({
    brand: "Ford",
    model: "Mustang",
    year: "1964",
    color: "red"
  });

  const updateColor = () => {
    setCar(previousState => {
      return { ...previousState, color: "blue" }
    });
  }

  return (
    <>
      <h1>My {car.brand}</h1>
      <p>
        It is a {car.color} {car.model} from {car.year}.
      </p>
      <button
        type="button"
        onClick={updateColor}
      >Blue</button>
    </>
  )
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```

STATEFUL DAN STATELESS

Stateful components adalah komponen yang menggunakan state. Sedangkan Stateless adalah komponen yang tidak menggunakan state.

Stateful components juga dikenal dengan sebutan Container dan Smart components.

Stateless juga dikenal dengan sebutan Presentation dan Dumb Components.

METHOD CLASS REACTJS

Example

Create a Class component called `Car`

```
class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}
```

THANK YOU