

BAB I

REACT FRAMEWORK

1. Environment Development

- Toolchain

React toolchain membantu *create*, *build*, *run*, dan *deploy* aplikasi React. Reaksi *toolchain* pada dasarnya menyediakan template proyek awal dengan semua kode yang diperlukan untuk mem-bootstrapkan aplikasi.

Silakan download terlebih dahulu NODE JS versi terbaru melalui link ini <https://nodejs.org/en/>. Setelah berhasil menginstal bukalah terminal untuk mengecek apakah node js dan npm sudah terinstal dengan memasukan syntax dibawah ini:

node -v	npm -v
---------	--------

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Febri>node -v
v18.14.0

C:\Users\Febri>npm -v
8.13.2

C:\Users\Febri>_

```

2. React Project

- Create React App

Create React App adalah alat CLI modern untuk membuat aplikasi React halaman tunggal. Ini adalah alat standar yang didukung oleh komunitas React. Ini menangani kompiler babel juga. Mari kita instal *Create React App* di sistem lokal kita. Untuk membuat proyek, jalankan syntax dibawah ini:

```

npx create-react-app my-app
cd my-app
npm start

```

Ini akan membuat folder baru *my-app* dengan kode template startup.

Selanjutnya masuk kedalam project path *my-app* dan selanjutnya menjalankan project dengan syntax *npm start*.

```

npm start

Compiled successfully!

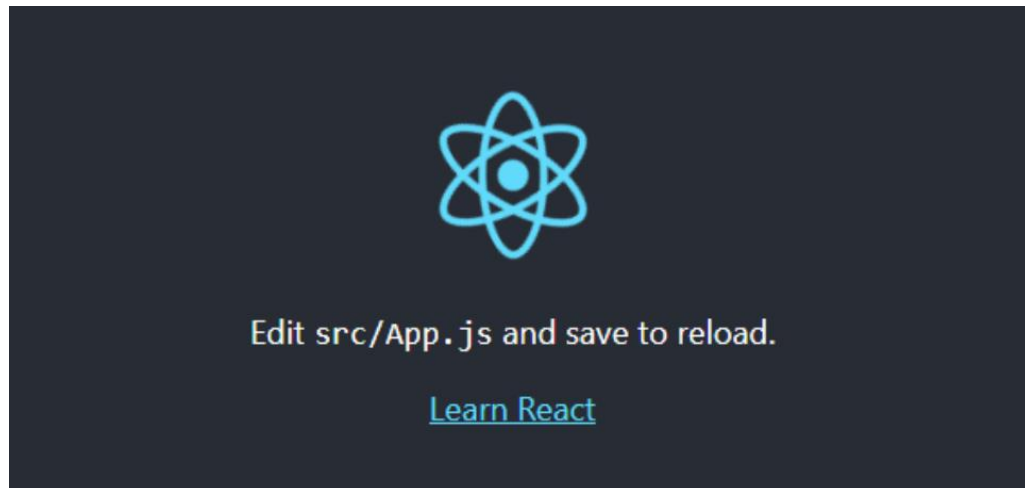
You can now view react-cra-web-app in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

```

Selanjutnya, buka browser dan masukkan <http://localhost:3000> di bilah alamat dan tekan enter. Server web react akan melayani halaman web kami seperti yang ditunjukkan di bawah ini.

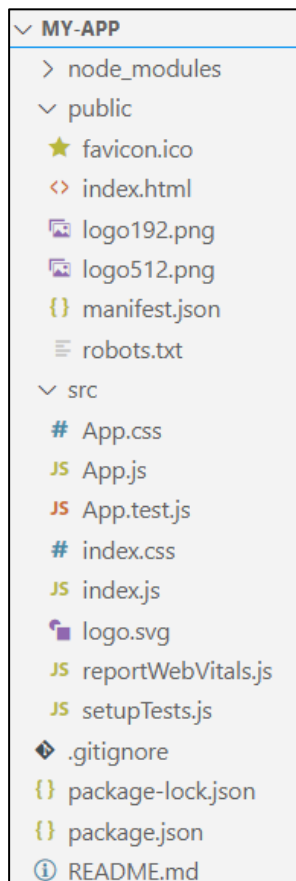


Create React App tidak menangani logika *backend* atau *database*; itu hanya membuat *frontend build pipeline*, sehingga Anda dapat menggunakannya dengan *backend* apa pun yang diinginkan. Di balik layar, ia menggunakan Babel dan webpack, tetapi Anda tidak perlu tahu apa-apa tentangnya.

Babel adalah kompiler JavaScript yang mengkompilasi banyak varian (es2015, es6, dll.) JavaScript ke dalam kode JavaScript standar yang didukung oleh semua browser. React menggunakan JSX, ekstensi JavaScript untuk mendesain kode antarmuka pengguna. Babel digunakan untuk mengkompilasi kode JSX menjadi kode JavaScript. Saat Anda siap menerapkan ke produksi, menjalankan `npm run build` akan membuat build aplikasi Anda yang dioptimalkan di folder build.

- React JS Architecture

Isi dari aplikasi React adalah sebagai berikut:



package.json adalah file inti yang mewakili proyek. Itu mengkonfigurasi seluruh proyek dan terdiri dari nama proyek, dependensi proyek, dan perintah untuk membangun dan menjalankan aplikasi.

```
{
  "name": "my-app",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

Package.json - merujuk pada dependencies React di bawah seperti:

1. *react* dan *react-dom* adalah pustaka reaksi inti yang digunakan untuk mengembangkan aplikasi web.
2. *web-vitals* adalah library umum untuk mendukung aplikasi di browser yang berbeda.
3. *react-scripts* adalah skrip reaksi inti yang digunakan untuk membangun dan menjalankan aplikasi.
4. *@testing-library/jest-dom*, *@testing-library/react* dan *@testing-library/user-event* adalah library pengujian yang digunakan untuk menguji aplikasi setelah pengembangan.

Folder public - Berisi file inti, index.html dan sumber daya web lainnya seperti gambar, logo, robot, dll., index.html akan memuat code aplikasi react kita dan merendernya di browser pengguna.

Folder src - Berisi kode aplikasi yang sebenarnya.

Index.js – Titik awal masuk aplikasi kita. Ini menggunakan metode *ReactDOM.render* untuk memulai dan memulai aplikasi.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

React.StrictMode adalah komponen *build-in* yang digunakan untuk mencegah bug tak terduga dengan menganalisis komponen untuk lifecycle yang tidak aman, penggunaan API yang tidak aman, penggunaan API yang terdepresiasi, dll., dan menampilkan peringatan yang relevan.

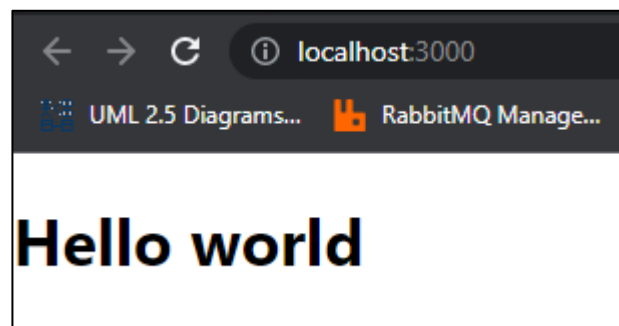
- App adalah komponen kustom dan root, sebagai titik awal masuk ke aplikasi. Semua lainnya komponen akan dirender di dalam App comp

App.js - Komponen root dari aplikasi kita. Mari kita ganti JSX yang ada dan tampilkan pesan hello react sederhana seperti yang ditunjukkan di bawah ini:

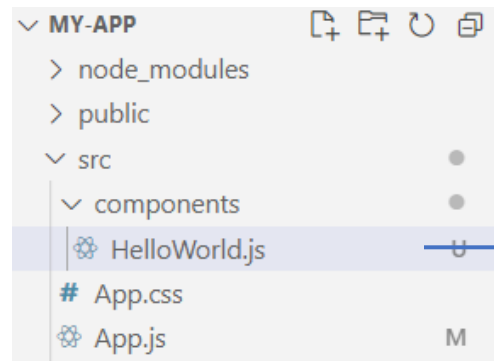
```
import './App.css';

function App() {
  return (
    <h1>Hello world</h1>
  );
}

export default App;
```



Mari kita buat komponen baru, HelloWorld untuk mengonfirmasi bahwa pengaturan kita berfungsi dengan baik. Buat file, *HelloWorld.js* di dalam folder components dan tulislah komponen sederhana untuk menampilkan pesan Hello World.



```
import React, { Component } from 'react'

export default class HelloWorld extends Component {
  render() {
    return (
      <div>
        <h1>HelloWorld</h1>
      </div>
    )
  }
}
```

Selanjutnya, buat komponen HelloWorld menjadi titik awal sebagai root pada file index.js seperti dibawah ini:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import HelloWorld from './components/HelloWorld';
import './index.css';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <HelloWorld />
  </React.StrictMode>
);
reportWebVitals();
```

3. React - JSX

React JSX adalah ekstensi untuk JavaScript. Ini memungkinkan pengembang untuk membuat DOM virtual menggunakan sintaks XML. Itu mengkompilasi ke JavaScript murni (panggilan fungsi *React.createElement*). Karena dikompilasi ke JavaScript, ini dapat digunakan di dalam kode JavaScript yang valid. Misalnya seperti kode di bawah ini benar-benar valid:

1. Assign kedalam variable

```
var greeting = <h1>Hello React!</h1>
```

2. Assign ke dalam variable dengan kondisi

```
var canGreet = true;
if(canGreet) {
  greeting = <h1>Hello React!</h1>
}
```

3. Dapat digunakan sebagai bentuk pengembalian nilai

```
function Greeting() {  
    return <h1>Hello React!</h1>  
}  
greeting = Greeting()
```

4. Digunakan untuk membuat sebuah argumen

```
function Greet(message) {  
    ReactDOM.render(message, document.getElementById('react-app'))  
}  
Greet(<h1>Hello React!</h1>)
```

3.1. Expressions

JSX mendukung ekspresi dalam sintaks JavaScript murni. Ekspresi harus diapit di dalam kurung kurawal, {}. Ekspresi dapat berisi semua variabel yang tersedia dalam konteks, di mana JSX didefinisikan. Mari kita buat JSX sederhana dengan ekspresi.

```
import React from 'react'  
  
export default function Expression() {  
    const today = new Date();  
    return (  
        <div>  
            <h1>Today is {today.toTimeString()}</h1>  
        </div>  
    )  
}
```

Di sini, variable today digunakan di JSX menggunakan ekspresi. Output dari kode di atas adalah sebagai berikut:

Today is 10:42:41 GMT+0700 (Western Indonesia Time)

3.2. Functions

JSX mendukung fungsi JavaScript yang ditentukan pengguna. Penggunaan fungsi mirip dengan ekspresi. Mari kita membuat fungsi sederhana dan menggunakannya di dalam JSX.

```
import React from 'react'  
  
export default function Expression() {  
    const today = new Date();  
    function myName() {  
        return "Febry D F";  
    }  
    const CurrCourse = () =>{  
        return <h3>Pemograman Web Lanjut</h3>;  
    }  
    return (  
        <div>
```

```
        <h1>Today is {today.toString()}</h1>
        <h2>My name is {myName()}</h2>
        {CurrCourse()}
    </div>
  )
}
```

Berikut adalah hasil output dari code diatas untuk implementasi function pada JSX:

Today is 10:47:27 GMT+0700 (Western Indonesia Time)

My name is Febry D F

Pemograman Web Lanjut

3.3. Attributes

JSX mendukung HTML seperti atribut. Semua tag HTML dan atributnya didukung. Atribut harus ditentukan menggunakan konvensi *camelCase* (mengikuti JavaScript DOM API), bukan nama atribut HTML normal. Misalnya, atribut *class* dalam HTML harus didefinisikan sebagai *className*. Berikut ini adalah beberapa contoh lainnya:

```
import React from 'react'

export default function Attributes() {
  const today = new Date;
  function myName() {
    return "Febry D F";
  }
  const CurrCourse = () =>{
    return <h3 style={{fontSize:"40px", color:"green"}}>
      Pemograman Web Lanjut</h3>;
  }
  const bgBlue = {
    backgroundColor:"blue",
    color:"white"
  }
  return (
    <div>
      <h1 style={bgBlue}>Today is {today().toString()}</h1>
      <h2>My name is <span className="red">{myName()}</span></h2>
      {CurrCourse()}
    </div>
  )
}
```

Berikut adalah hasil outputnya:

Today is 10:56:21 GMT+0700 (Western Indonesia Time)

My name is **Febry D F**

Pemograman Web Lanjut

4. React – Component

React library memiliki dua tipe komponen. Jenis-jenis tersebut dikategorikan berdasarkan cara pembuatannya.

- Komponen fungsi - Menggunakan fungsi JavaScript biasa.
- Komponen class ES6 - Menggunakan kelas ES6.

Perbedaan inti antara fungsi dan komponen class adalah:

- Komponen kelas mendukung manajemen *state* di luar area sedangkan komponen fungsi tidak mendukung manajemen *state*. Tapi, React menyediakan sebuah *hook*, *useState()* untuk komponen fungsi untuk mempertahankan statusnya.
- Komponen kelas memiliki *life cycle* dan akses ke setiap peristiwa *life cycle* melalui api callback khusus. Komponen fungsi tidak memiliki *life cycle*. Sekali lagi, React menyediakan sebuah *hook*, *useEffect()* untuk komponen fungsi untuk mengakses tahapan komponen yang berbeda.

5. State dan Life cycle

5.1.State pada React Function Component

```
import React from 'react'

export default function Expression() {
  const today = new Date;
  return (
    <div>
      <h1>Today is {today().toString()}</h1>
    </div>
  )
}
```

5.2.State pada React Class Component

Tambahkan *constructor class* untuk menginisialisasi *this.state*:

```
import React, { Component } from 'react'

export default class HelloWorld extends Component {
  constructor(props) {
    super(props);
    this.state = {
      today: new Date(),
      myname: "Febry D F"
    };
  }
  render() {
```



```

    return (
      <div>
        <h1>Today is {this.state.today.toTimeString()}</h1>
        <h2>My Name is {this.state.myname}</h2>
      </div>
    )
  }
}

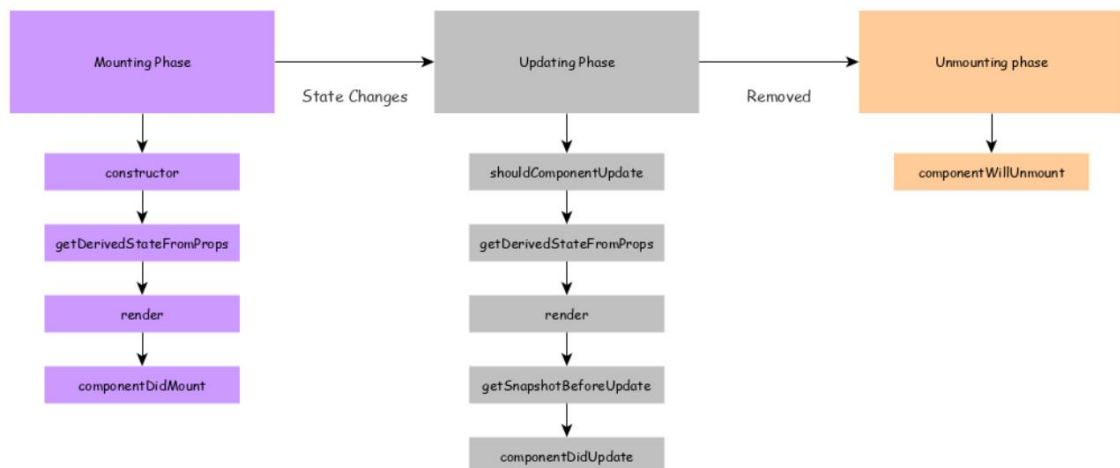
```

5.3. Life cycle pada React Function Component (Component Life Cycle API)

Setiap komponen React memiliki tiga tahapan berbeda.

- **Mounting** - Mounting merepresentasikan rendering komponen React di node DOM yang diberikan.
- **Updating** - Memperbarui mewakili rendering ulang komponen React di node DOM yang diberikan selama perubahan / pembaruan status.
- **Unmounting** - Unmounting menunjukkan penghapusan komponen React.

React menyediakan kumpulan peristiwa siklus hidup (atau callback API) untuk melampirkan fungsionalitas, yang akan dieksekusi selama berbagai tahapan komponen. Visualisasi siklus hidup dan urutan di mana peristiwa siklus hidup (API) dipanggil seperti yang ditunjukkan di bawah:



constructor() - Dipanggil selama fase konstruksi awal komponen React. Digunakan untuk mengatur status awal dan properti komponen.

render() - Dipanggil setelah pembangunan komponen selesai. Itu merender komponen dalam instance DOM virtual. Ini ditentukan sebagai pemasangan komponen di pohon DOM.

componentDidMount() - Dipanggil setelah pemasangan awal komponen di pohon DOM. Ini adalah tempat yang baik untuk memanggil titik akhir API dan melakukan permintaan jaringan. Dalam komponen jam kami, fungsi setInterval dapat diatur di sini untuk memperbarui status (tanggal saat ini dan waktu) untuk setiap detik.

componentWillUnmount() - Dipanggil setelah komponen dilepas dari DOM. Ini adalah tempat yang bagus untuk membersihkan objek. Dalam contoh jam kami, kami dapat berhenti memperbarui tanggal dan waktu dalam fase ini.

componentDidUpdate() - Mirip dengan **ComponentDidMount()** tetapi dipanggil selama fase pembaruan. Permintaan jaringan dapat dilakukan selama fase ini tetapi hanya jika ada perbedaan dalam properti komponen saat ini dan sebelumnya.

Berikut ini adalah contoh penggunaan React Life Cycle API :

```
import React, { Component } from 'react'

export default class HelloWorld extends Component {
  constructor(props) {
    super(props);
    this.CountTimer = this.CountTimer.bind(this);
    this.state = {
      today: new Date()
    };
  }
  CountTimer() {
    this.setState({
      today: new Date()
    });
  }

  componentDidMount() {
    this.timerID = setInterval(
      () => this.CountTimer(),
      1000
    );
  }
  componentWillUnmount() {
    clearInterval(this.timerID);
  }
  render() {
    return (
      <div>
        <h1>Today is {this.state.today.toTimeString()}</h1>
      </div>
    )
  }
}
```

5.4. Life cycle pada React Class Component (Hooks)

React Hooks adalah fungsi khusus yang disediakan oleh *React* untuk menangani fungsionalitas tertentu di dalam komponen fungsional *React*. *React* menyediakan fungsi *Hook* untuk setiap fitur yang didukung. Misalnya, *React* menyediakan fungsi *useState()* untuk mengelola status dalam suatu fungsi komponen. Ketika komponen fungsional *React* menggunakan *React Hooks*, *React Hooks* melampirkan dirinya sendiri ke dalam komponen dan menyediakan fungsionalitas tambahan.

Code umum dari penggunaan *useState()* *Hook* adalah sebagai berikut:

```
const [<state variable>, <state update function>] = useState(<initial value>);
```

React Hooks menyediakan Hook khusus, *useEffect()* untuk mengeksekusi fungsionalitas tertentu pada life cycle komponen. *useEffect()* menggabungkan siklus hidup *componentDidMount*, *componentDidUpdate*, dan *componentWillUnmount* ke dalam satu api. Berikut adalah cara penggunaan *useEffect()* secara umum:

```
useEffect(()=>{  
  <executeFn>  
})
```

Berikut adalah contoh penggunaan React Hooks:

```
import React, { useEffect, useState } from 'react'  
  
export default function Expression() {  
  
  const today = new Date();  
  const [currDate, setCurrDate] = useState(today);  
  
  const countTimer = () =>{  
    setCurrDate(new Date());  
  }  
  
  useEffect(()=>{  
    setInterval(  
      () => countTimer(),  
      1000  
    );  
  }, [])  
  
  return (  
    <div>  
      <h1>Today is {currDate.toTimeString()}</h1>  
    </div>  
  )  
}
```

6. List and Keys

Kode di bawah ini, kami menggunakan fungsi *map()* untuk mengambil array angka dan menggandakan nilainya. Kami menetapkan array baru yang dikembalikan oleh *map()* ke variabel dua kali lipat dan mencatatnya:

```
const numbers = [1, 2, 3, 4, 5];  
const listItems = numbers.map((number) =>  
  <li>{number}</li>  
);
```

Kemudian, kita bisa menyertakan seluruh array *listItems* di dalam elemen ** pada JSX:

```
<ul>{listItems}</ul>
```

- 1
- 2
- 3
- 4
- 5

7. Latihan Praktikum

1. Buatlah project react bernama Project-PWL-NPM.
2. Tambahkan framework CSS yaitu Bootstrap kedalam project anda.
3. Buatlah sebuah form data diri yaitu:
 - NPM
 - First Name
 - Middle Name
 - Last Name
 - Birthdate

Buatlah sebuah form submit sesuai dengan isian field diatas dengan kondisi:

- Isian NPM hanya untuk NUMERIC dan jumlah maksimum sebanyak 10
- Isian Birthdate boleh menggunakan library datepicker atau manual dengan inputan format YYYY-MM-DD
- Seluruh form isian wajib diisi kecuali Middle Name
- Jika sudah terisi dengan benar, ketika mensubmit maka akan menampilkan informasi dalam bentuk Pop UP (Modal) seperti dibawah ini:

NPM : 3320190306
Fullname : Febry Damatraseta Fairuz
Age : 70th

Nilai isian Age diambil dari kalkulasi isian field Birthdate.

Pengumpulan tugas Latihan praktikum dikumpulkan kedalam GITHUB masing-masing mahasiswa berdasarkan repository yang telah dibuat PWL-TI-20-PA-NPM. File source code disimpan sesuai nama project-praktikum dan masukan kedalam repositori tersebut. Buatlah file dokumen dalam bentuk file pdf yang berisi Screen Capture dari hasil program yang telah dikerjakan. Simpan dalam file PDF tersebut kedalam project tersebut.

Tambahkan Collaborator management access pada repository anda ke *@FebryFairuz*