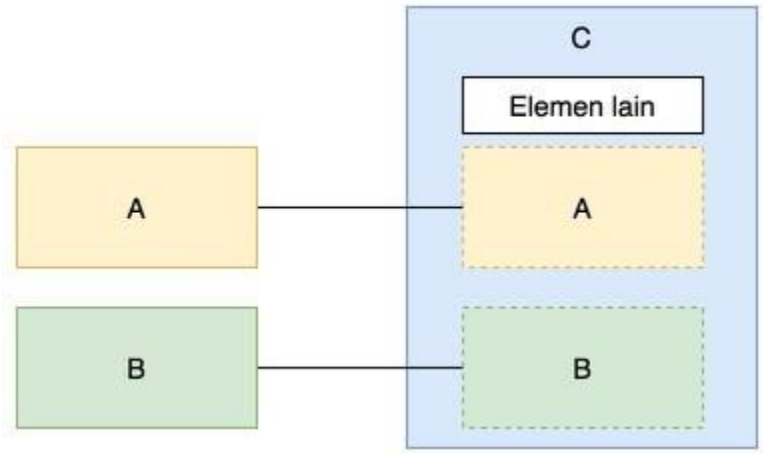
MODUL FRAMEWORK

REDUX DAN HOC

Higher-order Component (HOC) adalah teknik untuk membuat komponen ReactJS baru dari komponen yang udah ada. Jadi kita dapat membuat komponen baru yang punya fitur atau fungsi tambahan dengan cara komposisi yaitu satu atau lebih komponen digabung & dibungkus komponen lain.

Misalnya kita punya komponen A & B. HOC dapat membuat komponen baru C yang isinya A & B plus fitur-fitur & komponen/elemen DOM yang lain. Berikut ini gambaran komponennya



redux-form dapat mengelola formulir yang didukung oleh Redux. Higher-Order-Component (HOC) yang menggunakan react-redux memastikan bentuk HTML di React menggunakan Redux untuk menyimpan semua statusnya.

redux-form memiliki komponen-komponen sebagai berikut:

- formReducer (): Ini adalah fungsi yang memberitahu cara memperbarui Redux berdasarkan perubahan yang berasal dari aplikasi; perubahan tersebut dijelaskan oleh tindakan Redux. formReducer harus dipasang pada kondisi Redux di form.

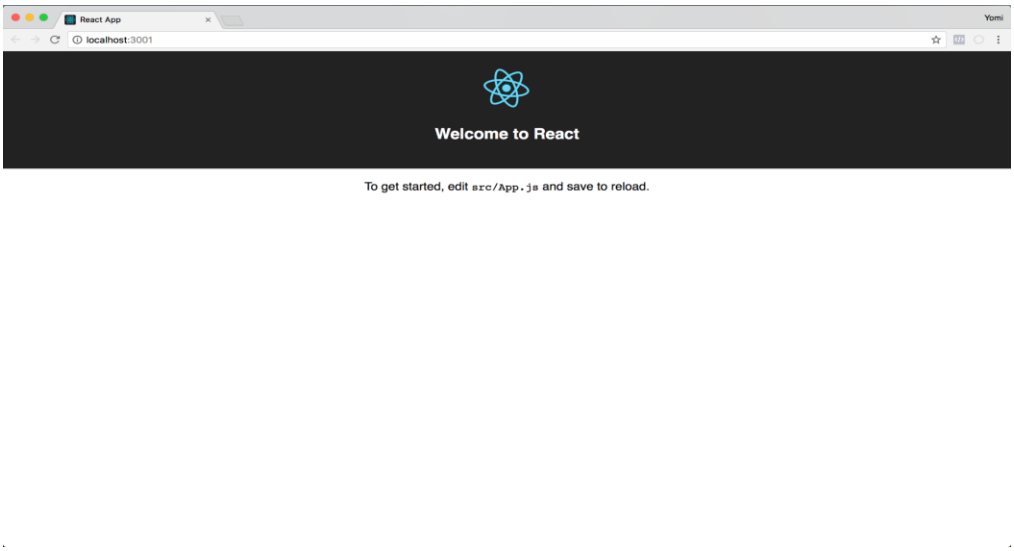
- reduxForm (): Fungsi reduxForm () adalah komponen untuk mengambil objek konfigurasi dan selalu mengembalikan fungsi baru.

- Komponen <Field />: Komponen yang berfungsi sebagai cara untuk menghubungkan elemen input dalam formulir ke logika redux-form.

Praktikum :

1. Buat dan install redux

   npx create-react-app contact-redux
   npm install -g create-react-app
   create-react-app contact-redux
   npm start



2. Install Form

   npm install --save redux react-redux redux-form

3. Ubah file index.html pada folder /public/index.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta name="theme-color" content="#000000">

  <link rel="manifest" href="%PUBLIC_URL%/manifest.json">
  <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bulma/0.6.2/css/bulma.min.css">

  <title>React App</title>
 </head>
 <body>
  <noscript>
   You need to enable JavaScript to run this app.
  </noscript>
  <div id="root"></div>

 </body>
</html>
```

4. Ubah file App.js pada folder /src/App.js

```
import React, { Component } from 'react';
import { reduxForm, Field } from 'redux-form';
import logo from './logo.svg';
import './App.css';

let SignInForm = props => {
 const { handleSubmit } = props;
 return <form onSubmit={handleSubmit} className="form">
  <div className="field">
   <div className="control">
    <Field name="firstName" component={renderField} type="text" label="First Name"/>
   </div>
  </div>

  <div className="field">
   <div className="control">
    <Field name="lastName" component={renderField} type="text" label="Last Name"/>
   </div>
  </div>

  <div className="field">
   <div className="control">
    <Field name="email" component={renderField} type="email" label="Email Address"/>
   </div>
  </div>

  <div className="field">
   <div className="control">
    <Field name="age" component={renderField} type="number" label="Age"/>
   </div>
  </div>

  <div className="field">
   <div className="control">
    <label className="label">Proficiency</label>
    <div className="select">
     <Field className="input" name="proficiency" component="select">
      <option />
      <option value="beginner">Beginner Dev</option>
      <option value="intermediate">Intermediate Dev</option>
      <option value="expert">Expert Dev</option>
     </Field>
    </div>
   </div>
  </div>

  <div className="field">
   <div className="control">
    <label className="label">Gender</label>
    <label className="radio">
     <Field name="gender" component="input" type="radio" value="male" />
     {' '}
     Male
    </label>
    <label className="radio">
     <Field name="gender" component="input" type="radio" value="female" />
     {' '}
     Female
    </label>
   </div>
  </div>
```

```
    <div className="field">
      <div className="control">
        <label className="checkbox">
          <Field name="saveDetails" id="saveDetails" component="input" type="checkbox"/>
          Save Details
        </label>
      </div>
    </div>

    <div className="field">
      <div className="control">
        <label className="label">Message</label>
        <Field className="textarea" name="message" component="textarea" />
      </div>
    </div>

    <div className="field">
      <div className="control">
        <button className="button is-link">Submit</button>
      </div>
    </div>

  </form>;
};

const validate = val => {
  const errors = {};
  if (!val.firstName) {
    console.log('First Name is required');
    errors.firstName = 'Required';
  }
  if (!val.lastName) {
    console.log('Last Name is required');
    errors.lastName = 'Required';
  }
  if (!val.email) {
    console.log('email is required');
    errors.email = 'Required';
  } else if (!/^.+@.+$/i.test(val.email)) {
    console.log('email is invalid');
    errors.email = 'Invalid email address';
  }
  if (!val.age) {
    errors.age = 'Required'
  } else if (isNaN(Number(val.age))) {
    errors.age = 'Must be a number'
  } else if (Number(val.age) < 18) {
    errors.age = 'Sorry, you must be at least 18 years old'
  }
  return errors;
};

const renderField = ({ input, label, type, meta: { touched, error, warning } }) => (
  <div>
    <div className="control">
      <label className="field">{label}</label>
      <input className="input" {...input} placeholder={label} type={type}/>
      {touched && ((error && <span>{error}</span>) || (warning && <span>{warning}</span>))}
    </div>
  </div>
)
```

```
SignInForm = reduxForm({
  form: 'signIn',
  validate,
})(SignInForm);

class App extends Component {

  handleSignIn = values => {
    console.log(values);
  };

  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h1 className="App-title">Welcome to React x redux-form</h1>
        </header>
        <div className="container">
          <p className="App-intro">
            Contact Form
          </p>
          <SignInForm onSubmit={this.handleSignIn} />
        </div>
      </div>
    );
  }
}

export default App;
```

5. Ubah file /src/index.js

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

import { createStore, combineReducers } from 'redux';
import { Provider } from 'react-redux';
import { reducer as formReducer } from 'redux-form';

import './index.css';
import App from './App';
import registerServiceWorker from './registerServiceWorker';

const rootReducer = combineReducers({
  form: formReducer,
});

const store = createStore(rootReducer);

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root')
);

registerServiceWorker();
```

6. Running hasilnya dan buatlah laporan

Keterangan :

1. Importing redux ke dalam react app

src/index.js

```
import { createStore, combineReducers } from 'redux';
import { Provider } from 'react-redux';
import { reducer as formReducer } from 'redux-form';
```

2. Mengkoneksikan Form ke redux form

src/index.js

```
SignInForm = reduxForm({
  form: 'signIn',
})(SignInForm);
```

3. Validasi redux form ke dalam Higher Order Component

src/index.js

```
SignInForm = reduxForm({
  form: 'signIn',
  validate,
})(SignInForm);
```

4. Membangun HOC dalam form redux

src/index.js

```
const renderField = ({ input, label, type, meta: { touched, error, warning } }) => (
  <div>
    <div className="control">
      <label className="field">{label}</label>
      <input className="input" {...input} placeholder={label} type={type}/>
      {touched && ((error && <span>{error}</span>) || (warning && <span>{warning}</span>))}
    </div>
  </div>
)
```