Form validator using re-usable components

Exp. No.: 7

Name: Akash M

Roll No.: 19CSR006

Code:

App.js:

import React, { Component } from "react";

import TextField from "./TextField";

import "./Main.css";

export default class Main extends Component {

constructor() {

super();

this.state = {

isValid: false,

touched: false,

msg: "",

};

this.nameRef = React.createRef();

}

componentDidMount() {

this.nameRef.current.focus();

}

nameValidator(name) {

// name length must be atleast 3 and should not contain any characters other then space and dot

if (name.length < 3)

return { success: false, msg: "Name must be atleast 3 characters long!" };

const pattern = /^[a-zA-Z. ]+$/;

if (!pattern.test(name)) {

return {

success: false,

msg: "Name can have only alphabets, . and space",

};

}

return { success: true, msg: "Validation Passed!" };

}

ageValidator(age) {

if (age < 0) return { success: false, msg: "Age must always be positive" };

if (age < 18)

return { success: false, msg: "You must be older then 18 to validate" };

return { success: true, msg: "Validation Passed!" };

}

formValidator(errFlag) {

let msg = "";

if (errFlag === true) msg = "All validations passed!";

else msg = "Please check all the fields";

this.setState({

isValid: errFlag,

msg: msg,

});

}

submitHandler(event) {

event.preventDefault();

if (!this.state.isValid) {

this.setState({

msg: "Fields are not validated correctly, please check again!",

});

} else {

this.setState({

isDisabled: true,

touched: true,

msg: "All validations passed!",

});

}

}

render() {

return (

<div>

<h1>Validation</h1>

<form onSubmit={this.submitHandler.bind(this)}>

{this.state.touched && (

<p

className={`form\_\_text\_\_container ${

this.state.isValid ? "success" : "err"

}`}

>

{this.state.msg}

</p>

)}

<TextField

type="text"

init=""

name="name"

validator={this.nameValidator.bind(this)}

formValidator={this.formValidator.bind(this)}

isDisabled={false}

refVar={this.nameRef}

/>

<TextField

type="number"

init="0"

name="age"

validator={this.ageValidator.bind(this)}

formValidator={this.formValidator.bind(this)}

isDisabled={false}

/>

<button

disabled={!this.state.isValid}

style={{ color: "black" }}

type="submit"

>

Submit

</button>

</form>

</div>

);

}

}

export default Toggle;

App.css:

.form\_\_text\_\_container {

padding: 0.5rem;

border-radius: 4px;

}

.success {

border: 1px solid rgb(8, 80, 8);

background-color: rgb(47, 140, 47);

}

.err {

border: 1px solid rgb(145, 6, 6);

background-color: rgb(149, 60, 60);

}

form {

border: 2px solid lightgray;

border-radius: 4px;

padding: 1rem;

width: fit-content;

margin: 2rem auto 0 auto;

}

button {

padding: 0.5rem 1rem;

border: 1px solid transparent;

border-radius: 4px;

font-weight: bold;

}

TextField.js:

import { useEffect, useState } from "react";

import "./TextField.css";

const TextField = (props) => {

const [field, setField] = useState(props.init);

const [err, setErr] = useState(false);

const [msg, setMsg] = useState("");

const [touched, setTouched] = useState(false);

useEffect(() => {

if (touched) {

const result = props.validator(field);

setErr(result.success);

setMsg(result.msg);

props.formValidator(result.success);

}

}, [field, touched]);

const changeHandler = (event) => {

event.preventDefault();

if (touched === false) setTouched(true);

setField(event.target.value);

};

return (

<div className="textField\_\_container">

<label>{props.name}:</label>

<input

value={field}

onChange={changeHandler}

type={props.type}

name={props.name}

disabled={props.isDisabled}

ref={props.refVar}

/>

{msg.length > 0 && (

<p className={`msg\_\_container ${err ? "success" : "err"}`}>{msg}</p>

)}

</div>

);

};

export default TextField;

TextField.css:

label {

text-transform: capitalize;

margin-right: 0.3rem;

}

.textField\_\_container {

/\* border: 1px solid lightgray; \*/

width: fit-content;

margin: 1rem auto;

}

.msg\_\_container {

padding: 0.5rem;

border-radius: 4px;

}

.success {

border: 1px solid rgb(8, 80, 8);

background-color: rgb(47, 140, 47);

}

.err {

border: 1px solid rgb(145, 6, 6);

background-color: rgb(149, 60, 60);

}

input {

color: black;

}

OUTPUT:

