import './App.css';

import { useRef } from 'react';

function UserForm() {

    const name = useRef();

    const age = useRef();

    const handleSubmit = (e)=>{

        e.preventDefault();

        console.log("hello Submit")

        console.log("Name: ", name.current.value)

        console.log("Age: ", age.current.value)

    }

  return (

    <div>

        <div>

            User Form

        </div>

        <form onSubmit={handleSubmit}>

        <div>

            <label htmlFor="">Name</label>

            <input id="name" ref={name} type="text"/>

        </div>

        <div>

            <label htmlFor="">Age</label>

            <input id="age" ref = {age} type="number"/>

        </div>

        <div>

            <button type="submit">Submit</button>

        </div>

        </form>

    </div>

  );

}

export default UserForm;

We have created a form. We take two inputs name and age. useRef is used to update inputs of forms. We use ref from useRef. It takes updated value from useRef and also update useRef with that input value. On submit, show hello submit, name and age on console. To handle errors like if name is not there or name has more than 20 characters, we have to do them separately. And this is only for one field, if larger form, more errors and more hustle. So to avoid this, we use formic

FORMIK:

INSTALLATION:

npm install formik –save

OR

yarn add formik

Now to implement form, I created another file name UserFormWithFormik.js

function UserFormWithFormik() {

    const formik = useFormik({

        initialValues: {

            name: "",

            age: 0

        },

        onSubmit: (values) => {

            console.log(values)

        },

        validate: (values) => {

            const error = {};

            if (!values.name){

                error.name = "Name field is Required"

            }

            else if (values.name.length > 20) {

                error.name = "Name field must be less than 20 characters"

            }

            return error;

        }

    })

As in our form, till now we have only two fields, name and age. So in Formik, we pass object with first property is initialValues have name an empty string and age is 0.

Now we pass operation that what happened when we click on Submit. So onSubmit, we take all values and print them on console.

Now to validate our forms, we define empty error object. First if there is no name in name field, it raise an error.

Similarly for name greater than 20 characters.

        <div>

            User Form

        </div>

        <form onSubmit={formik.handleSubmit}>

        <div>

            <label htmlFor="">Name</label>

            <input id="name"  type="text"

                onChange={formik.handleChange}

                value={formik.values.name}

            />

            {

                formik.errors.name?

                <div style={{color: "red"}}>{formik.errors.name}</div>:

                null

            }

        </div>

Now we use OnSubmit from formic. Everything same as previous or normal forms, but we define two new properties onChange and value in every field. onChange will handle any change in that name field. And it will save or update or use value form formic values.

And in order to handle the error if their, we use true or false. If error.name exist, then show error other wise null.

In above we seen, it requires lots of if else for validation. To get rid from that, we use yup library.

YUP:

INSTALLATION:

npm install yup –save

OR

yarn add yup

Now we update the previous validation part. Create new file UserFormwithFormikValidation.js

       onSubmit: (values) => {

            console.log(values)

        },

        validationSchema: Yup.object(

            {

                name: Yup.string()

                        .required("Name field is required")

                        .max(20, "Name field should be less than 20 characters"),

                age: Yup.number()

                        .max(60, "Age should be less than 60")

                        .min(10, "age should be greater than 10"),

            }

        )

    })

  return

We use validation scheme from yup to validate our form. We created an object, as we have to validate the two fields only, so pass two fields in it. Name which should be string and it must be the required field and it should not more than 20.

Similarly for age, it should be number with max of 60 and min of 10, by this, we get rid from the if else.

We have also update the onChange part as it same for all input fields just change name of the field.

           <label htmlFor="">Name</label>

            <input id="name"  type="text"

                onChange={formik.handleChange}

                value={formik.values.name}

            />

            {

                formik.errors.name?

                <div style={{color: "red"}}>{formik.errors.name}</div>:

                null

            }

        </div>

        <div>

            <label htmlFor="">Age</label>

            <input id="age"  type="number" {...formik.getFieldProps("age")}

            />

            {

                formik.errors.age?

                <div style={{color: "red"}}>{formik.errors.age}</div>:

                null

Input field of age is minimized by just getting all the formic from previous field, and update it according to the age property. So we can do this with many new fields.

Now to minimized our code further, we will use formic context.

INITIAL PROGRESS:

function UserFormWithFormikValidationContext() {

  return (

    <div>

        <div>

            User Form

        </div>

        <Formik initialValues={{

            name: "",

            age: 0

        }}

        onSubmit={(values) => {

            console.log(values)

        }}

        validationSchema={

            Yup.object(

                {

                    name: Yup.string()

                            .required("Name field is required")

                            .max(20, "Name field should be less than 20 characters"),

                    age: Yup.number()

                            .max(60, "Age should be less than 60")

                            .min(10, "age should be greater than 10"),

                }

            )

        }

        >

        {

            (formik)=>(

                <form onSubmit={formik.handleSubmit}>

        <div>

            <label htmlFor="">Name</label>

            <input id="name"  type="text"

                onChange={formik.handleChange}

                value={formik.values.name}

            />

            {

                formik.errors.name?

                <div style={{color: "red"}}>{formik.errors.name}</div>:

                null

            }

        </div>

        <div>

            <label htmlFor="">Age</label>

            <input id="age"  type="number" {...formik.getFieldProps("age")}

            />

            {

                formik.errors.age?

                <div style={{color: "red"}}>{formik.errors.age}</div>:

                null

            }

        </div>

        <div>

            <button type="submit">Submit</button>

        </div>

        </form>

)}

        </Formik>

    </div>

  );

}

export default UserFormWithFormikValidationContext;

Now we provide, all properties like initial values, onsubmit, validation schema under Fomik tag. This behave as provider, in order to consume that, we created object inside same Formik tag, call formik their, and use our previous form that take all values from formik. Now can also minimized it further like removing input fields etc. So lets see this also.

  (formik)=>(

                <Form onSubmit={formik.handleSubmit}>

                <div>

                    <label htmlFor="">Name</label>

                    <Field type="text" name="name" id="name" />

                    <ErrorMessage name="name" render={(msg)=>(

                        <span style={{color: "red"}}>{msg}</span>

                    )}/>

                </div>

                <div>

                    <label htmlFor="age">Age</label>

                    <Field type="number" name="age" id="age" />

                    <ErrorMessage name="age"/>

                 </div>

                <div>

                    <button type="submit">Submit</button>

                </div>

            </Form>

    )}

        </Formik>

Now it can be seen that code is very much reduced as not input and onchange stuff is their, For input, we use Formik Field and for error, we use Formik ErrorMessage, and we can handle error msg, its color or position by passing render in ErrorMessage tag.