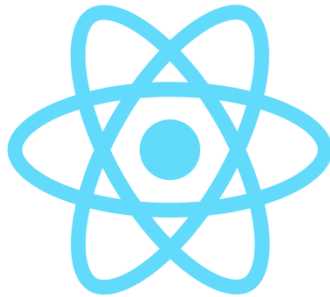


“React JS from Zero”



In collaboration with



Created By:

Onesinus Saut Parulian

Reference:

Eko Andri Subarnanto @ React JS Training Batch #1

2019

Daftar Isi

1.	Intalasi React, Menjalankan Aplikasi	4
1.1	Instalasi React	4
1.2	Menjalankan React	6
1.3	Membuat code pertama React	8
2.	React Class Component, Functional Component, Props	11
2.1	Membuat Class Component & Functional Component	11
2.2	Membuat dan menggunakan Props pada Component	14
2.3	Membuat component pada file terpisah (Export, Import Component)	19
2.4	Membuat component dengan ekstensi .jsx.....	24
2.5	Membuat component List.....	25
2.6	Memindahkan component ke sebuah folder.....	27
2.7	Cara mengubah functional component menjadi class component	28
2.8	Mengembangkan component List	30
2.9	Memindahkan tombol di List.jsx ke App.js (Memindahkan logic dari class component ke functional component)	32
2.10	Membuat Component Input.....	36
2.11	Membuat event handler ketika menekan key enter di inputan	38
2.12	Membuat component Form	41
2.13	Membuat component dengan property object	43
2.14	Membuat component button fixed style dengan sebuah property pembeda (Using external Css).....	46
2.15	Mengubah component List menjadi lebih dinamis.....	50
3.	Persiapan Project (Setup Database, dll).....	58
3.1	Membuat Database dan Table	59
3.2	Membuat web service (Welcome & get list data).....	61
3.3	Membuat api untuk memasukan data ke mysql	69
3.4	Uji api post ke mysql.....	70
3.5	Membuat form ketika disubmit simpan data ke table	73
4.	Mini Project dengan React	78

About The Author	78
About The React JS Training Instructor	80

1. Instalasi React, Menjalankan Aplikasi

Dibagian pertama ini kita akan menginstall aplikasi pertama kita dengan react, dan kita akan coba menjalankan aplikasi react, serta mencoba melakukan perubahan pada halaman awal

Sebelum melanjutkan bagian ini pastikan sudah menginstall node js dikomputer anda, dan sudah bisa menjalankan perintah npm

1.1 Instalasi React

Buka command promp / terminal sesuai sistem operasi anda dan jalankan perintah

`npm install -g create-react-app`

```
PS E:\buku\Pemula> npm install -g create-react-app
[.....] \ extract:create-react-app: sill extract create-react-app@latest extracted to C:\Users\onesi\AppData
```

Jika sudah selesai maka akan tampil output seperti ini

```
PS E:\buku\Pemula> npm install -g create-react-app
C:\Users\onesi\AppData\Roaming\npm\create-react-app -> C:\Users\onesi\AppData\Roaming\npm\create-react-app@3.2.0
+ create-react-app@3.2.0
updated 4 packages in 107.382s
```

Selanjutnya buat aplikasi pertama kita dengan perintah

`create-react-app my-first-react`

```
PS E:\buku\Pemula> create-react-app my-first-react

Creating a new React app in E:\buku\Pemula\my-first-react.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...

[.....] / fetchMetadata: sill resolveWithNewModule js-tokens@4.0.0 chec
```

Jika sudah berhasil membuat aplikasi pertama, maka akan tampil output seperti ini

*Dalam proses instalasi mungkin akan memerlukan waktu, tergantung dari koneksi internet yang kita miliki, maka dari itu pastikan terhubung ke internet dan memiliki koneksi yang baik

*Dalam proses instalasi ini akan mendownload struktur folder yang dibutuhkan untuk aplikasi react kita seperti folder node_module

```
PS E:\buku\Pemula> create-react-app my-first-react

Creating a new React app in E:\buku\Pemula\my-first-react.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...

> core-js@2.6.10 postinstall E:\buku\Pemula\my-first-react\node_modules\babel-runtime\node_modules\core-js
> node postinstall || echo "ignore"

> core-js@3.2.1 postinstall E:\buku\Pemula\my-first-react\node_modules\core-js
> node scripts/postinstall || echo "ignore"

+ react-scripts@3.2.0
+ react@16.11.0
We suggest that you begin by typing:

  cd my-first-react
  npm start
```

Setelah proses instalasi berhasil maka akan terbentuk project folder seperti ini

```
▼ my-first-react ●
  > node_modules ●
  > public ●
  > src ●
  ◆ .gitignore A
  {} package-lock.json A
  {} package.json A
  ⓘ README.md A
```

Juga kita bisa menjalankan perintah npm start diterminal console kita

1.2 Menjalankan React

Sekarang mari kita coba menjalankan aplikasi default react yang baru saja kita install

Buka folder project kita

Cd my-first-react

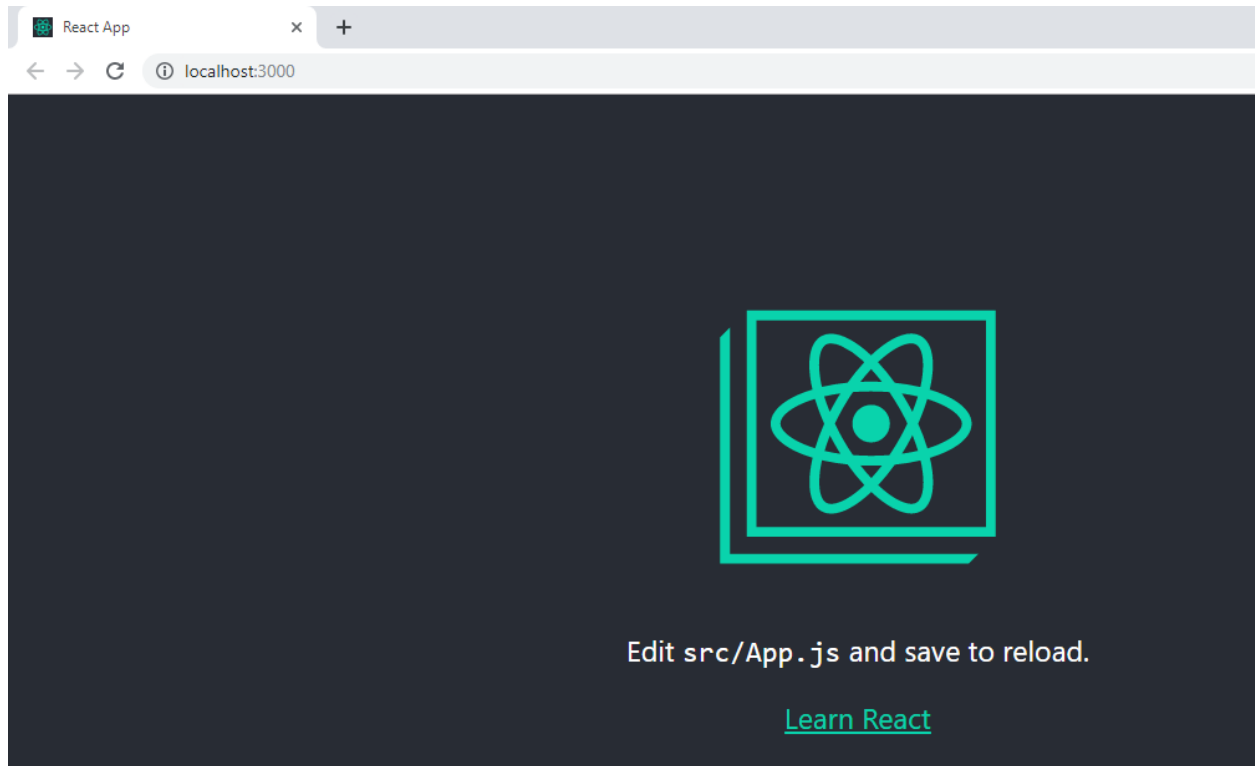
```
PS E:\buku\Pemula> cd .\my-first-react\  
PS E:\buku\Pemula\my-first-react> |
```

Jika sudah masuk ke folder my-first-react maka sekarang tinggal jalankan npm start

```
E:\buku\Pemula\my-first-react>npm start
```

```
Compiled successfully!  
  
You can now view my-first-react in the browser.  
  
http://localhost:3000/  
  
Note that the development build is not optimized.  
To create a production build, use npm run build.  
  
|
```

Maka akan terbuka dibrowser aplikasi pertama react



*Jika tidak terbuka otomatis maka akses dari browser dengan url localhost:3000 sebagai url default dari react js yang diinstal dengan create-react-app

1.3 Membuat code pertama React

Sekarang coba buka App.js

```
JS App.js x
my-first-react > src > JS App.js > ...
 1  import React from 'react';
 2  import logo from './logo.svg';
 3  import './App.css';
 4
 5  function App() {
 6    return (
 7      <div className="App">
 8        <header className="App-header">
 9          <img src={logo} className="App-logo" alt="logo" />
10          <p>
11            Edit <code>src/App.js</code> and save to reload.
12          </p>
13          <a
14            className="App-link"
15            href="https://reactjs.org"
16            target="_blank"
17            rel="noopener noreferrer"
18          >
19            Learn React
20          </a>
21        </header>
22      </div>
23    );
24  }
25
26  export default App;
27
```

Sekarang mari kita coba mengubah code di App.js menjadi seperti ini


```
JS App.js x
my-first-react > src > JS App.js > App
1 import React from 'react';
2 import logo from './logo.svg';
3 import './App.css';
4
5 function App() {
6   return (
7     <div className="App">
8       <header className="App-header">
9         <img src={logo} className="App-logo" alt="logo" />
10        <p>
11          This is my first app
12        </p>
13        <a
14          className="App-link"
15          href="https://reactjs.org"
16          target="_blank"
17          rel="noopener noreferrer"
18        >
19          Learn React
20        </a>
21      </header>
22    </div>
23  );
24 }
25
26 export default App;
```

Kemudian simpan App.js. maka browser otomatis refresh dan tampilan akan berubah menjadi



This is my first app

[Learn React](#)

2. React Class Component, Functional Component, Props

Pada bagian ini kita akan mencoba membuat Class Component pada React, juga membuat Functional Component, dan bagaimana cara membuat dan menggunakan Props pada React

2.1 Membuat Class Component & Functional Component

Ada 2 cara membuat component direct

1. Class Component
2. Functional Component

Ubah App.js dan buat menjadi seperti ini

```
import React, { useState, Component } from "react";
import './App.css';

export default function App() {
  return (
    <div className="App">
      <header className="App-header">
        <Hello />
        <HelloClass />
      </header>
    </div>
  )
}

function Hello() {
  const [nama, setNama] = useState("Lesiong");
  const [jabatan, setJabatna] = useState("Santuy Senior");
  const [usia, setUsia] = useState(41);
  return (
    <div>
      <h1>Hello {nama}</h1>
      <p>jabatan anda: {jabatan}</p>
      <p>usia anda: {usia}</p>
    </div>
  )
}
```

```

)
}

class HelloClass extends Component {
  constructor(props) {
    super(props);
    this.state = {
      nama: "Santuy",
      jabatan: "Melendoy developer",
      usia: 14
    }
  }
}

render() {
  const { nama, jabatan, usia } = this.state;
  return (
    <div>
      <h1>Hello {nama}</h1>
      <p>jabatan anda: {jabatan}</p>
      <p>usia anda: {usia}</p>
    </div>
  )
}
}

```

Dari code diatas kita sudah membuat 2 component yang pertama component Hello dan yang kedua component HelloClass

Di fungsi App kita menampilkan 2 component yang berasal dari Class maupun dari Functional component

Jika kita lihat dibrowser

Hello Lesiong
jabatan anda: Santuy Senior
usia anda: 41

Component `<Hello />`
yang kita buat
menggunakan
functional component

Hello Santuy
jabatan anda: Melendoy developer
usia anda: 14

Component `<HelloClass />`
yang kita buat
menggunakan
Class Component

Sesuai dengan code di App.js yang sudah menampilkan 2 component tersebut

```
export default function App() {  
  return (  
    <div className="App">  
      <header className="App-header">  
        <Hello />  
        <HelloClass />  
      </header>  
    </div>  
  )  
}
```

Yang mana 2 component tersebut dibuat di code ini

```

function Hello ) {
  const [nama, setNama] = useState("Lesiong");
  const [jabatan, setJabatna] = useState("Santuy Senior");
  const [usia, setUsia] = useState(41);
  return (
    <div>
      <h1>Hello {nama}</h1>
      <p>jabatan anda: {jabatan}</p>
      <p>usia anda: {usia}</p>
    </div>
  )
}

```

```

class HelloClass extends Component {
  constructor(props) {
    super(props);
    this.state = {
      nama: "Santuy",
      jabatan: "Melendoy developer",
      usia: 14
    }
  }

  render() {
    const { nama, jabatan, usia } = this.state;
    return (
      <div>
        <h1>Hello {nama}</h1>

```

2.2 Membuat dan menggunakan Props pada Component

Sekarang mari kita coba buat component lagi, agar kita semakin terbiasa dengan pembuatan dan penggunaan componen react dan juga dapat mempelajari tentang Props

```

function Button(){
  const style = {
    padding: "5px 10px",
    color: "red",
    border: "solid 2px red"
  }
  return <button style={style}>Test Button</button>
}

function ButtonProps(props){
  const style = {
    padding: "5px 10px",

```

```

    color: props.color,
    border: 'solid 2px ${props.color}'
  }
  return <button style={style}>{props.nama}</button>
}

function ButtonDenganChildrenProps(props){
  const style = {
    padding: "5px 10px",
    color: props.color,
    border: "solid 2px ${props.color}"
  }
  return <button style={style}>{props.children}</button>
}

```

Kemudian kita coba panggil 1 component dari 3 component yang sudah kita buat

```

export default function App() {
  return (
    <div className="App">
      <header className="App-header">
        <Hello />
        <HelloClass />
        <Button />
      </header>
    </div>
  )
}

```

Maka tampilan aplikasi kita akan menjadi

Hello Lesiong

jabatan anda: Santuy Senior

usia anda: 41

Hello Santuy

jabatan anda: Melendoy developer

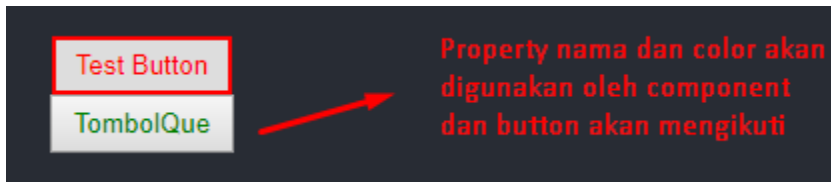
usia anda: 14

Test Button

Sekarang untuk memahami konsep property (props) di react js, mari kita coba panggil component ButtonProps

```
export default function App() {  
  return (  
    <div className="App">  
      <header className="App-header">  
        <Hello />  
        <HelloClass />  
        <Button />  
        <ButtonProps nama="TomboIQue" color="green" />  
      </header>  
    </div>  
  )  
}
```


Karena kita memanggil component ButtonProps dengan 2 property yaitu nama dan color maka component tampilan aplikasi kita akan menjadi



Mari kita coba tambahkan sebuah property lagi untuk component ButtonProps kita

Ubah fungsi App kita menjadi

```
export default function App() {
  function onClickTombolQue(){
    console.log("tombol que diklik");
  }
  return (
    <div className="App">
      <header className="App-header">
        <Hello />
        <HelloClass />
        <Button />
        <ButtonProps nama="TombolQue" color="green" onClick={onClickTombolQue} />
      </header>
    </div>
  )
}
```

Perubahan pada fungsi App

The image shows a code snippet with two red boxes. The first box highlights the function definition: `function onClickTombolQue(){ console.log("tombol que diklik"); }`. The second box highlights the prop value in the component: `onClick={onClickTombolQue}`. A red arrow points from the function definition to the prop value.

```
export default function App() {
  function onClickTombolQue(){
    console.log("tombol que diklik");
  }
  return (
    <div className="App">
      <header className="App-header">
        <Hello />
        <HelloClass />
        <Button />
        <ButtonProps nama="TombolQue" color="green" onClick={onClickTombolQue} />
      </header>
    </div>
  )
}
```

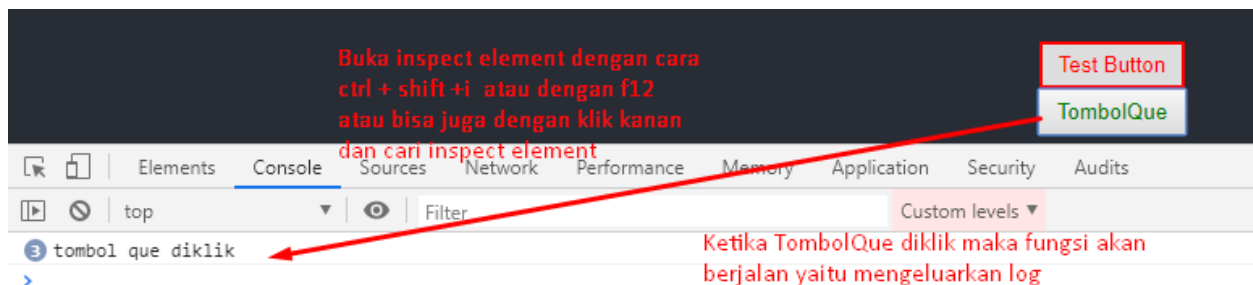
Kemudian ubah fungsi ButtonProps menjadi seperti ini

```
function ButtonProps(props){
  const style = {
    padding: "5px 10px",
    color: props.color,
    border: 'solid 2px ${props.color}'
  }
  return <button style={style} onClick={props.onClick}>{props.nama}</button>
}
```

Perubahan yang terjadi pada fungsi Button Props

```
function ButtonProps(props){
  const style = {
    padding: "5px 10px",
    color: props.color,
    border: 'solid 2px ${props.color}'
  }
  return <button style={style} onClick={props.onClick}>{props.nama}</button>
}
```

Kemudian jika kita buka aplikasi react kita dan klik TombolQue maka akan terjadi seperti ini



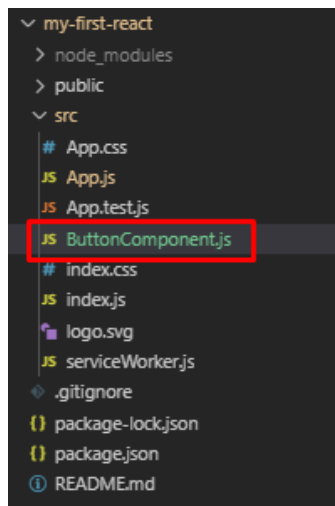
Nah sekarang kita sudah cukup familiar dengan Class Component, Functional Component, cara memanggil serta menggunakan property dari component-component tersebut

2.3 Membuat component pada file terpisah (Export, Import Component)

Sekarang tugas kita adalah memastikan semua component tidak menumpuk hanya disatu file, tentu saja kita bisa memisahkan component menjadi sebuah file terpisah agar code kita tidak berantakan dan numpuk

Mari kita coba untuk mulai memindahkan 3 component button ke sebuah file yaitu ButtonComponent.js

Pertama buat file ButtonComponent.js di folder src



Kemudian pindahkan 3 fungsi button yang kita buat defile App.js

```
JS App.js X JS ButtonComponent.js
my-first-react > src > JS App.js > ButtonDenganChildrenProps
53 }
54 }
55 }
56 function Button(){
57   const style = {
58     padding: "5px 10px",
59     color: "red",
60     border: "solid 2px red"
61   }
62   return <button style={style}>Test Button</button>
63 }
64
65 function ButtonProps(props){
66   const style = {
67     padding: "5px 10px",
68     color: props.color,
69     border: 'solid 2px ${props.color}'
70   }
71   return <button style={style} onClick={props.onClick}>{props.nama}</button>
72 }
73
74 function ButtonDenganChildrenProps(props){
75   const style = {
76     padding: "5px 10px",
77     color: props.color,
78     border: "solid 2px ${props.color}"
79   }
80   return <button style={style}>{props.children}</button>
81 }
```

Sehingga file ButtonComponent.js menjadi

```
JS App.js JS ButtonComponent.js X
my-first-react > src > JS ButtonComponent.js > ...
1 import React from "react";
2
3 function Button() {
4   const style = {
5     padding: "5px 10px",
6     color: "red",
7     border: "solid 2px red"
8   }
9   return <button style={style}>Test Button</button>
10 }
11
12 function ButtonProps(props) {
13   const style = {
14     padding: "5px 10px",
15     color: props.color,
16     border: 'solid 2px ${props.color}'
17   }
18   return <button style={style} onClick={props.onClick}>{props.nama}</button>
19 }
20
21 function ButtonDenganChildrenProps(props) {
22   const style = {
23     padding: "5px 10px",
24     color: props.color,
25     border: "solid 2px ${props.color}"
26   }
27   return <button style={style}>{props.children}</button>
28 }
29
30 export default Button;
31 export [ButtonProps, ButtonDenganChildrenProps]
```

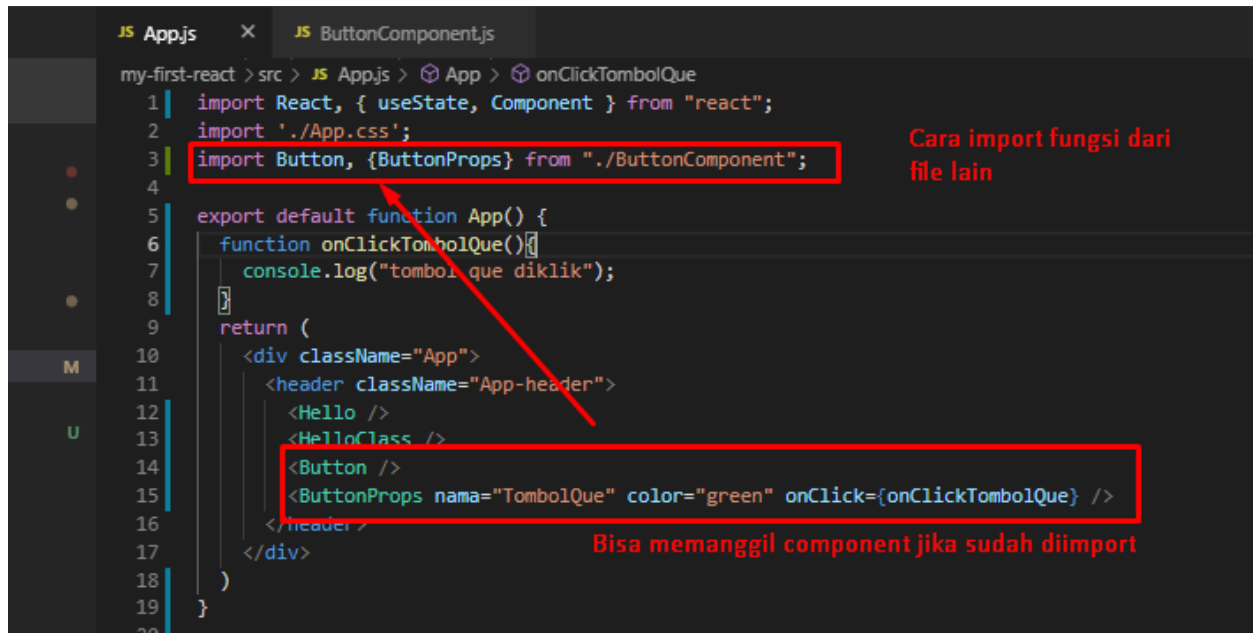
Wajib Import react karena didalamnya kita akan membuat component direact

Fungsi-fungsi yang kita pindahkan dari App.js

Untuk dapat digunakan dari file lain (App.js) Perlu kita export

Kemudian di file App.js

Import fungsi-fungsi tersebut



```
1 import React, { useState, Component } from "react";
2 import './App.css';
3 import Button, { ButtonProps } from './ButtonComponent';
4
5 export default function App() {
6   function onClickTombolQue() {
7     console.log("tombol que diklik");
8   }
9   return (
10    <div className="App">
11      <header className="App-header">
12        <Hello />
13        <HelloClass />
14        <Button />
15        <ButtonProps nama="TombolQue" color="green" onClick={onClickTombolQue} />
16      </header>
17    </div>
18  )
19 }
```

Cara import fungsi dari file lain

Bisa memanggil component jika sudah diimport

Begitu cara memindahkan component-component ke file lain sesuai kegunaannya

Jika kita perhatikan ada perbedaan cara memanggil component

Yaitu component **Button** dengan component **ButtonProps**

Perbedaan ini terjadi karena perbedaan pada saat export di file ButtonComponent.js

```
export default Button;
export {ButtonProps, ButtonDenganChildrenProps}
```

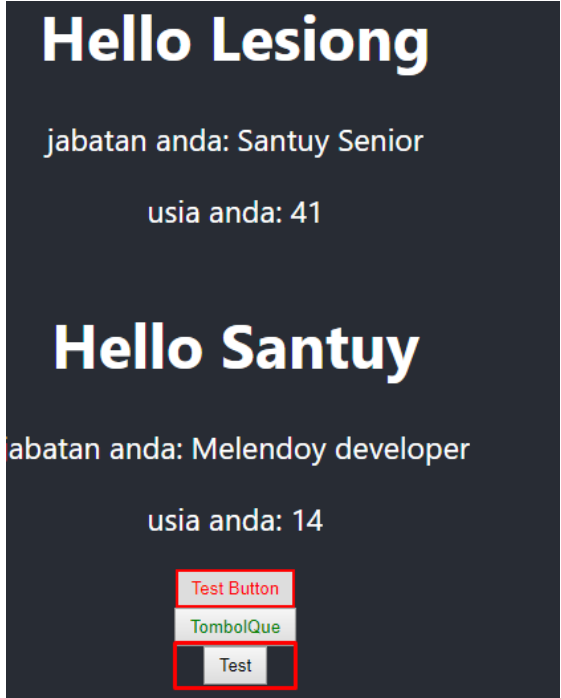
Jika export default panggil diluar kurung kurawal, tetapi jika bukan export default panggil didalam kurung kurawal.

Sekarang mari kita coba fungsi yang kita buat terakhir yaitu yang menggunakan special property children

```
JS App.js x JS ButtonComponent.js
my-first-react > src > JS App.js > App > onClickTombolQue
1 import React, { useState, Component } from "react";
2 import './App.css';
3 import Button, { ButtonProps, ButtonDenganChildrenProps } from './ButtonComponent';
4
5 export default function App() {
6   function onClickTombolQue() {
7     console.log("tombol que diklik");
8   }
9   return (
10    <div className="App">
11      <header className="App-header">
12        <Hello />
13        <HelloClass />
14        <Button />
15        <ButtonProps nama="TombolQue" color="green" onClick={onClickTombolQue} />
16        <ButtonDenganChildrenProps>
17          <div>Test</div>
18        </ButtonDenganChildrenProps>
19      </header>
20    </div>
21  )
22 }
```

Props.children adalah property dimana apapun element yang ada didalam component ButtonDenganChildrenProps akan dianggap sebagai children

Tampilan aplikasi kita akan menjadi seperti ini



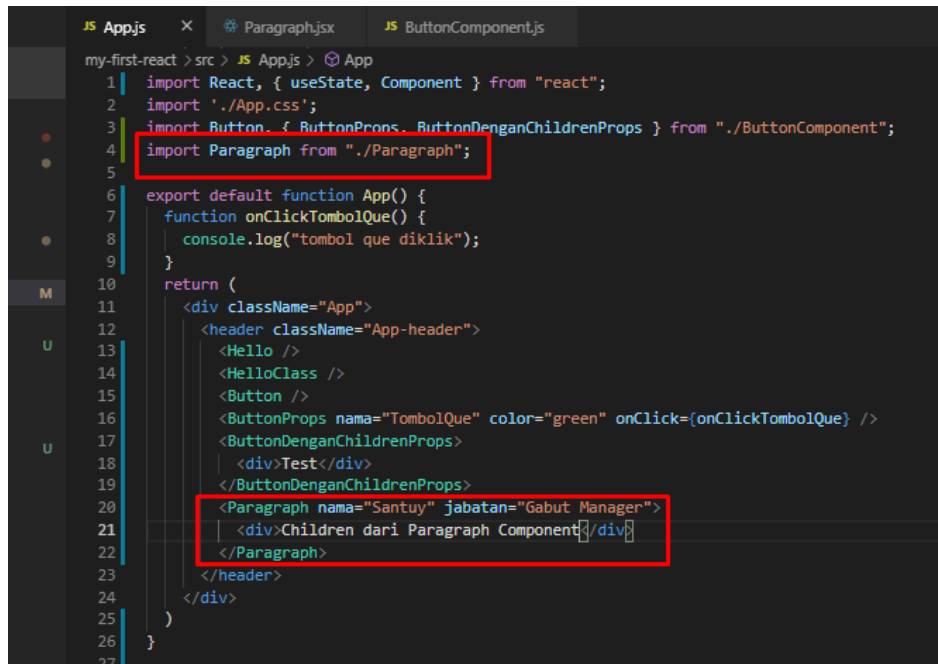
2.4 Membuat component dengan ekstensi .jsx

Buat sebuah file bernama Paragraph.jsx Dan isi code menjadi seperti ini

```
import React from "react";

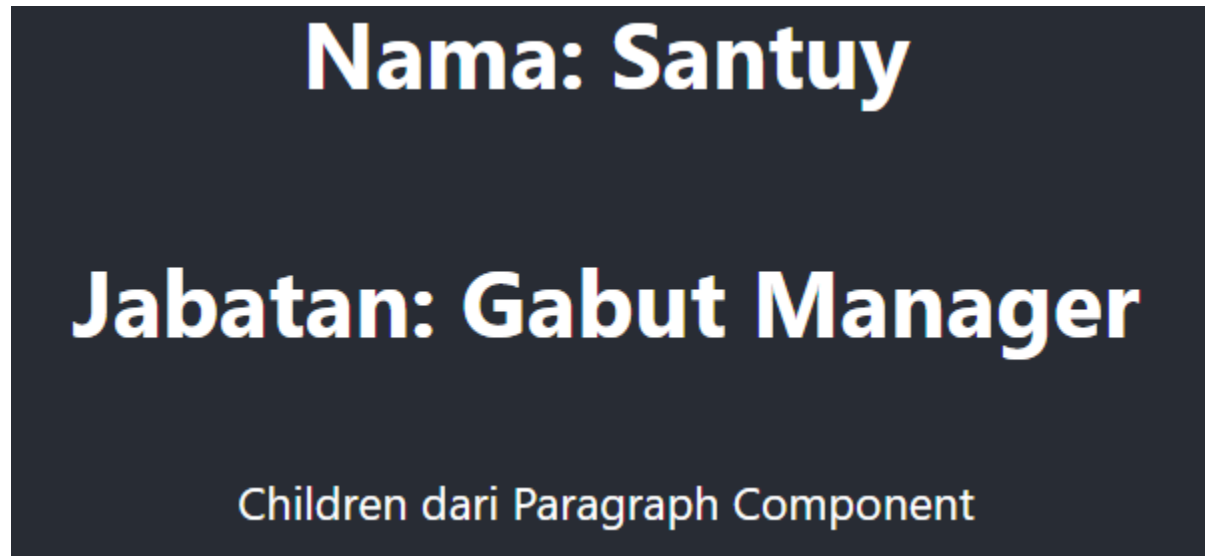
export default function Paragraph(props){
  return(
    <React.Fragment>
      <h1>Nama: {props.nama}</h1>
      <h1>Jabatan: {props.jabatan}</h1>
      <p>{props.children}</p>
    </React.Fragment>
  )
}
```

Kemudian di file App.js import dan gunakan component Paragraph dengan memberikan sebuah children dengan code seperti ini



```
JS Appjs x Paragraph.jsx JS ButtonComponent.js
my-first-react > src > JS Appjs > App
1 import React, { useState, Component } from "react";
2 import './App.css';
3 import Button, { ButtonProps, ButtonDenganChildrenProps } from './ButtonComponent';
4 import Paragraph from './Paragraph';
5
6 export default function App() {
7   function onClickTombolQue() {
8     console.log("tombol que diklik");
9   }
10  return (
11    <div className="App">
12      <header className="App-header">
13        <Hello />
14        <HelloClass />
15        <Button />
16        <ButtonProps nama="TombolQue" color="green" onClick={onClickTombolQue} />
17        <ButtonDenganChildrenProps>
18          <div>Test</div>
19        </ButtonDenganChildrenProps>
20        <Paragraph nama="Santuy" jabatan="Gabut Manager">
21          <div>Children dari Paragraph Component</div>
22        </Paragraph>
23      </header>
24    </div>
25  )
26 }
27
```


Maka tampilan aplikasi kita akan menjadi



2.5 Membuat component List

Buat sebuah file bernama List.jsx dan buat code seperti ini

```
import React from "react";

export default function List(props){
  return (
    <ul>
      {
        props.data.map(item => (
          <React.Fragment>
            <li>
              Nama: {item.name}, Usia: {item.age}{" "}
            </li>
          </React.Fragment>
        ))
      }
    </ul>
  )
}
```

Kemudian panggil component List di App.js

```
JS App.js x List.jsx
my-first-react > src > JS App.js > App
1 import React, { useState, Component } from "react";
2 import './App.css';
3 import Button, { ButtonProps, ButtonDenganChildrenProps } from './ButtonComponent';
4 import Paragraph from './Paragraph';
5 import List from './List';
6
7 export default function App() {
8   const data = [{name: "Onesinus SPT", age: 22}, {name: "Melendoy", age: 23}];
9
10  function onClickTombolQue() {
11    console.log("tombol que diklik");
12  }
13  return (
14    <div className="App">
15      <header className="App-header">
16        <Hello />
17        <HelloClass />
18        <Button />
19        <ButtonProps nama="TombolQue" color="green" onClick={onClickTombolQue} />
20        <ButtonDenganChildrenProps>
21          <div>Test</div>
22        </ButtonDenganChildrenProps>
23        <Paragraph nama="Santuy" jabatan="Gabut Manager">
24          <div>Children dari Paragraph Component</div>
25        </Paragraph>
26
27        <List data={data} />
28      </header>
29    </div>
30  )
31
32
```

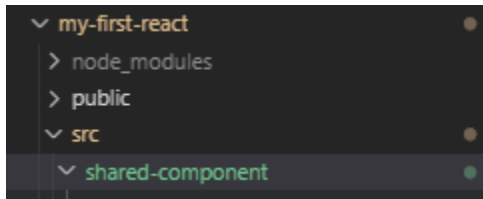
Maka tampilan aplikasi react kita akan menjadi seperti ini

- Nama: Onesinus SPT, Usia: 22
- Nama: Melendoy, Usia: 23

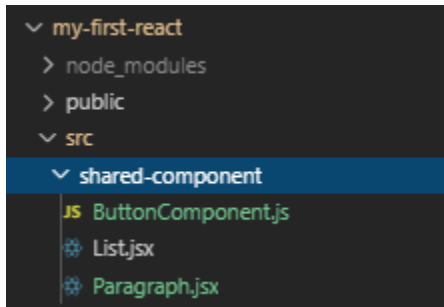
2.6 Memindahkan component ke sebuah folder

Sekarang mari kita pindahkan semua component kita ke sebuah folder agar tidak menjadi menumpuk hanya di folder **src**

Buat sebuah folder bernama shared-component di folder src



Kemudian pindahkan ButtonComponent.js dan Paragraph.js dan juga List.js ke folder shared-component, sehingga folder share-component menjadi seperti ini



Kemudian untuk import component di App.js harus kita sesuaikan menjadi seperti ini

```
my-first-react > src > JS App.js > Hello > nama
1 | import React, { useState, Component } from "react";
2 | import './App.css';
3 | import Button, { ButtonProps, ButtonDenganChildrenProps } from "./shared-component/ButtonComponent";
4 | import Paragraph from "./shared-component/Paragraph";
5 | import List from "./shared-component/List";
6 |
```

2.7 Cara mengubah functional component menjadi class component

Sekarang kita akan coba mengubah Functional component menjadi Class component pada file List.jsx

Ubah file List.jsx menjadi

```
import React, { Component } from "react";

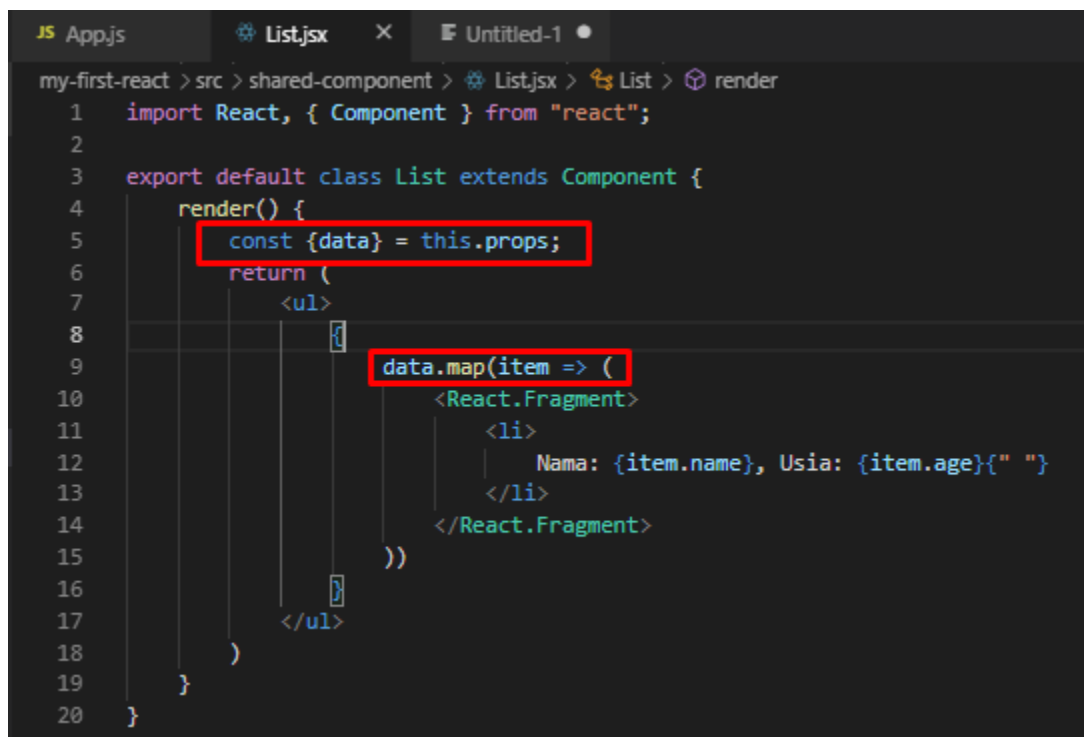
export default class List extends Component {
  render() {
    return (
      <ul>
        {
          this.props.data.map(item => (
            <React.Fragment>
              <li>
                Nama: {item.name}, Usia: {item.age}{" "}
              </li>
            </React.Fragment>
          ))
        }
      </ul>
    )
  }
}
```

Jika kita lihat ada beberapa perbedaan antara class component dengan functional component, ini code kita sebelum kita ubah

```
import React from "react";

export default function List(props){
  return (
    <ul>
      {
        props.data.map(item => (
          <React.Fragment>
            <li>
              Nama: {item.name}, Usia: {item.age}{" "}
            </li>
          </React.Fragment>
        ))
      }
    </ul>
  )
}
```

Kita bisa melakukan penyederhanaan dalam penggunaan props dengan meletakkan setiap property ke sebuah variable



```
JS App.js List.jsx x Untitled-1
my-first-react > src > shared-component > List.jsx > List > render
1 import React, { Component } from "react";
2
3 export default class List extends Component {
4   render() {
5     const {data} = this.props;
6     return (
7       <ul>
8         {
9           data.map(item => (
10            <React.Fragment>
11              <li>
12                Nama: {item.name}, Usia: {item.age}{" "}
13              </li>
14            </React.Fragment>
15          ))
16        }
17      </ul>
18    )
19  }
20 }
```

2.8 Mengembangkan component List

Ubah List.jsx menjadi seperti ini

```
import React, { Component } from "react";
import { ButtonProps } from "./ButtonComponent";

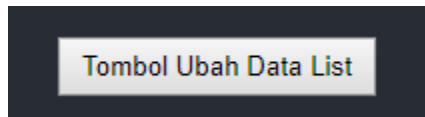
const dataBaru = [];
for (let i = 0; i < 10; i++) {
  dataBaru.push({
    key: i,
    name: 'One',
    usia: `Usia ke ${i}`
  })
}

export default class List extends Component {
  state = { nama: "Andry Bryan", data: [] };

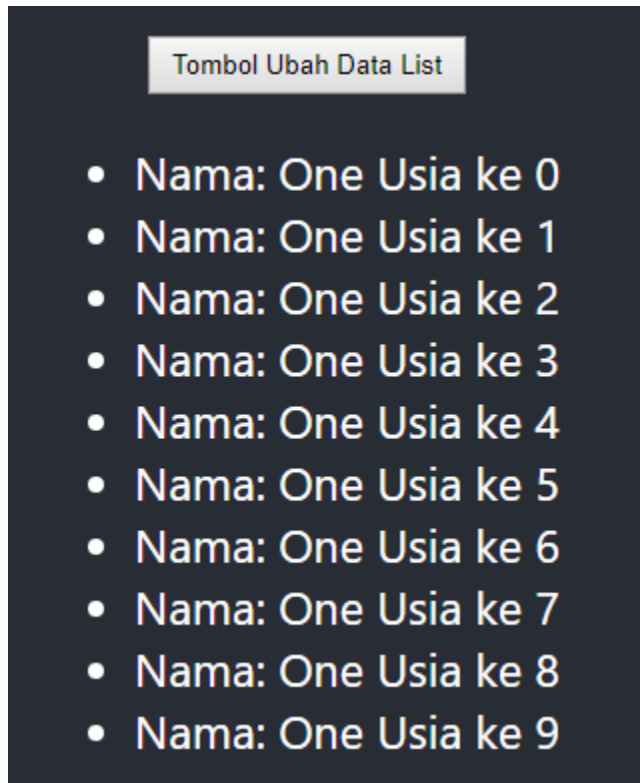
  handleChange = () => {
    this.setState({ data: dataBaru });
  }

  render() {
    const { data } = this.state;
    return (
      <div>
        <ButtonProps nama="Tombol Ubah Data List" onClick={this.handleChange} />
        <ul>
          {
            data.map(item => (
              <li key={item.key}>
                Nama: {item.name} {item.usia}{ " "}
              </li>
            ))
          }
        </ul>
      </div>
    )
  }
}
```

Sekarang lihat aplikasi kita, akan ada sebuah tombol bernama “Tombol Ubah Data List”



Yang jika kita klik akan mengubah data ke List component yang kita buat



Code diatas adalah contoh membuat state sederhana dengan class component, dan mengubah nilai state dengan fungsi ketika tombol diklik

2.9 Memindahkan tombol di List.jsx ke App.js (Memindahkan logic dari class component ke functional component)

Pada file List.jsx ubah code menjadi seperti ini

```
import React, { Component } from "react";
import { ButtonProps } from "../ButtonComponent";

export default class List extends Component {
  render() {
    const { data } = this.props;
    return (
      <div>
        <ul>
          {
            data.map(item => (
              <li key={item.key}>
                Nama: {item.name} {item.usia}{" "}
              </li>
            ))
          }
        </ul>
      </div>
    )
  }
}
```


Sekarang ubah code App.js sehingga menjadi seperti ini

```
JS App.js x Listjsx
my-first-react > src > JS App.js > App
1 | import React, { useState, Component } from "react";
2 | import './App.css';
3 | import Button, { ButtonProps, ButtonDenganChildrenProps } from "../shared-component/ButtonComponent";
4 | import Paragraph from "../shared-component/Paragraph";
5 | import List from "../shared-component/List";
6 |
7 | const dataBaru = [];
8 | for (let i = 0; i < 10; i++) {
9 |   dataBaru.push({
10 |     key: i,
11 |     name: 'One',
12 |     usia: `Usia ke ${i}`
13 |   })
14 | }
15 |
16 | export default function App() {
17 |   const data = [{ name: "Onesinus SPT", age: 22 }, { name: "Melendoy", age: 23 }];
18 |
19 |   const [dataList, setDataList] = useState([]);
20 |
21 |   function onClickTombolQue() {
22 |     console.log("tombol que diklik");
23 |   }
24 |
25 |   function handleChange() {
26 |     setDataList(dataBaru)
27 |   }
28 | }
```

Tidak ada perubahan untuk code ini, karena ini basic javascript

Penyimpanan data functional component

Perubahan fungsi ke functional component

```
return (
  <div className="App">
    <header className="App-header">
      <Hello />
      <HelloClass />
      <Button />
      <ButtonProps nama="TombolQue" color="green" onClick={onClickTombolQue} />
      <ButtonDenganChildrenProps>
        <div>Test</div>
      </ButtonDenganChildrenProps>
      <Paragraph nama="Santuy" jabatan="Gabut Manager">
        <div>Children dari Paragraph Component</div>
      </Paragraph>
      <ButtonProps nama="Tombol Ubah Data List" onClick={handleChange} />
      <List data={dataList} />
    </header>
  </div>
)
```

Kenapa kita harus memindahkan code Tombol kita dari List.jsx? jawabannya adalah karena fungsi dari component di List.jsx adalah hanya untuk menampilkan list, sedangkan untuk menambahkan button component kita dapat menambahkannya kemana yang membutuhkan saja bukan disemua yang memanggil List component

Secara keseluruhan App.js saat ini akan menjadi seperti ini

```
import React, { useState, Component } from "react";
import './App.css';
import Button, { ButtonProps, ButtonDenganChildrenProps } from "./shared-component/ButtonComponent";
import Paragraph from "./shared-component/Paragraph";
import List from "./shared-component/List";

const dataBaru = [];
for (let i = 0; i < 10; i++) {
  dataBaru.push({
    key: i,
    name: 'One',
    usia: `Usia ke ${i}`
  })
}

export default function App() {
  const data = [{ name: "Onesinus SPT", age: 22 }, { name: "Melendoy", age: 23 }];
  ;

  const [dataList, setDataList] = useState([]);

  function onClickTombolQue() {
    console.log("tombol que diklik");
  }

  function handleChange() {
    setDataList(dataBaru)
  }

  return (
    <div className="App">
      <header className="App-header">
        <Hello />
        <HelloClass />
        <Button />
        <ButtonProps nama="TombolQue" color="green" onClick={onClickTombolQue} />
        <ButtonDenganChildrenProps>
          <div>Test</div>
        </ButtonDenganChildrenProps>
        <Paragraph nama="Santuy" jabatan="Gabut Manager">
          <div>Children dari Paragraph Component</div>
        </Paragraph>
      </div>
    )
  )
}
```

```

        <ButtonProps nama="Tombol Ubah Data List" onClick={handleChange} />
        <List data={dataList} />
    </header>
</div>
)
}

function Hello() {
    const [nama, setNama] = useState("Lesiong");
    const [jabatan, setJabatna] = useState("Santuy Senior");
    const [usia, setUsia] = useState(41);
    return (
        <div>
            <h1>Hello {nama}</h1>
            <p>jabatan anda: {jabatan}</p>
            <p>usia anda: {usia}</p>
        </div>
    )
}

class HelloClass extends Component {
    constructor(props) {
        super(props);
        this.state = {
            nama: "Santuy",
            jabatan: "Melendoy developer",
            usia: 14
        }
    }
}

render() {
    const { nama, jabatan, usia } = this.state;
    return (
        <div>
            <h1>Hello {nama}</h1>
            <p>jabatan anda: {jabatan}</p>
            <p>usia anda: {usia}</p>
        </div>
    )
}
}

```

2.10 Membuat Component Input

Buat sebuah file didalam folder shared-component dengan nama Input.jsx

```
import React from "react";
export default function Input(props){
  const styleInput = {
    padding: "5px",
    border: "2px solid orange"
  }
  return(
    <input style={styleInput} onChange={props.onChange} />
  )
}
```

Kemudian lakukan langkah yang sama pada bagian-bagian sebelumnya yaitu import component Input, dan panggil component Input pada App.js

```
import Input from "../shared-component/Input";
```

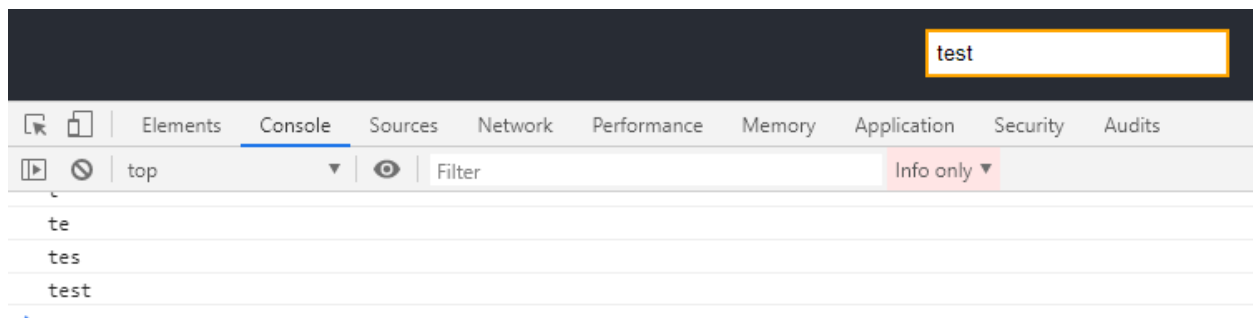
Pada bagian fungsi

```
function onChangeInput(e){
  console.log(e.target.value);
}
```

Pada bagian return

```
<Input onChange={onChangeInput}/>
```

Jika kita lihat hasilnya dibrowser



Jika ingin menampung value yang diinput maka buat sebuah state

```
const [valueInput, setValueInput] = useState("User belum melakukan input");
```

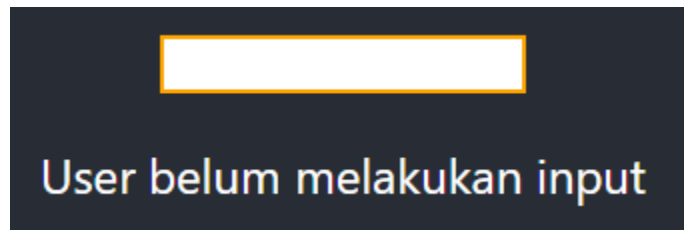
kemudian ubah fungsi onChangeInput menjadi seperti ini

```
function onChangeInput(e){  
  console.log(e.target.value);  
  setValueInput(e.target.value);  
}
```

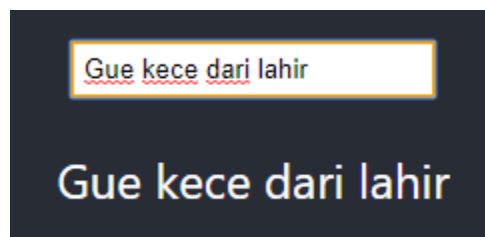
Kemudian dibagian return tambahkan sebuah tag <p> untuk menampilkan apa yang user input

```
<Input onChange={onChangeInput}/>  
<p>{valueInput}</p>
```

Maka tampilannya akan seperti ini



Ketika kita input



Tag <p> akan berubah menyesuaikan apa yang diinput

2.11 Membuat event handler ketika menekan key enter di inputan

Kita juga bisa mengubah state lain, misalnya saat inputan ditekan enter, dataList yang sudah kita buat sebelumnya akan bertambah sesuai inputan yang kita isi

Pertama, tambahkan event onKeyDown pada pemanggilan component Input

```
<Input onChange={onChangeInput} onKeyDown={onEnterInput} />
```

Kemudian pada Input.jsx bagian return juga ubah menjadi seperti ini

```
<input style={styleInput} onChange={props.onChange} onKeyDown={props.onKeyDown} />
```

Untuk fungsi onEnterInput buat fungsi seperti ini

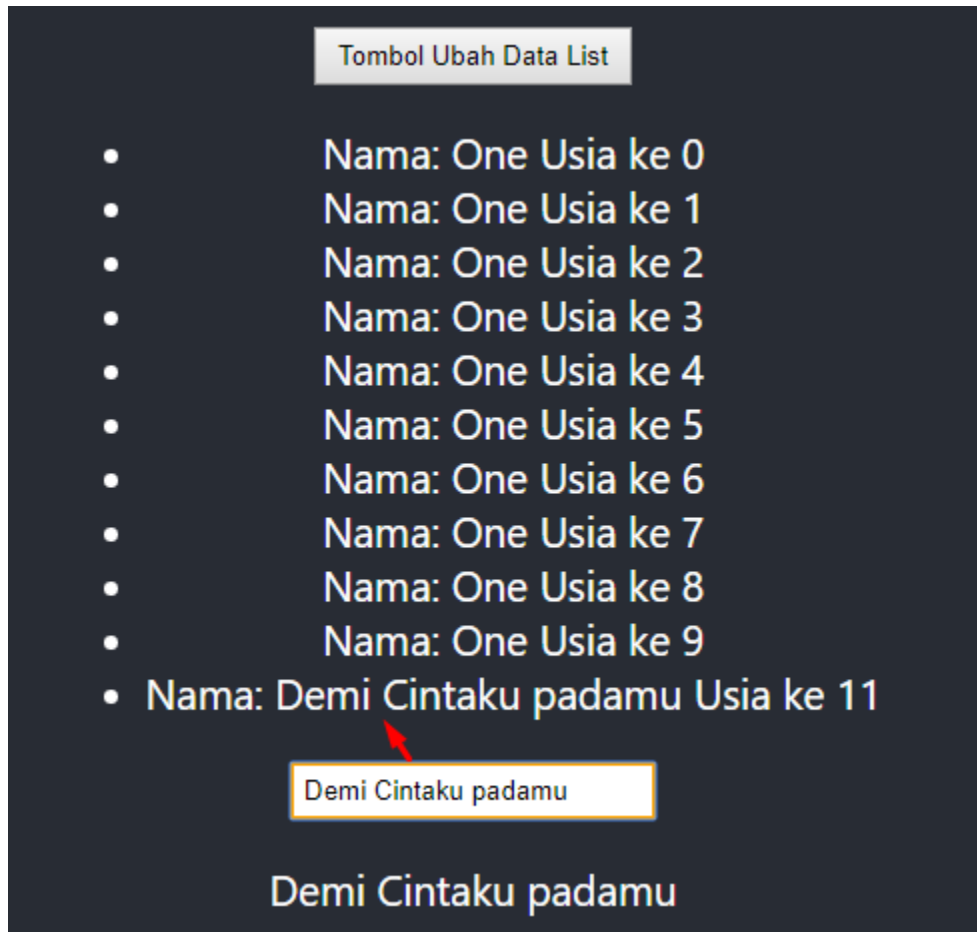
```
function onEnterInput(e) {  
  if (e.key == "Enter") {  
    let gabunganData = dataList.concat([ {  
      key: dataList.length + 1,  
      name: e.target.value,  
      usia: `Usia ke ${dataList.length + 1}`  
    } ] );  
    setDataList(gabunganData);  
  }  
}
```

Maka ketika kita input dan tekan enter data pada component List akan bertambah, ini adalah tampilan sebelum kita input dan tekan enter

- Nama: One Usia ke 0
- Nama: One Usia ke 1
- Nama: One Usia ke 2
- Nama: One Usia ke 3
- Nama: One Usia ke 4
- Nama: One Usia ke 5
- Nama: One Usia ke 6
- Nama: One Usia ke 7
- Nama: One Usia ke 8
- Nama: One Usia ke 9

User belum melakukan input

Ketika input data dan tekan enter



Jika ingin menambahkan sebaliknya (bukan nambah ke akhir data, tetapi menambahkan ke data pertama) kita dapat menukar posisi concatnya, sehingga fungsi onEnterInput menjadi seperti ini

```
function onEnterInput(e) {  
  if (e.key == "Enter") {  
    let gabunganData = [{  
      key: dataList.length + 1,  
      name: e.target.value,  
      usia: `Usia ke ${dataList.length + 1}`  
    }].concat(dataList);  
    setdataList(gabunganData);  
  }  
}
```

Perbedaannya hanya variable **dataList** terletak setelah **concat()** yang mana sebelumnya disebelum **concat()**

Jika kita menambahkan data (dengan cara input dan tekan enter), data akan menambah ke atas

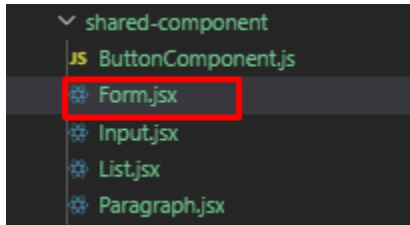
- Nama: lexiong Usia ke 4
- Nama: santuy Usia ke 3
- Nama: testtest Usia ke 2
- Nama: test Usia ke 1

- Nama: Tambahkan keatas Usia ke 5
- Nama: lexiong Usia ke 4
- Nama: santuy Usia ke 3
- Nama: testtest Usia ke 2
- Nama: test Usia ke 1

Tambahkan keatas

2.12 Membuat component Form

Buat sebuah file didalam folder shared-component bernama Form.jsx dan buat code seperti ini



```
import React from "react";

export default function Form(props) {
  return (
    <React.Fragment>
      {props.children.map(child => {
        if (child.type === "label") {
          return <Label nama={child.props.children} />
        } else {
          return child;
        }
      })}
    </React.Fragment>
  );
}

function Label(props) {
  let style = {
    color: "gray",
    fontSize: "11pt"
  }
  return (
    <div>
      <label style={style}>{props.nama}</label>
    </div>
  )
}
```

Import form dan gunakan component di App.js

```
import Form from "../shared-component/Form";
```

Buat sebuah hooks untuk menampung data dari form

```
const [arrForm, setArrForm] = useState({});
```

Buat fungsi untuk handle perubahan pada input, dan mengumpulkan, agar ketika tombol simpan diklik akan mendapatkan semua value yang dibutuhkan

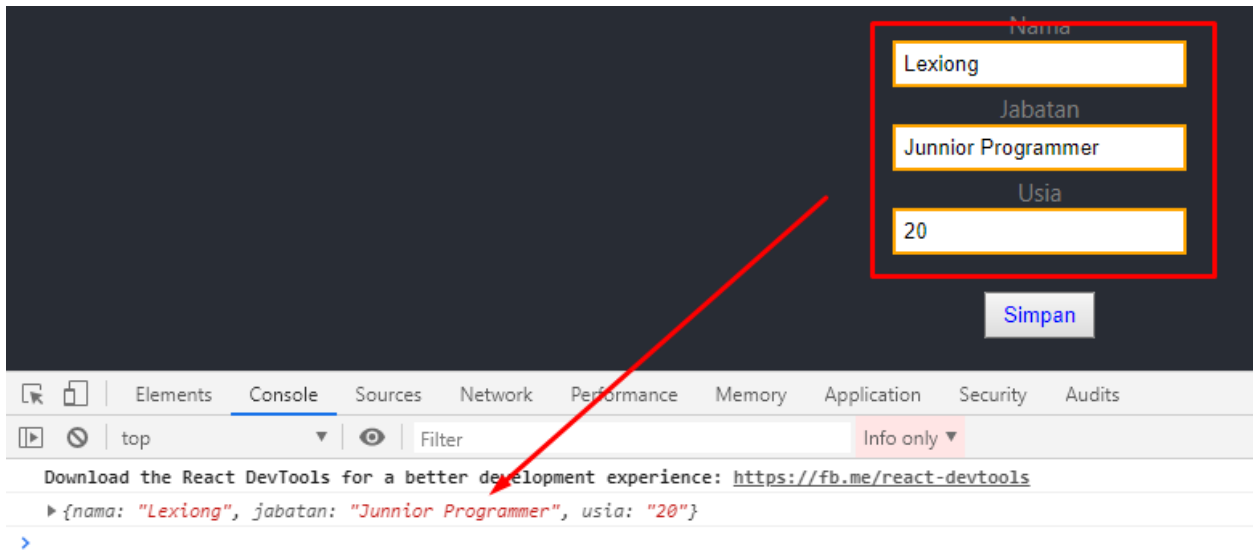
```
function onClickBtnSimpan(){
  console.log(arrForm);
}

function onChangeFormInput(field, e){
  let dataForm = arrForm;
  dataForm[field] = e.target.value;
  setArrForm(dataForm);
}
```

Kemudian untuk App.js dibagian return gunakan component form, dan isi child seperti ini

```
<Form>
  <label>Nama</label>
  <Input onChange={e => onChangeFormInput("nama", e)} />
  <label>Jabatan</label>
  <Input onChange={e => onChangeFormInput("jabatan", e)} />
  <label>Usia</label>
  <Input onChange={e => onChangeFormInput("usia", e)} />
  <br/>
  <ButtonProps nama="Simpan" color="blue" onClick={onClickBtnSimpan} />
  <p>Data dari form</p>
</Form>
```

Maka akan tampil seperti ini



2.13 Membuat component dengan property object

Sebuah component bila terlalu banyak memiliki property akan membuat sulit dan menyebalkan ketika memanggil component tersebut, padahal kita bisa saja menyatukan property-property yang dibutuhkan component itu kedalam sebuah object, untuk itu mari kita coba membuat sebuah component di file ButtonComponent.js dengan menambahkan sebuah functional component bernama ButtonWithObjectProps

```
function ButtonWithObjectProps(props){  
  return <button style={props.style}>{props.name}</button>  
}
```

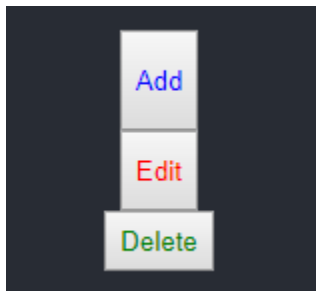
```
JS App.js JS ButtonComponent.js X
my-first-react > src > shared-component > JS ButtonComponent.js > ButtonWithObjectProps
13   const style = {
14     padding: "5px 10px",
15     color: props.color,
16     border: 'solid 2px ${props.color}'
17   }
18   return <button style={style} onClick={props.onClick}>{props.nama}</button>
19 }
20
21 function ButtonDenganChildrenProps(props) {
22   const style = {
23     padding: "5px 10px",
24     color: props.color,
25     border: "solid 2px ${props.color}"
26   }
27   return <button style={style}>{props.children}</button>
28 }
29
30 function ButtonWithObjectProps(props) {
31   return <button style={props.style}>{props.name}</button>
32 }
33
34 export default Button;
35 export {ButtonProps, ButtonDenganChildrenProps, ButtonWithObjectProps}
```

Kemudian coba panggil component ButtonWithObjectProps di App.js

```
<ButtonWithObjectProps style={{color: "blue", name: "Add"}} />
<ButtonWithObjectProps style={{color: "red", name: "Edit"}} />
<ButtonWithObjectProps style={{color: "green", name: "Delete"}} />
```

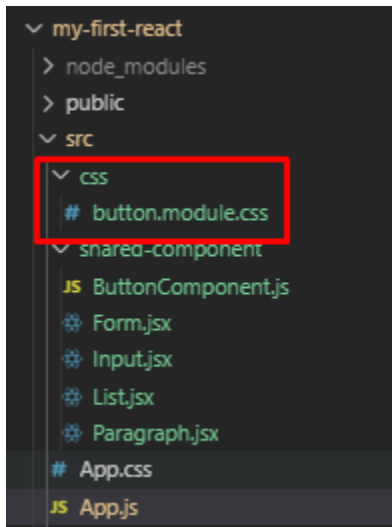
```
JS App.js x JS ButtonComponent.js
my-first-react > src > JS App.js > App
69 </ButtonDenganChildrenProps>
70 <Paragraph nama="Santuy" jabatan="Gabut Manager">
71   <div>Children dari Paragraph Component</div>
72 </Paragraph>
73
74 <ButtonProps nama="Tombol Ubah Data List" onClick={handleChange} />
75 <List data={dataList} />
76 <Input onChange={onChangeInput} onKeyDown={onEnterInput} />
77 <p>{valueInput}</p>
78
79 <Form>
80   <label>Nama</label>
81   <Input onChange={(e) => onChangeFormInput("nama", e)} />
82   <label>Jabatan</label>
83   <Input onChange={(e) => onChangeFormInput("jabatan", e)} />
84   <label>Usia</label>
85   <input type="text" onChange={(e) => onChangeFormInput("usia", e)} />
86   <br />
87   <ButtonProps nama="Simpan" color="blue" onClick={onClickBtnSimpan} />
88   <p>Data dari form</p>
89 </Form>
90 <ButtonWithObjectProps name="Add" style={{color: "blue", height: "50px"}} />
91 <ButtonWithObjectProps name="Edit" style={{color: "red", height: "40px"}} />
92 <ButtonWithObjectProps name="Delete" style={{color: "green", height: "30px"}} />
93 <br /><br />
```

Maka hasilnya akan menjadi seperti ini



2.14 Membuat component button fixed style dengan sebuah property pembeda (Using external Css)

Buat folder css didalam src, dan buat sebuah file bernama button



Kemudian isi file button.module.css menjadi seperti ini

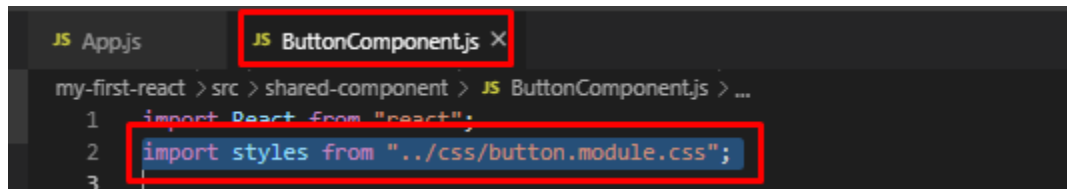
```
.btn{
  border: 3px solid purple;
}

.btn_add{
  color: green;
}

.btn_edit{
  color: blue;
}

.btn_delete{
  color: yellow;
}
```

Pada ButtonComponent.js import css external tersebut



```
JS App.js JS ButtonComponent.js X
my-first-react > src > shared-component > JS ButtonComponent.js > ...
1 import React from "react";
2 import styles from "../css/button.module.css";
3
```

Kemudian buat sebuah functional component masih di file yang sama yaitu ButtonComponent.js

```
function ButtonFixedWithType({type, onClick}){
  const addStyle = `${styles.btn} ${styles.btn_add}`;
  const editStyle = `${styles.btn} ${styles.btn_edit}`;
  const deleteStyle = `${styles.btn} ${styles.btn_delete}`;

  const typeOfButton = [
    {
      type: "add",
      name: "Add",
      style: addStyle
    },
    {
      type: "edit",
      name: "Edit",
      style: editStyle
    },
    {
      type: "delete",
      name: "Delete",
      style: deleteStyle
    }
  ];

  const renderButton = typeOfButton.map(item => {
    if(item.type === type){
      return(
        <button className={item.style} onClick={onClick} key={item.name}>
          {item.name}
        </button>
      )
    }
  })
}

return <React.Fragment>{renderButton}</React.Fragment>
}
```

Jangan lupa juga untuk melakukan export pada functional component yang baru saja kita buat

```
JS App.js JS ButtonComponent.js X
my-first-react > src > shared-component > JS ButtonComponent.js > ...
50     style: editStyle
51   },
52   {
53     type: "delete",
54     name: "Delete",
55     style: deleteStyle
56   },
57 ];
58
59 const renderButton = typeOfButton.map(item => {
60   if(item.type === type){
61     return(
62       <button className={item.style} onClick={onClick} key={item.name}>
63         {item.name}
64       </button>
65     )
66   }
67 })
68
69 return <React.Fragment>{renderButton}</React.Fragment>
70 }
71
72 export default Button;
73 export {ButtonProps, ButtonDenganChildrenProps, ButtonWithObjectProps, ButtonFixedWithType}
```

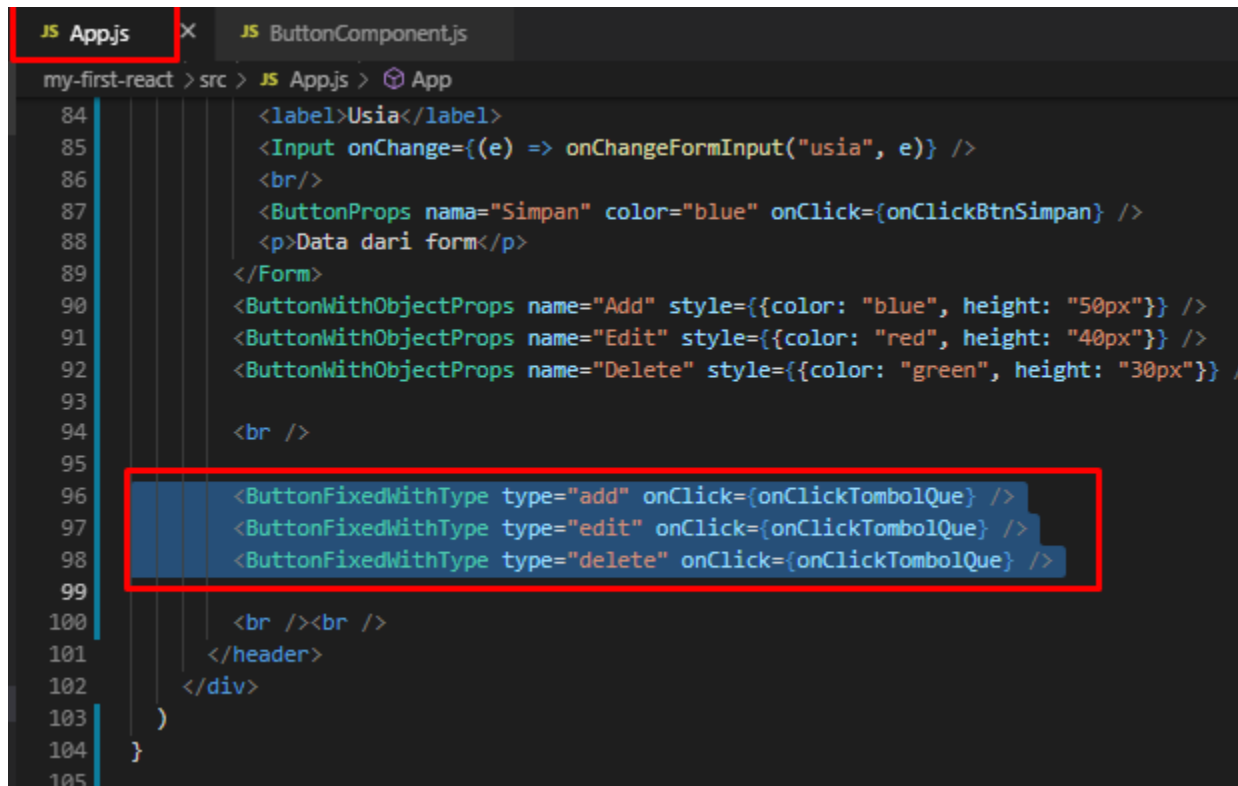
Sekarang mari kita coba panggil component tersebut di App.js

Import terlebih dahulu componentnya

```
JS App.js X JS ButtonComponent.js
my-first-react > src > JS App.js > ...
1  import React, { useState, useEffect, Component } from "react";
2  import './App.css';
3  import Button, { ButtonProps, ButtonDenganChildrenProps, ButtonWithObjectProps, ButtonFixedWithType }
4  import Paragraph from './shared-component/Paragraph';
5  import List from './shared-component/List';
6  import Input from './shared-component/Input';
7  import Form from './shared-component/Form';
8
9  const dataBaru = [];
10 for (let i = 0; i < 10; i++) {
11   dataBaru.push({
12     key: i,
13     name: 'One',
14     usia: `Usia ke ${i}`
15   })
16 }
17
18 export default function App() {
19   const data = [{ name: "Onesinus SPT", age: 22 }, { name: "Melendoy", age: 23 }];
20
21   const [dataList, setdataList] = useState([]);
22
23   const [valueInput, setValueInput] = useState("User belum melakukan input");
24
25   const [arrForm, setArrForm] = useState({});
```


Kemudian panggil componentnya

```
<ButtonFixedWithType type="add" onClick={onClickTombolQue} />  
<ButtonFixedWithType type="edit" onClick={onClickTombolQue} />  
<ButtonFixedWithType type="delete" onClick={onClickTombolQue} />
```



```
JS App.js x JS ButtonComponent.js  
my-first-react > src > JS App.js > App  
84 <label>Usia</label>  
85 <Input onChange={(e) => onChangeFormInput("usia", e)} />  
86 <br />  
87 <ButtonProps nama="Simpan" color="blue" onClick={onClickBtnSimpan} />  
88 <p>Data dari form</p>  
89 </Form>  
90 <ButtonWithObjectProps name="Add" style={{color: "blue", height: "50px"}} />  
91 <ButtonWithObjectProps name="Edit" style={{color: "red", height: "40px"}} />  
92 <ButtonWithObjectProps name="Delete" style={{color: "green", height: "30px"}} />  
93  
94 <br />  
95  
96 <ButtonFixedWithType type="add" onClick={onClickTombolQue} />  
97 <ButtonFixedWithType type="edit" onClick={onClickTombolQue} />  
98 <ButtonFixedWithType type="delete" onClick={onClickTombolQue} />  
99  
100 <br /><br />  
101 </header>  
102 </div>  
103 )  
104 }  
105
```

Maka tampilan akan menjadi seperti ini



2.15 Mengubah component List menjadi lebih dinamis

Dikarenakan component list kita saat ini masih bersifat static, yang mana kita hanya membatasi component kita hanya pada 2 property yaitu nama dan usia

```
import React, { Component } from "react";
import { ButtonProps } from "./ButtonComponent";

export default class List extends Component {
  render() {
    const { data } = this.props;
    return (
      <div>
        <ul>
          {
            data.map(item => (
              <li key={item.key}>
                Nama: {item.name} {item.usia}{" "}
              </li>
            ))
          }
        </ul>
      </div>
    )
  }
}
```

Sedangkan pada saat memanggil component dan mengirimkan data sebagai property, bisa saja kita melempar lebih dari 2 property

```
<List data={dataList} />
```

Yang mana isi dari variable dataList adalah array dari hasil loop data kita sebelumnya

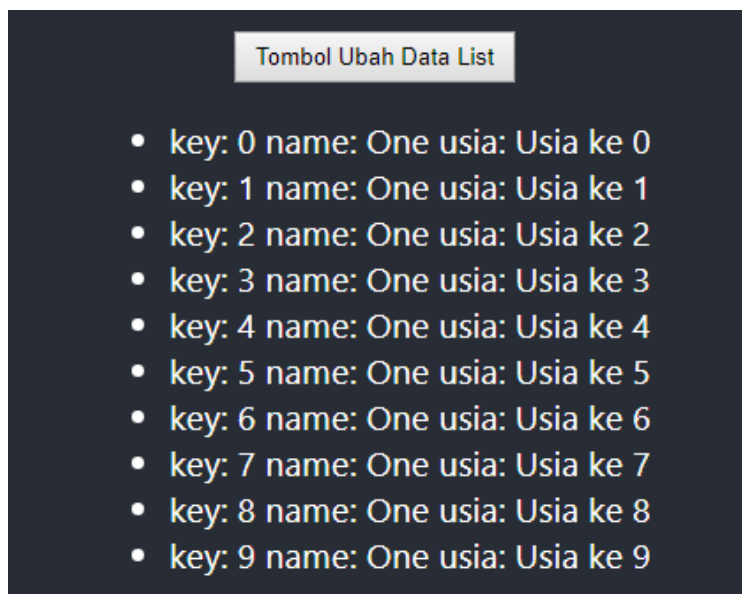
```
const dataBaru = [];
for (let i = 0; i < 10; i++) {
  dataBaru.push({
    key: i,
    name: 'One',
    usia: `Usia ke ${i}`
  })
}
```

Sekarang mari kita ubah List.jsx menjadi seperti ini

```
import React, { Component } from "react";
import { ButtonProps } from "../ButtonComponent";

export default class List extends Component {
  render() {
    const { data } = this.props;
    let list = [];
    for (var idx_data in data) {
      let list_string = ""
      Object.keys(data[idx_data]).forEach(function (key, idx_key) {
        list_string += `${key}: ${data[idx_data][key]} `
      })
      list.push(
        <li key={data.key}>{list_string}</li>
      )
    }
    return (
      <div>
        <ul>{list}</ul>
      </div>
    )
  }
}
```

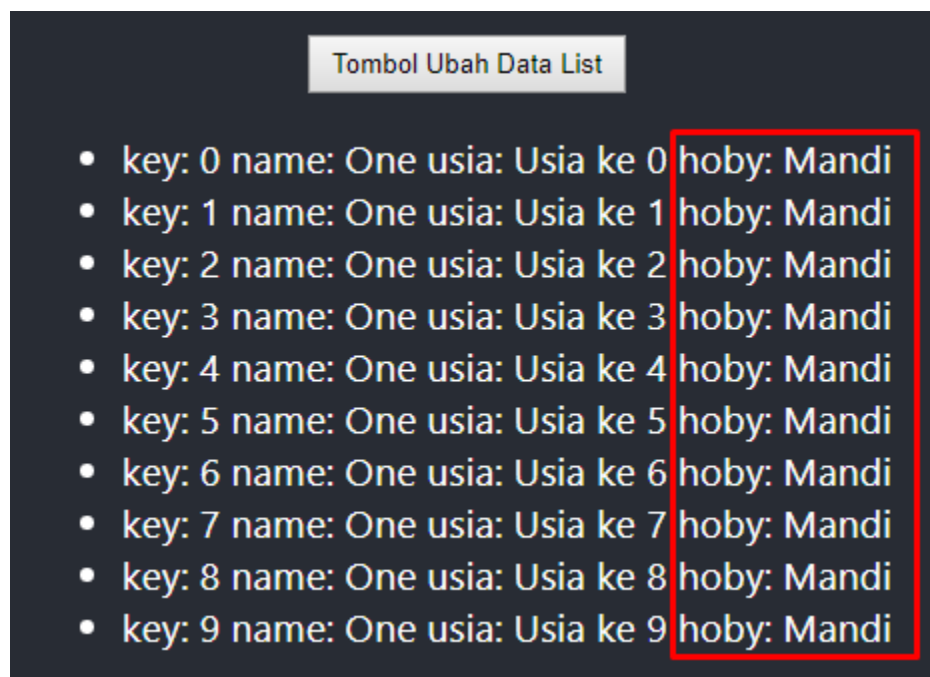
Maka tampilan kita akan sama seperti awal



Tetapi jika kita menambahkan sebuah key lagi didalam object yang menjadi property kita ke component List

```
const dataBaru = [];  
for (let i = 0; i < 10; i++) {  
  dataBaru.push({  
    key: i,  
    name: 'One',  
    usia: `Usia ke ${i}`,  
    hoby: 'Mandi'  
  })  
}
```

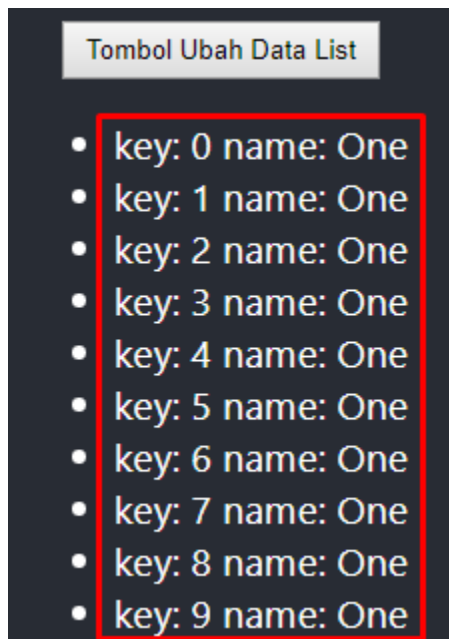
Dan kita refresh, maka tampilan akan menjadi seperti ini



Jika kita ubah menjadi hanya 2

```
const dataBaru = [];  
for (let i = 0; i < 10; i++) {  
  dataBaru.push({  
    key: i,  
    name: 'One'  
  })  
}
```

Maka tampilan akan menjadi seperti ini



Akan tetapi, coding tersebut sulit sekali dibaca dan dimengerti, maka dari itu penting bagi kita untuk membagi setiap logic kedalam function function, eits yang saya maksud bukan functional component loh, tetapi hanya sebuah fungsi javascript biasa yang kita gunakan untuk mengolah logika atau manipulasi data kita

Oke mari kita buat code diatas menjadi lebih manusiawi 😊

Ubah List.jsx code menjadi seperti ini

```
import React, { Component } from "react";
import { ButtonProps } from "../ButtonComponent";

export default class List extends Component {
  render() {
    const { data } = this.props;

    // variable ini digunakan untuk menampung element List yang akan kita ren-
    der pada return dibawah
    let list = [];
    for (var idx_data in data) {
      // Disini kita me-
      looping data yang dikirimkan melalui property pada saat pemanggilan component Lis-
      t
      list.push(<li key={data.key}>{loopKeyPadaObjectData(data)}</li>);
    }

    function loopKeyPadaObjectData() {
      // Contoh Object: { key: 1, name: 'One', usia: `Usia ke ${i}`, hob-
      y: 'Mandi' }
      let variable_object = Object.keys(data[idx_data]);

      // Fungsi ini dibuat untuk me-
      looping data object yang diberikan dan mengembalikan dalam sebuah "string" gabung-
      an keys

      let list_string = ""
      variable_object.forEach(function (key, idx_key) {
        list_string += `${key}: ${data[idx_data][key]} `
      })
      return list_string;
    }

    return (
      <div>
        <ul>{list}</ul>
      </div>
    )
  }
}
```

Code diatas masih seperti spaghetti yang “berantakan” tetapi setidaknya lebih bisa dibaca daripada code kita pertama, nah tugas kalian adalah buat versi yang lebih “readable” dan cakep 😊

Oke ada 1 contoh lagi deh codenya, kalau code kali ini kita menggunakan state dari class component ya, yang pastinya lebih better dari 2 cara sebelumnya, karena kita bisa menambahkan loading, dll

```
import React, { Component } from "react";

export default class List extends Component {
  state = {
    objectKeys: [],
    loading: true
  };

  // Lifecycle Hooks hanya diproses 1x
  componentDidMount() {
    if (this.props.data[0]) {
      this.setState({
        objectKeys: Object.keys(this.props.data[0])
      });
    }
  }

  // Kita hanya ambil key nya saja // ["key", "name", dst]

  // Lifecycle Hooks, hanya diproses pada saat
  // props atau state berubah
  componentDidUpdate(prevProps, prevState) {
    if (this.state.objectKeys.length !== prevState.objectKeys.length) {
      console.log(this.state.objectKeys);
      setTimeout(() => {
        this.setState({ loading: false });
      }, 500);
    }
  }

  render() {
    // DESTRUCTURE
    const { data } = this.props;
    const { loading, objectKeys } = this.state;

    return (
      <div>
        {loading ? (
          <h1>Loading...</h1>
        ) : (
          <ul>
```

```

        {/* {renderList()} */}
        {data.map((item) => {
            return (
                <li key={item.key}>
                    {objectKeys.map((key, index) => {
                        if (index === 0) {
                            return null;
                        }
                        return (
                            <div className="tc">
                                {objectKeys[index]}: {item[ob
jectKeys[index]]}
                            </div>
                        );
                    })}
                </li>
            );
        })}
    </ul>
    </div>
    );
}
}

```

Kita ubah juga diApp.js

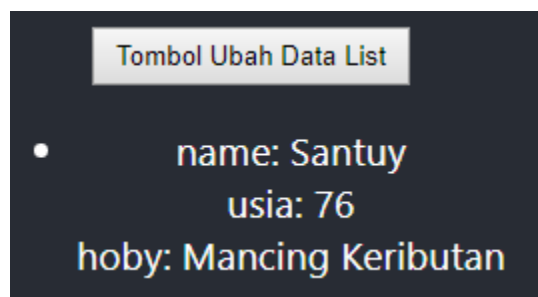
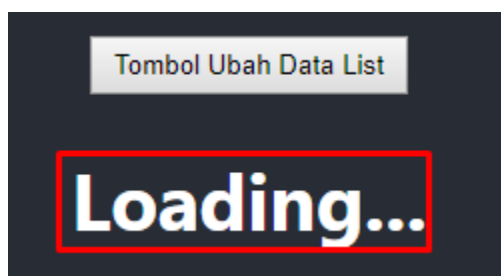
Variable dataList untuk memiliki nilai awal, ubah baris const [dataList, setdataList] menjadi

```

const [dataList, setdataList] = useState([
    {key: -1, name: "Santuy", usia: 76, hoby:"Mancing Keributan"}
]);

```

Maka tampilan akan menjadi



Ketika tombol diklik

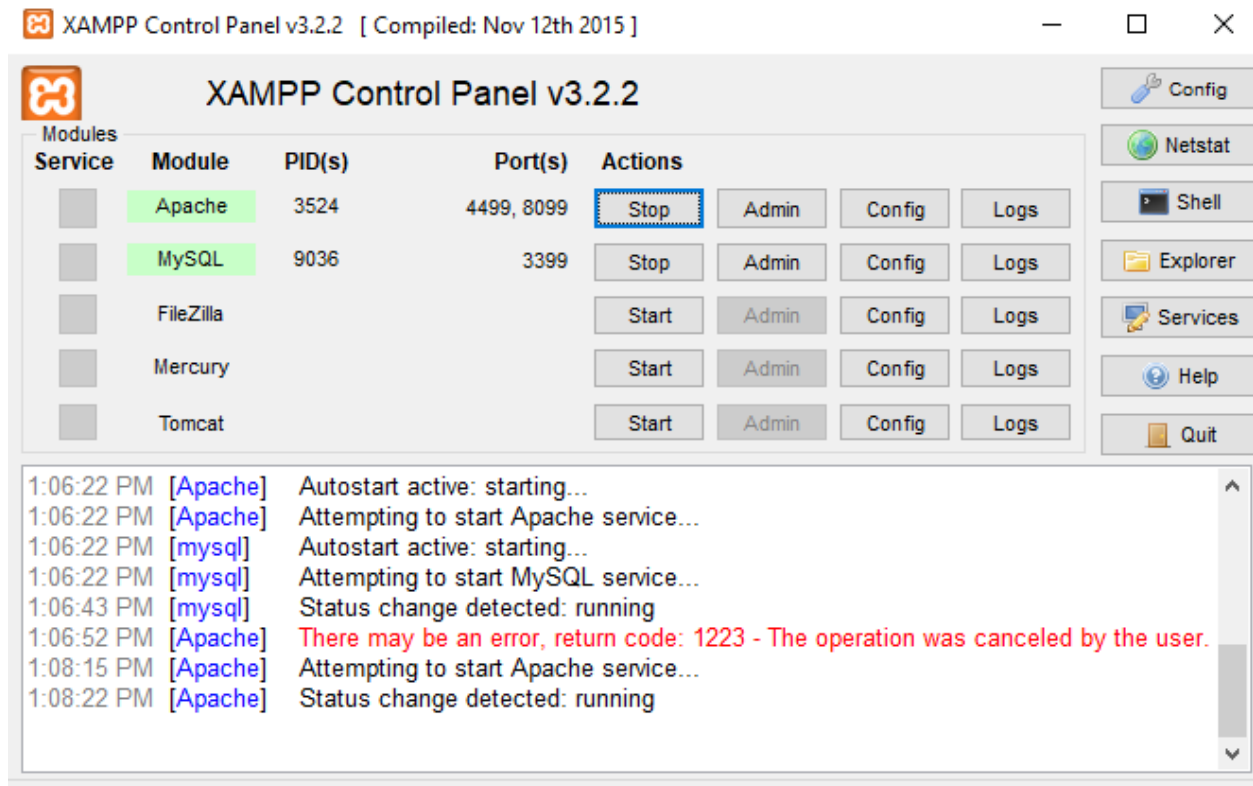
Tombol Ubah Data List

- name: One
usia: Usia ke 0
hoby: Mandi
- name: One
usia: Usia ke 1
hoby: Mandi
- name: One
usia: Usia ke 2
hoby: Mandi
- name: One
usia: Usia ke 3
hoby: Mandi

3. Persiapan Project (Setup Database, dll)

Pada bagian ini kita akan belajar sebagai persiapan untuk masuk ke project nantinya, dibagian ini akan belajar cara membuat database, table dan bagaimana cara mengambil, mengubah, menghapus data dari database.

Didalam pembelajaran ini kita akan menggunakan xampp, maka pastikan sudah ada xampp terinstall pada komputer masing-masing dan dalam keadaan sudah berjalan

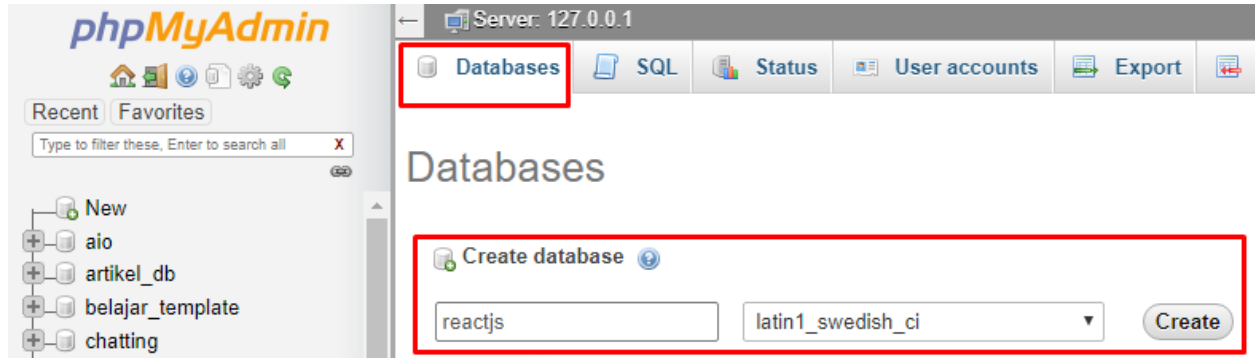


Note:

Penulis menggunakan xampp dengan menggunakan port, apache dengan port 8099 dan mysql (database) dengan port 3399, normalnya port untuk apache adalah 80 dan mysql adalah 3306, jadi pastikan menyesuaikan dengan pengaturan masing-masing

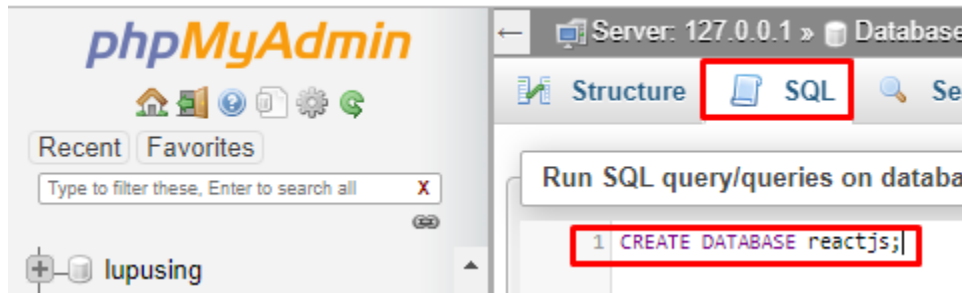
3.1 Membuat Database dan Table

Pada bagian ini kita akan membuat sebuah database dengan nama “reactjs”, anda dapat membuat database menggunakan phpmyadmin ataupun menggunakan SQL command line

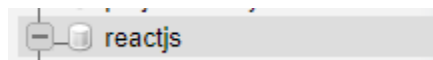


Atau bisa juga menggunakan perintah SQL

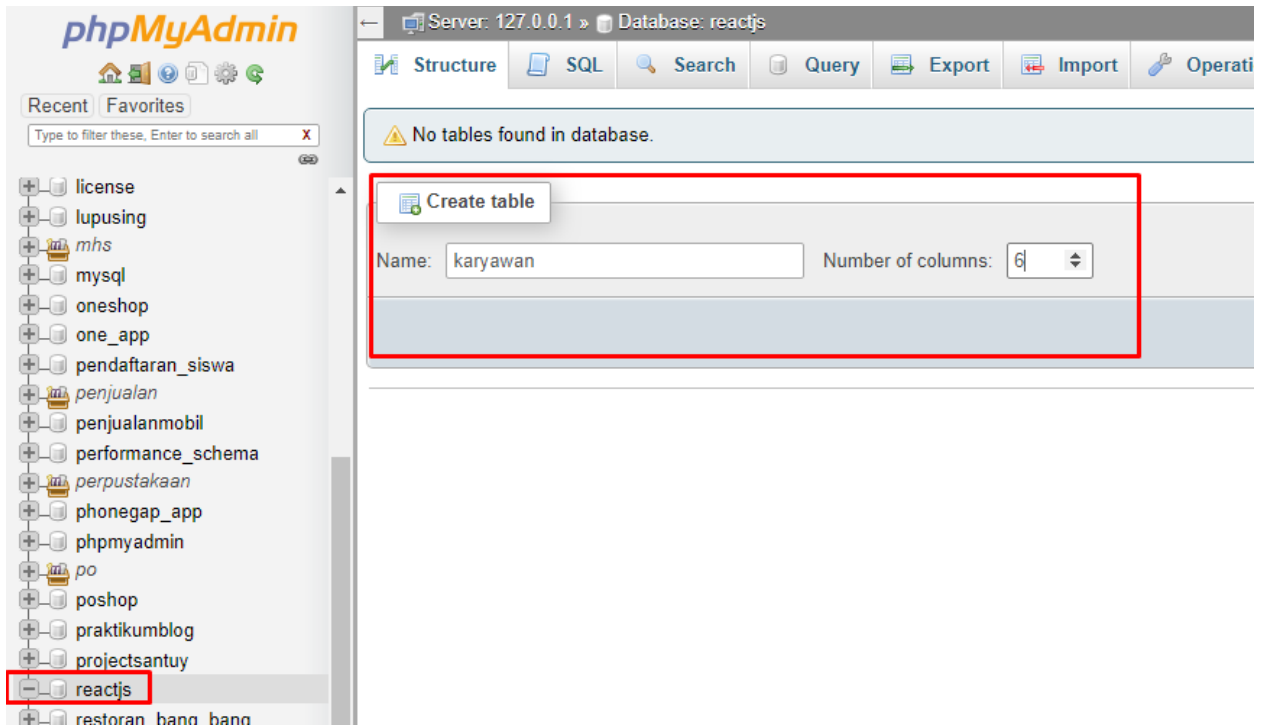
```
CREATE DATABASE reactjs;
```



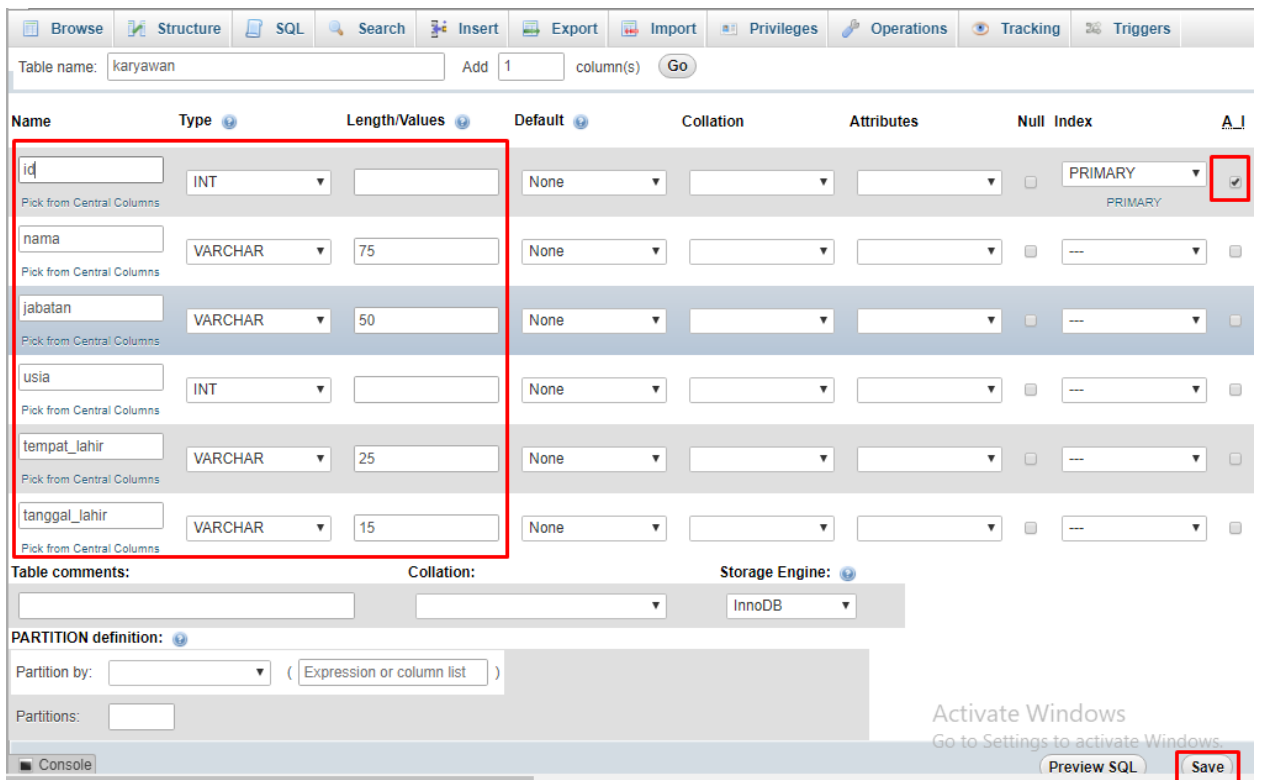
Sehingga ada database “reactjs”



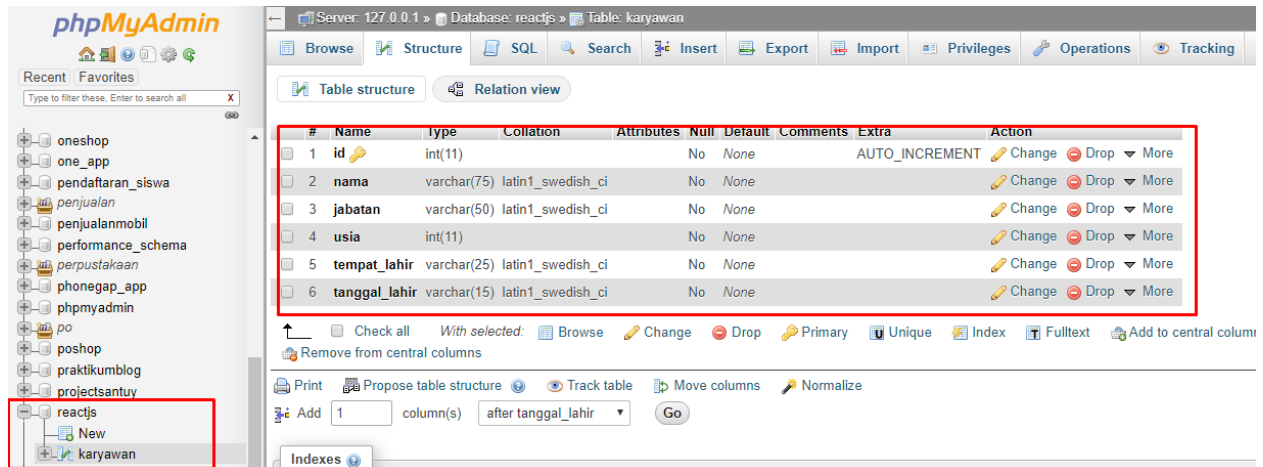
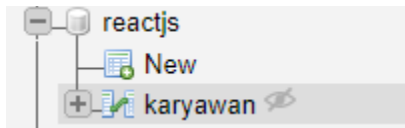
Klik database reactjs dan buat sebuah table bernama “karyawan”



Klik Tombol “Go” dipojok kanan, dan isi seperti ini pada tampilan berikutnya



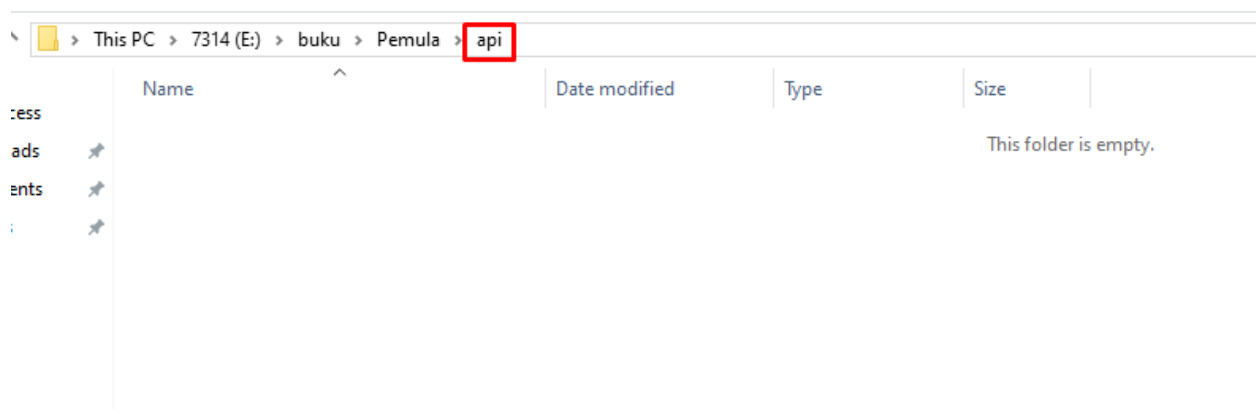
Maka akan ada table karyawan



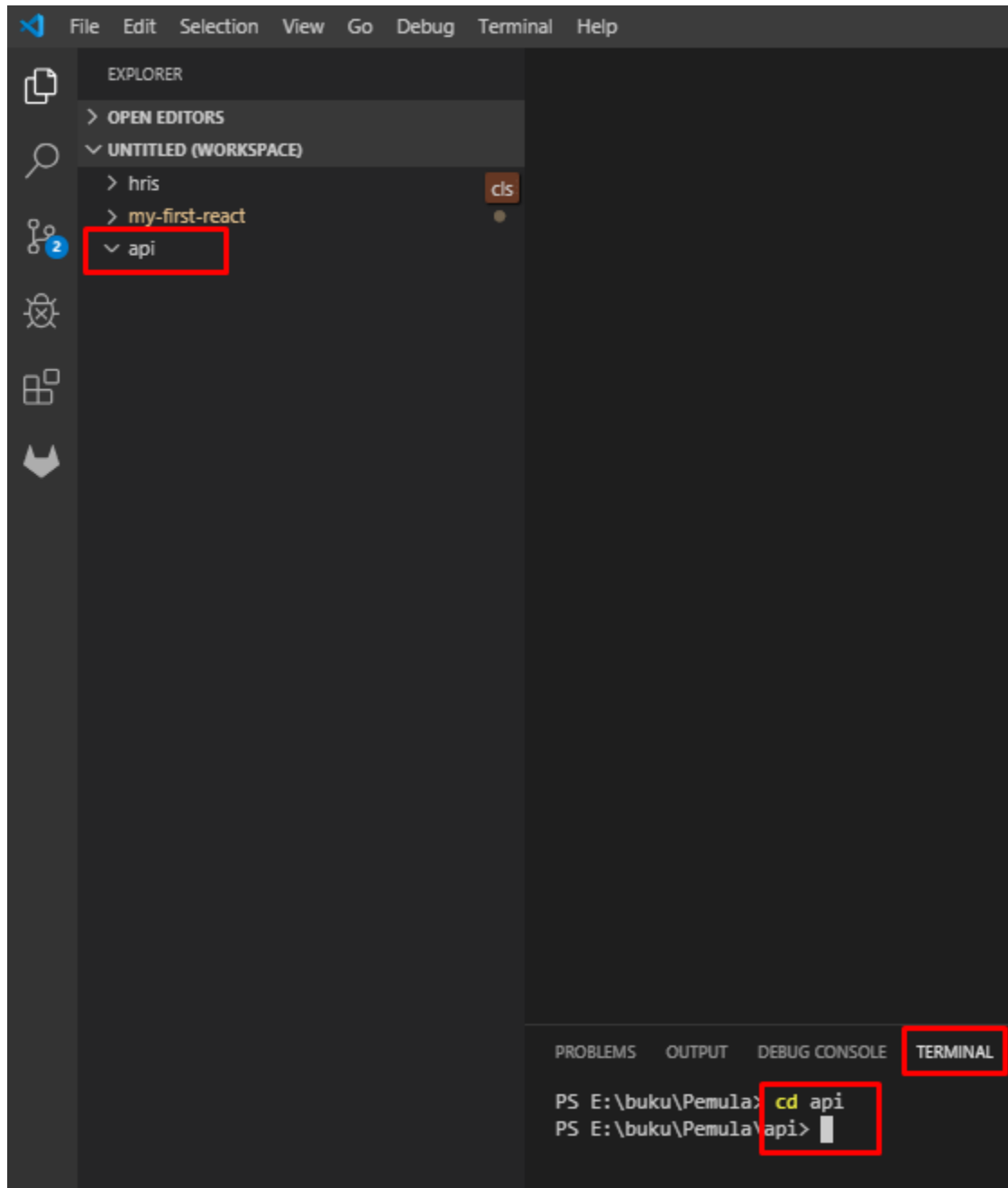
3.2 Membuat web service (Welcome & get list data)

Pada bagian ini kita akan membuat web service menggunakan nodejs/expressjs

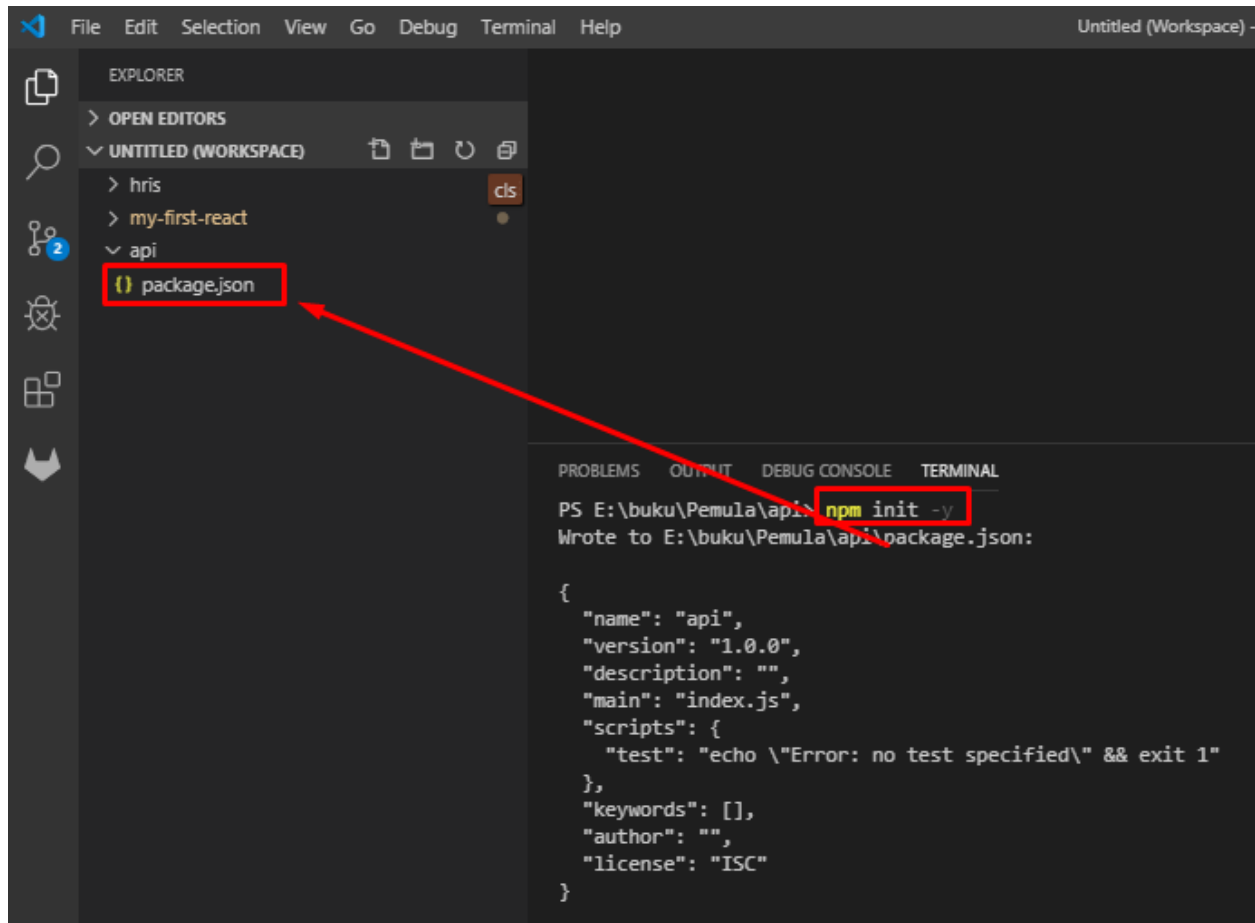
Buat sebuah folder baru bernama **api**



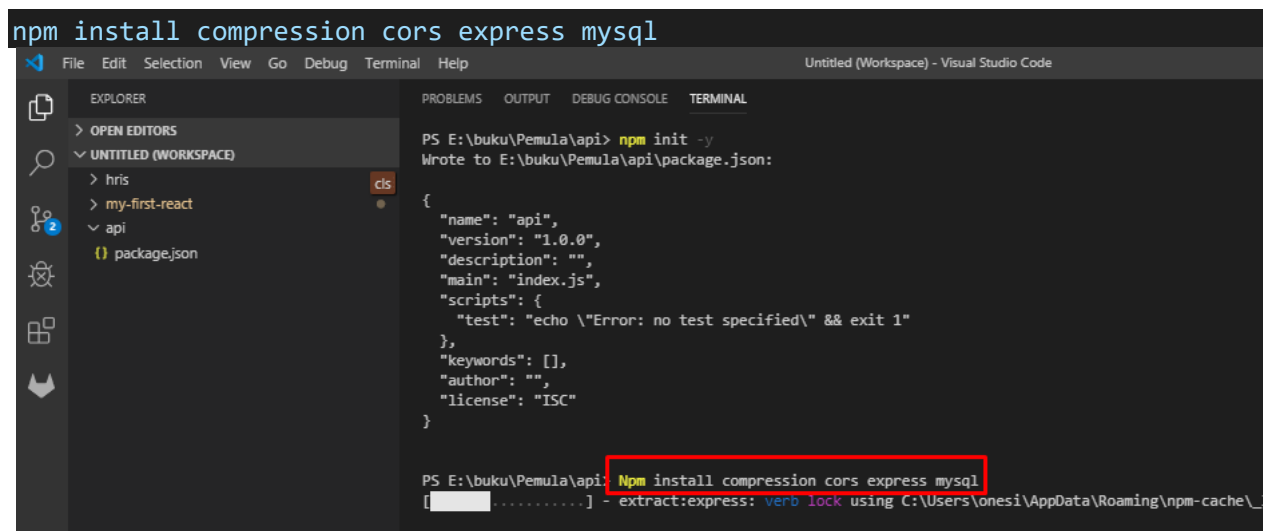
Kemudian buka folder tersebut menggunakan terminal / cmd atau visual studio code



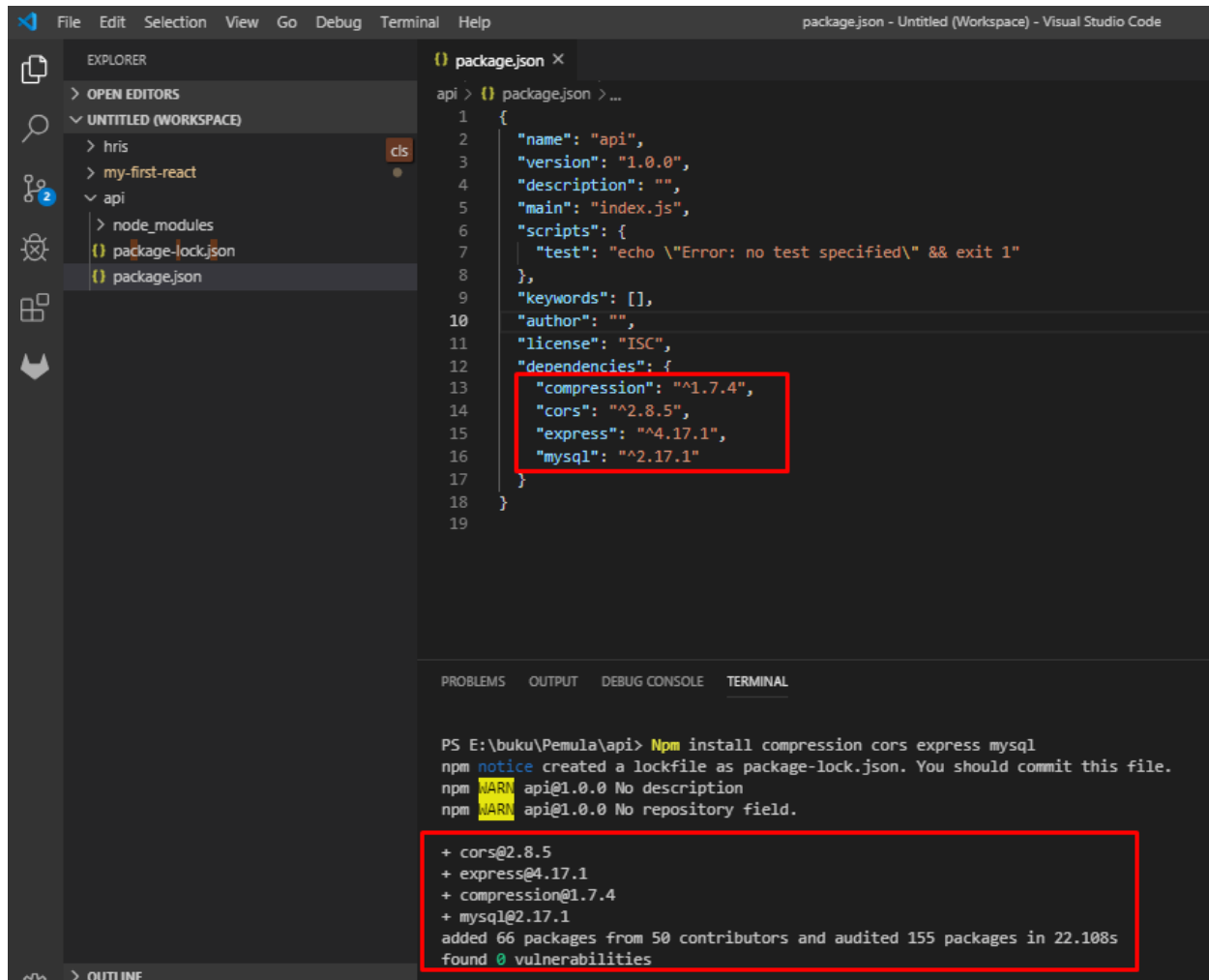
Jalankan perintah `npm init -y`



Kemudian install package yang dibutuhkan untuk membuat api seperti express dan mysql



Ketika sudah berhasil maka package.json akan menambahkan dibagian dependencies



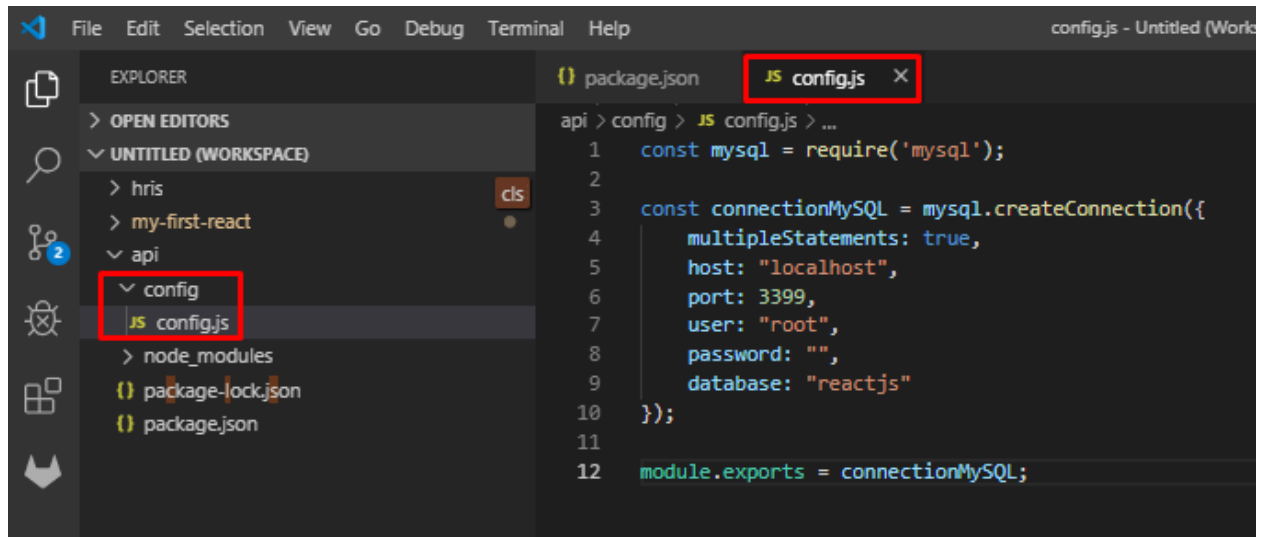
The screenshot shows the Visual Studio Code interface with the Explorer view on the left showing a project structure. The main editor displays the `package.json` file. The `dependencies` section is highlighted with a red box, showing the following content:

```
1 {
2   "name": "api",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\Error: no test specified\\ && exit 1"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "compression": "^1.7.4",
14    "cors": "^2.8.5",
15    "express": "^4.17.1",
16    "mysql": "^2.17.1"
17  }
18 }
```

The terminal at the bottom shows the command `npm install compression cors express mysql` and its output, which is also highlighted with a red box:

```
PS E:\buku\Pemula\api> Npm install compression cors express mysql
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN api@1.0.0 No description
npm WARN api@1.0.0 No repository field.
+ cors@2.8.5
+ express@4.17.1
+ compression@1.7.4
+ mysql@2.17.1
added 66 packages from 50 contributors and audited 155 packages in 22.108s
found 0 vulnerabilities
```


Kemudian didalam folder api buat sebuah folder bernama config dan tambahkan sebuah file bernama config.js



Code config.js

```
const mysql = require('mysql');

const connectionMySQL = mysql.createConnection({
  multipleStatements: true,
  host: "localhost",
  port: 3399,
  user: "root",
  password: "",
  database: "reactjs"
});

module.exports = connectionMySQL;
```

Kemudian didalam folder api (root folder), buat sebuah file bernama server.js

```
const compression = require('compression');
const express = require('express');
const server = express();
const bodyParser = require('body-parser');
const cors = require("cors");

// Database MySQL - Definisikan route
const getRoute = require('./routes/route-get');

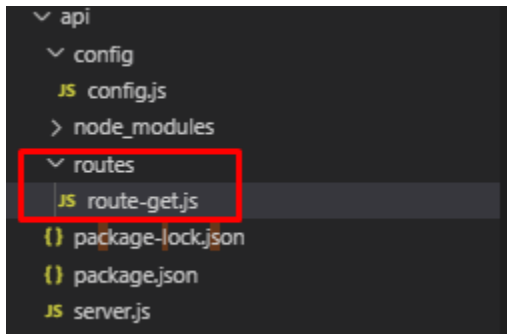
server.use(cors());
server.use(compression());
server.use(express.static("public"));
server.use(bodyParser.json({limit: "4mb"}));
server.use(bodyParser.urlencoded({extended: true}));

const port = process.env.PORT || 5001;
server.listen(port, () => console.log(`Listening on port ${port}`));

server.use(getRoute);
```

```
server.js - Untitled (Workspace) - Visual Studio Code
EXPLORER
  OPEN EDITORS
  UNTITLED (WORKSPACE)
    hris
    my-first-react
      api
      config
      JS config.js
      node_modules
      package-lock.json
      package.json
      JS server.js
  package.json
  JS config.js
  JS server.js x
api > JS server.js > ...
1  const compression = require('compression');
2  const express = require('express');
3  const server = express();
4  const bodyParser = require('body-parser');
5  const cors = require("cors");
6
7  // Database MySQL - Definisikan route
8  const getRoute = require('./routes/route-get');
9
10 server.use(cors());
11 server.use(compression());
12 server.use(express.static("public"));
13 server.use(bodyParser.json({limit: "4mb"}));
14 server.use(bodyParser.urlencoded({extended: true}));
15
16 const port = process.env.PORT || 5001;
17 server.listen(port, () => console.log(`Listening on port ${port}`));
18
19 server.use(getRoute);
```

Dikarenakan ada variable bernama `getRoute` difile `server.js` maka kita harus membuat sebuah folder bernama `routes`, dan didalamnya ada sebuah file bernama `route-get.js`



```
const express = require("express");
const router = express.Router();
const mySQL = require("../config/config");

exports.getHome = router.get("/", (req, res) => {
  console.log("You are inside a home page");
  res.send("You are inside a home page");
});

exports.getAllData = router.get("/api/all", (req, res) => {
  mySQL.query("SELECT * FROM karyawan", (err, result) => {
    if(err) console.log(err);
    res.send(JSON.stringify(result));
  })
});

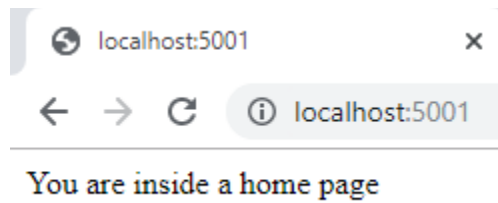
module.exports = router;
```

Sekarang jalankan `server.js` dengan `node` diterminal / cmd / visual studio code

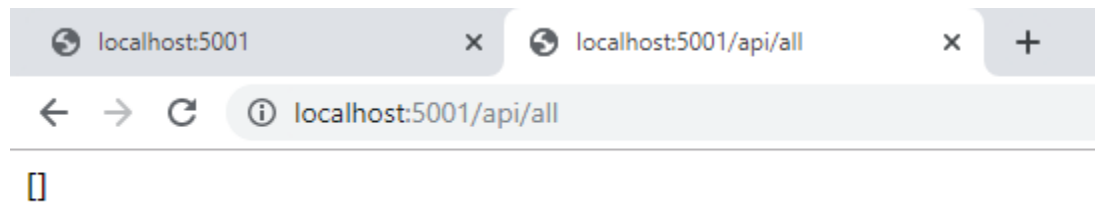
```
node server.js
```

```
PS E:\buku\Pemula\api> node .\server.js
Listening on port 5001
|
```

Jika kita akses <http://localhost:5001/>



Jika kita akses <http://localhost:5001/api/all>



Sampai sini kita sudah berhasil membuat web service dengan koneksi ke mysql database

3.3 Membuat api untuk memasukan data ke mysql

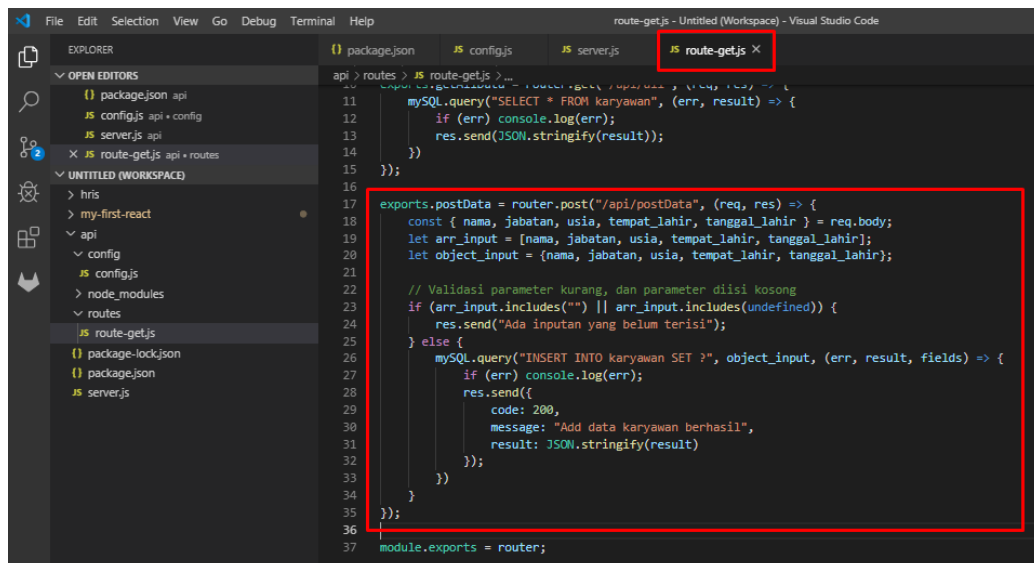
Pada bagian ini kita akan membuat api untuk memasukan data ke mysql

Tambahkan code ini diroute-get.js

```
exports.postData = router.post("/api/postData", (req, res) => {
  const { nama, jabatan, usia, tempat_lahir, tanggal_lahir } = req.body;
  let arr_input = [nama, jabatan, usia, tempat_lahir, tanggal_lahir];
  let object_input = {nama, jabatan, usia, tempat_lahir, tanggal_lahir};

  // Validasi parameter kurang, dan parameter diisi kosong
  if (arr_input.includes("") || arr_input.includes(undefined)) {
    res.send("Ada inputan yang belum terisi");
  } else {
    mysql.query("INSERT INTO karyawan SET ?", object_input, (err, result, fields) => {
      if (err) console.log(err);
      res.send({
        code: 200,
        message: "Add data karyawan berhasil",
        result: JSON.stringify(result)
      });
    });
  }
});
```

Sehingga route-get.js menjadi



```
File Edit Selection View Go Debug Terminal Help route-get.js - Untitled (Workspace) - Visual Studio Code
EXPLORER
  OPEN EDITORS
    package.json api
    JS config.js api + config
    JS server.js api
    JS route-get.js api + routes
  UNTITLED (WORKSPACE)
    hris
    my-first-react
    api
      config
      JS config.js
      node_modules
      routes
        JS route-get.js
    package-lock.json
    package.json
    server.js
  route-get.js
    11 mysql.query("SELECT * FROM karyawan", (err, result) => {
    12   if (err) console.log(err);
    13   res.send(JSON.stringify(result));
    14 });
    15 });
    16
    17 exports.postData = router.post("/api/postData", (req, res) => {
    18   const { nama, jabatan, usia, tempat_lahir, tanggal_lahir } = req.body;
    19   let arr_input = [nama, jabatan, usia, tempat_lahir, tanggal_lahir];
    20   let object_input = {nama, jabatan, usia, tempat_lahir, tanggal_lahir};
    21
    22   // Validasi parameter kurang, dan parameter diisi kosong
    23   if (arr_input.includes("") || arr_input.includes(undefined)) {
    24     res.send("Ada inputan yang belum terisi");
    25   } else {
    26     mysql.query("INSERT INTO karyawan SET ?", object_input, (err, result, fields) => {
    27       if (err) console.log(err);
    28       res.send({
    29         code: 200,
    30         message: "Add data karyawan berhasil",
    31         result: JSON.stringify(result)
    32       });
    33     });
    34   }
    35 });
    36
    37 module.exports = router;
```

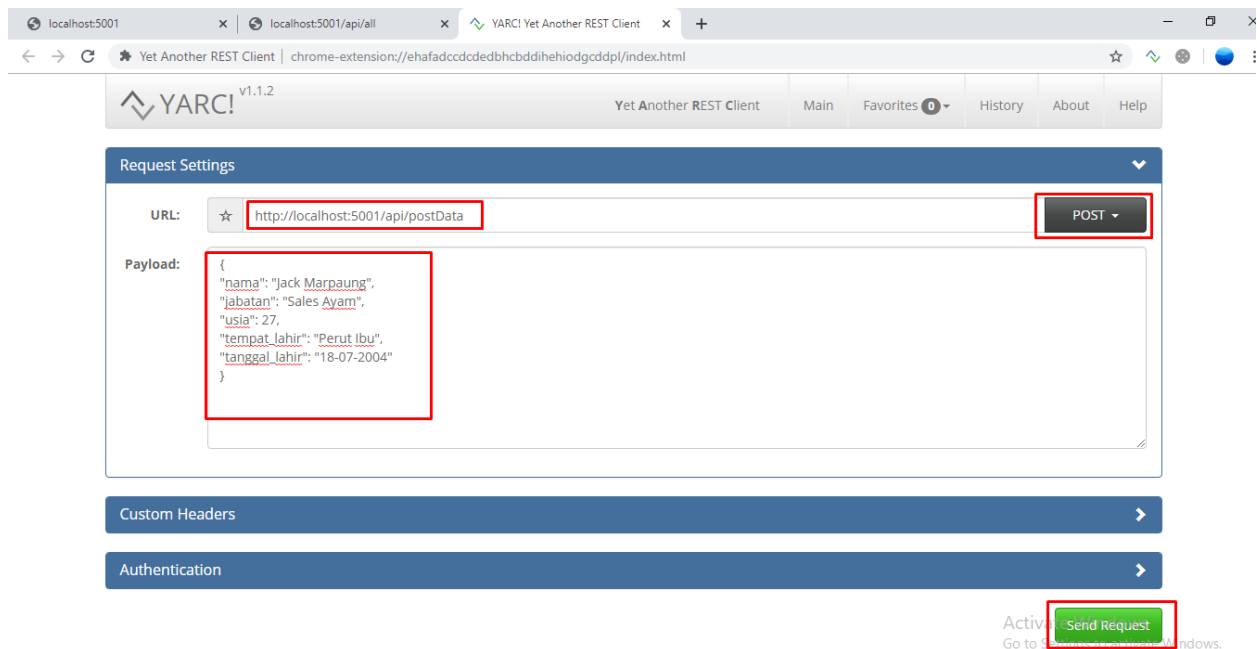
3.4 Uji api post ke mysql

Untuk menguji api post data kita ke database kita dapat menggunakan beberapa bantuan software seperti postman, juga dapat menggunakan extension yang ada di browser seperti http requester, yarc

Dalam uji api ini saya menggunakan yarc untuk test api, jika ingin menggunakan yang sama dapat menambahkan ke google chrome di link ini

<https://chrome.google.com/webstore/detail/yet-another-rest-client/ehafadccdcdbhcbddihehiodgcddpl>

Buat request post seperti ini



Url sesuaikan dengan api yang sudah kita buat

Payload sesuaikan dengan nama-nama field yang kita buat didatabase (sesuai table yang sudah kita buat yaitu table karyawan)

Maka akan mendapatkan request seperti ini

The screenshot shows a REST client interface with a blue header labeled "Authentication" and a green "Send Request" button. The "Response" section displays the following details:

- Request URL: <http://localhost:5001/api/postData>
- Request Method: POST
- Response Time: 0.176 seconds
- Response Status: 200 - OK

The response status "200" is highlighted with a red box. Below this, the "Response Body" tab is selected, showing a JSON response:

```
{
  "code": 200,
  "message": "Add data karyawan berhasil",
  "result": "{\"fieldCount\":0,\"affectedRows\":1,\"insertId\":17,\"serverStatus\":2,\"warningCount\":0,\"message\":\"\",\"protocol41\":true,\"changedRows\":0}"
}
```

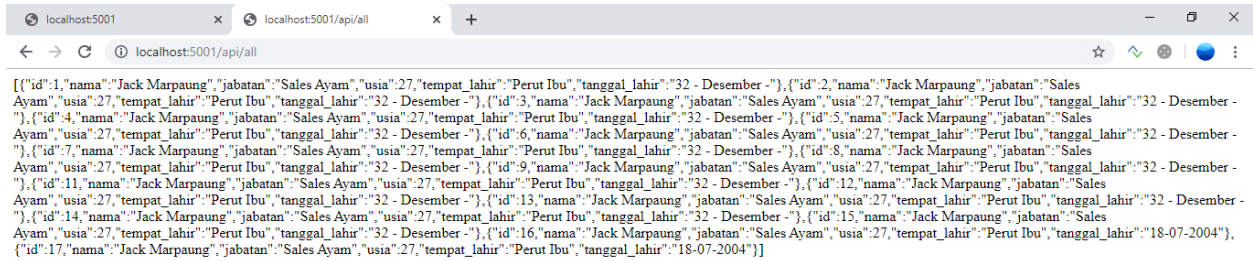
The JSON response is also highlighted with a red box. At the bottom, there are buttons for "Copy Request" and "Copy Response", and a watermark for "Activate Windows".

Jika kita lihat data didatabase maka akan bertambah datanya sesuai request post yang kita buat melalui YARC

The screenshot shows the HeidiSQL database interface. The left sidebar displays a tree view of databases, with "reactjs" and its sub-table "karyawan" highlighted with a red box. The main window shows the "karyawan" table data, which is also highlighted with a red box. The table has 17 rows, with the first 16 rows having a birth date of "32 - Desember -" and the last two rows having a birth date of "18-07-2004".

id	nama	jabatan	usia	tempat_lahir	tanggal_lahir
1	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
2	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
3	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
4	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
5	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
6	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
7	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
8	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
9	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
11	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
12	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
13	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
14	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
15	Jack Marpaung	Sales Ayam	27	Perut Ibu	32 - Desember -
16	Jack Marpaung	Sales Ayam	27	Perut Ibu	18-07-2004
17	Jack Marpaung	Sales Ayam	27	Perut Ibu	18-07-2004

Dan jika kita mencoba akses url <http://localhost:5001/api/all> pada browser, data yang tadinya array kosong [] menjadi array yang sudah ada isinya sesuai data dari table karyawan kita



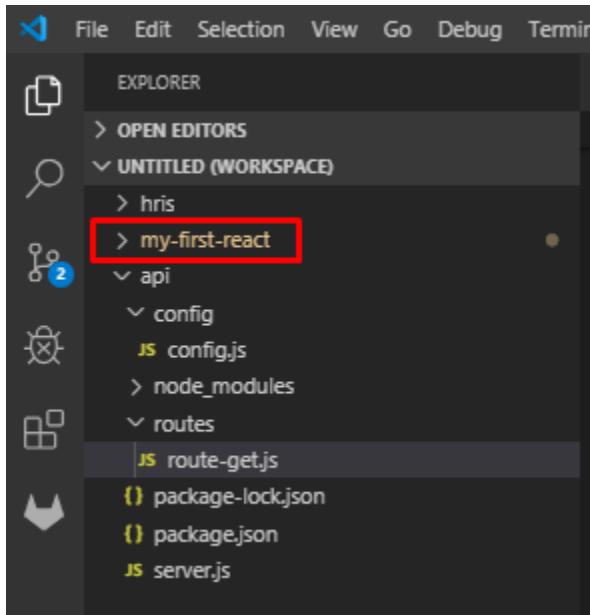
The screenshot shows a web browser window with the address bar displaying 'localhost:5001/api/all'. The main content area of the browser shows a JSON array of 17 employee records. Each record is an object with the following fields: 'id', 'nama', 'jabatan', 'usia', 'tempat_lahir', and 'tanggal_lahir'. The data is as follows:

```
[{"id":1,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":2,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":3,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":4,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":5,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":6,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":7,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":8,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":9,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":10,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":11,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":12,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":13,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":14,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"32 - Desember -"}, {"id":15,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"18-07-2004"}, {"id":16,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"18-07-2004"}, {"id":17,"nama":"Jack Marpaung","jabatan":"Sales Ayam","usia":27,"tempat_lahir":"Perut Ibu","tanggal_lahir":"18-07-2004"}]
```


3.5 Membuat form ketika disubmit simpan data ke table

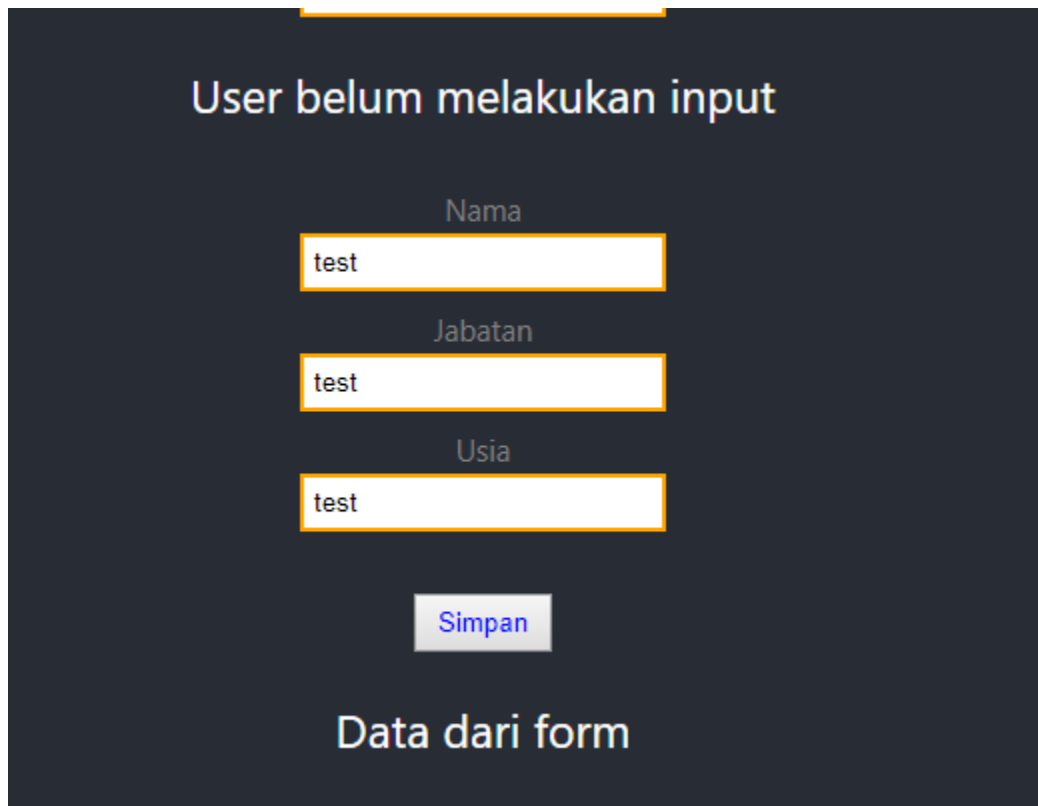
Jika kita sudah membuat api untuk mendapatkan data dan memasukan data ke mysql, sekarang kita akan membuat user interfacenya untuk user dapat melihat data dan memasukan data ke database.

Untuk membuat tampilan kita dapat menggunakan aplikasi pertama kita yaitu folder bernama “my-first-react”



Jalankan service my-first-react terlebih dahulu dengan Npm start

Kita akan mengubah code dari tampilan ini



```
File Edit Selection View Go Debug Terminal Help App.js - Untitled (Workspace) - Visual Studio Code

EXPLORER
my-first-react > src > JS App.js > App
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94

JS App.js
<ButtonDenganChildrenProps
  <div>Test</div>
</ButtonDenganChildrenProps>
<Paragraph nama="Santuy" jabatan="Gabut Manager">
  <div>Children dari Paragraph Component</div>
</Paragraph>

<ButtonProps nama="Tombol Ubah Data List" onClick={handleChange} />
<List data={dataList} />
<Input onChange={onChangeInput} onKeyDown={onEnterInput} />
<p>{valueInput}</p>

<Form>
  <label>Nama</label>
  <Input onChange={(e) => onChangeFormInput("nama", e)} />
  <label>Jabatan</label>
  <Input onChange={(e) => onChangeFormInput("jabatan", e)} />
  <label>Usia</label>
  <Input onChange={(e) => onChangeFormInput("usia", e)} />
  <br/>
  <ButtonProps nama="Simpan" color="blue" onClick={onClickBtnSimpan} />
  <p>Data dari form</p>
</Form>

<ButtonWithObjectProps name="Add" style={{color: "blue", height: "50px"}} />
<ButtonWithObjectProps name="Edit" style={{color: "red", height: "40px"}} />
<ButtonWithObjectProps name="Delete" style={{color: "green", height: "30px"}} />
```

Tapi sebelum kita membuat tampilan kita perlu package bernama axios untuk membantu kita hit api untuk mendapatkan data maupun menyimpan data

Jalankan perintah install axios

```
npm install axios
```

Tunggu hingga selesai

```
PS E:\buku\Pemula\my-first-react> npm install axios  
[.....] / rollbackFailedOptional: verb npm-session 254726ec711947a
```

Jika sudah terinstall

```
+ axios@0.19.0  
added 5 packages from 8 contri  
found 0 vulnerabilities
```

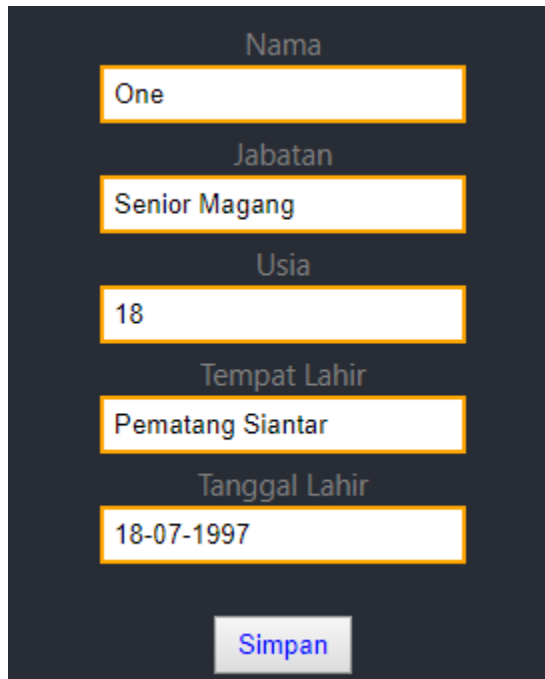
Dan jangan lupa untuk import axios sebelum menggunakannya

```
import axios from "axios";
```

Sekarang mari kita tambahkan dulu inputan diform kita menjadi seperti ini

```
<Form>  
  <label>Nama</label>  
  <Input onChange={e => onChangeFormInput("nama", e)} />  
  <label>Jabatan</label>  
  <Input onChange={e => onChangeFormInput("jabatan", e)} />  
  <label>Usia</label>  
  <Input onChange={e => onChangeFormInput("usia", e)} />  
  <label>Tempat Lahir</label>  
  <Input onChange={e => onChangeFormInput("tempat_lahir", e)} />  
  <label>Tanggal Lahir</label>  
  <Input onChange={e => onChangeFormInput("tanggal_lahir", e)} />  
  <br/>  
  <ButtonProps nama="Simpan" color="blue" onClick={onClickBtnSimpan} />  
  <p>Data dari form</p>  
</Form>
```

Form kita sudah menyiapkan semua inputan yang diperlukan oleh table kita



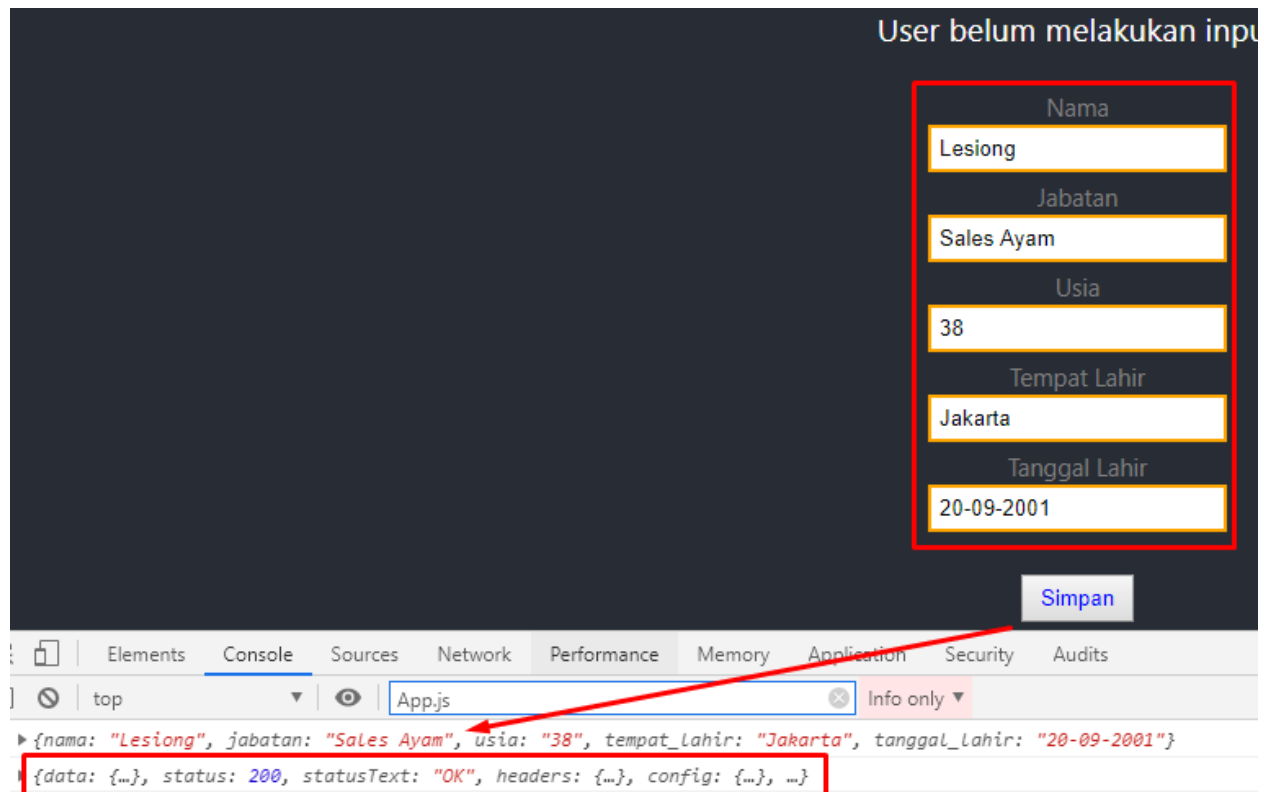
The image shows a form with a dark background. It contains six input fields with labels: 'Nama' (One), 'Jabatan' (Senior Magang), 'Usia' (18), 'Tempat Lahir' (Pematang Siantar), and 'Tanggal Lahir' (18-07-1997). A 'Simpan' button is at the bottom.

Karena pada saat tombol simpan diklik dihandle oleh fungsi `onClickBtnSimpan`, maka ubah fungsi `onClickBtnSimpan` menjadi seperti ini (juga ada tambahan sebuah fungsi baru yaitu `postData`)

```
async function postData(arrForm){
  let response = await axios.post("http://localhost:5001/api/postData", arrForm);
  console.log(response);
}

function onClickBtnSimpan(){
  console.log(arrForm);
  postData(arrForm);
}
```

Pada saat kita klik tombol simpan dan membuka inspect element



Axios akan melakukan request ke url yang kita berikan yaitu untuk memasukan data ke mysql

Sehingga jika kita lihat ditable karyawan akan ada data yang baru saja kita isi diform

18	One	Senior Magang	18	Pematang Siantar	18-07-1997
19	Lesiong	Sales Ayam	38	Jakarta	20-09-2001

4. Mini Project dengan React

About The Author



ONESINUS SAUT PARULIAN

Adalah seorang Mahasiswa Teknik Informatika di STIKOM Cipta Karya Informatika , Jakarta.

Dan juga seorang IT Consultant, pernah menjadi IT Support, Programmer PHP, Senior Staff Developer, React JS Programmer dan Freelancer.

My Hobbies are sharing what I can share ☺ gaming, also improving self ☺

Wanna get closer?

Email : onesinus231@gmail.com | phone / wa / line : 089626105445

Website : sautonesinus.blogspot.com | onespt.blogspot.com

Youtube: <https://www.youtube.com/channel/UCnta5kRZ7S4RIimfuYN-fWw>

LinkedIn: <https://www.linkedin.com/in/onesinus-spt-1a5653118/>

About The React JS Training Instructor



EKO ANDRI SUBARNANTO

“An overdrive thinker who enjoys improving things in life and at work”

Adalah seorang Senior Programmer di PT. Supranusa Sindata, juga seorang dosen di STMIK Swadharma.

Untuk mengenal lebih jauh dapat melihat link dibawah ini

LinkedIn: <https://www.linkedin.com/in/eko-andri-subarnanto/>

Website: <https://www.subarnanto.com/>