Contents

[CONVERTING APP 1](#_Toc78817741)

[ROUTING WITH React.lazy() 1](#_Toc78817742)

[USING React.memo() & more! 8](#_Toc78817743)

# CONVERTING APP

  useEffect(() => {

    props.onTryAutoSignup();

  },[])

# ROUTING WITH React.lazy()

import React,{useEffect,Suspense} from 'react';

import Layout from '../hoc/Layout/Layout';

import PizzaBuilder  from './PizzaBuilder/PizzaBuilder';

// import Checkout from './Checkout/Checkout';

// import Orders from './Orders/Orders';

// import Auth from './Auth/Auth';

import Logout from './Auth/logout/logout';

import { Route,Switch, withRouter,Redirect } from 'react-router-dom';

import {connect } from 'react-redux';

import \* as actions from '../store/actions/index'

const Checkout =  React.lazy( () => {

  return import('./Checkout/Checkout');

});

const Orders =  React.lazy( () => {

  return import('./Orders/Orders');

});

const Auth =  React.lazy( () => {

  return import('./Auth/Auth');

});

const App = (props) => {

  useEffect(() => {

    props.onTryAutoSignup();

  },[])

  let routes =(

    //JSX

    <Switch>

    <Route path="/auth" render={() => <Auth/> } />

    <Route path="/" exact component={PizzaBuilder} />

    <Redirect to="/" />

    </Switch>

  );

  if(props.isAuthenticated){

    routes=(

      <Switch>

       <Route path="/checkout" render={() => <Checkout/> } />

       <Route path="/orders" render={() => <Orders/> } />

       <Route path="/logout" component={Logout} />

       <Route path="/auth" render={() => <Auth/> } />

       <Route path="/" exact component={PizzaBuilder} />

      <Redirect to="/" />

      </Switch>

    );

  }

  return(

    <div className="App">

      <Layout>

      <Suspense fallback={<p>You can load anything like spinner loading</p>}>

       {routes}

    </Suspense>

      </Layout>

    </div>

  )

}

//APP.JS is root component which will loaded everytime. not matter which route we vist

//this makes great app to check authentication

const mapStateToProps = state =>{

  return{

    isAuthenticated: state.auth.token !=null

  };

};

const mapDispatchToProps =  dispatch => {

  return{

    onTryAutoSignup: () => dispatch(actions.authCheckState())

  };

};

export default withRouter(connect(mapStateToProps,mapDispatchToProps)(App));

Update:

let routes =(

    //JSX

    <Switch>

    <Route path="/auth" render={(props) => <Auth {...props}/> } />

    <Route path="/" exact component={PizzaBuilder} />

    <Redirect to="/" />

    </Switch>

  );

  if(props.isAuthenticated){

    routes=(

      <Switch>

       <Route path="/checkout" render={(props) => <Chec {...props}kout/> } />

       <Route path="/orders" render={(props) => <Orde {...props}rs/> } />

       <Route path="/logout" component={Logout} />

       <Route path="/auth" render={(props) => <Auth {...props}/> } />

       <Route path="/" exact component={PizzaBuilder} />

      <Redirect to="/" />

      </Switch>

    );

  }

AUth.js

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

import Input from '../../components/UI/Input-Form/Input';

import Button from '../../components/UI/Button/Button';

import './Auth.css';

import \* as actions from '../../store/actions/index';

import { connect } from 'react-redux';

import {Redirect} from 'react-router-dom';

const  Auth = props => {

    //state

    const [authForm, setAuthForm] = useState({

            email: {

                elementType: 'input',

                elementConfig: {

                    type: 'email',

                    placeholder: 'Mail Address'

                },

                value: '',

                validation: {

                    required: true,

                    isEmail: true

                },

                valid: false,

                touched: false

            },

            password: {

                elementType: 'input',

                elementConfig: {

                    type: 'password',

                    placeholder: 'Password'

                },

                value: '',

                validation: {

                    required: true,

                    minLength: 6

                },

                valid: false,

                touched: false

            }

        });

        //for sign up

       const [isSignup, setIsSignup] = useState(true);

    useEffect(() => {

        if(!props.buildingBurger && props.authRedirectPath !== '/' ){

            props.onSetAuthRedirectPath();

        }

    },[])

    const checkValidty = (value, rules) => {

        let isValid = true;

        //ALTERNATIVE ONE

        if(!rules) {

            return true;

        }

        if(rules.required){

            isValid = value.trim() !== '' && isValid;

        }

        if(rules.minLength){

            isValid = value.length >= rules.minLength && isValid;

        }

        if(rules.isEmail){

            const pattern = /[a-z0-9!#$%&'\*+=/=?]/;

            isValid = pattern.test(value) && isValid;

        }

        if(rules.maxLength){

            isValid = value.length <= rules.maxLength && isValid;

        }

        return isValid;

    }

    const submitHandler = (event) => {

        event.preventDefault();

        props.onAuth(authForm.email.value,authForm.password.value,isSignup)

    }

    const switchAuthModelHandler = () => {

        setIsSignup(!isSignup);

    }

    const formElementsArray = [];

    for(let key in authForm){

        formElementsArray.push({

            id: key,

            config: authForm[key]

        })

    }

    const form = formElementsArray.map(formElement => (

        <Input

            key={formElement.id}

            elementType={formElement.config.elementType}                elementConfig ={formElement.config.elementConfig}

            value={formElement.config.value}

            invalid={!formElement.config.valid}

            shouldValidate={formElement.config.validation}

            touched={formElement.config.touched}

            changed={(event) => inputChangedHandler(event,formElement.id)}

        />

    ));

    const inputChangedHandler =  (event, controlName) => {

        const updatedControls = {

            ...authForm,

               [controlName]:{

                   ...authForm[controlName],

                   value: event.target.value,

                   valid: checkValidty(event.target.value, authForm[controlName].validation),

                   touched: true

           }

        };

        setAuthForm(updatedControls);

    }

        //redirect to

    let authRedirect = null;

    if(props.isAuthenticated){

        authRedirect =<Redirect to={props.authRedirectPath} />

   }

     return(

        <div className='Auth'>

            {authRedirect}

            <form onSubmit={submitHandler}>

            {form}

           <Button btnType="Success">Submit</Button>

            </form>

            <Button

            clicked={switchAuthModelHandler}

            btnType="Danger"

            >

           SWITCH TO {isSignup ?

            'LOGIN' : 'SIGNUP'}</Button>

       </div>

     );

}

const mapStateToProps = state =>{

    return {

        isAuthenticated: state.auth.token !== null,

        buildingBurger: state.burger.building,

        authRedirectPath: state.auth.authRedirectPath

    };

};

const mapDispatchToProps = dispatch => {

    return {

        onAuth: (email,password, isSignup) => dispatch(actions.auth(email,password,isSignup)),

        onSetAuthRedirectPath: () => dispatch(actions.setAuthRedirectPath('/'))

    };

}

export default connect(mapStateToProps,mapDispatchToProps)(Auth);

Logout.js

import React, {useEffect} from 'react';

import { Redirect } from 'react-router-dom';

import {connect } from 'react-redux';

import \* as actions from '../../../store/actions/index';

const Logout = (props) => {

    useEffect(() => {

        props.onLogout();

    },[]);

    return <Redirect to="/" />;

}

const mapDispatchToProps = dispatch =>{

    return {

        onLogout: () => dispatch(actions.logout())

    };

}

export default connect(null,mapDispatchToProps)(Logout);

# USING React.memo() & more!

Modal.js

import React from 'react';

import './Modal.css';

import Aux from '../../../hoc/Auxilliary/Auxilliary';

import Backdrop from '../Backdrop/Backdrop';

const Modal = (props) => (

    <Aux>

        <Backdrop show={props.show} clicked={props.modalClosed} />

        <div className='Modal'

        style ={{

        transform: props.show ? 'translateY(0)' : 'translateY(-100vh)',

        opacity: props.show? '1' : '0'}}>

        {props.children}

        {/\* children can be anything even your components  \*/}

    </div>

    </Aux>

);

export default React.memo(

    Modal,

    (prevProps, nextProps) =>

        nextProps.show == prevProps.show &&

        nextProps.children == prevProps.children

);