App.js

import React, { Component } from "react";

import AppBar from "@mui/material/AppBar";

import Box from "@mui/material/Box";

import Toolbar from "@mui/material/Toolbar";

import Typography from "@mui/material/Typography";

import Button from "@mui/material/Button";

import IconButton from "@mui/material/IconButton";

import MenuIcon from "@mui/icons-material/Menu";

import FormInput from "./FormInput";

import CardTem from "./CardTem";

class App extends Component {

state = {

personData: [],

};

//create a function inside parent component

addData = (user) => {

console.log(user); //add user inside personData(state)

this.setState({

personData: [...this.state.personData, user],

});

};

render() {

return (

<>

<Box sx={{ flexGrow: 1 }}>

<AppBar position="static">

<Toolbar>

<IconButton

size="large"

edge="start"

color="inherit"

aria-label="menu"

sx={{ mr: 2 }}

>

<MenuIcon />

</IconButton>

<Typography variant="h6" component="div" sx={{ flexGrow: 1 }}>

Navbar

</Typography>

<Button color="inherit">Home</Button>

<Button color="inherit">About Us</Button>

<Button color="inherit">Contact</Button>

</Toolbar>

</AppBar>

</Box>

<FormInput addData={this.addData} />

<CardTem personData={this.state.personData} />

</>

);

}

}

export default App;

FormInput.js

import React, { Component } from "react";

import Grid from "@mui/material/Grid";

class FormInput extends Component {

state = {

name: "",

email: "",

birthday: "",

contact: "",

info: "",

error: {

nameError: "",

emailError: "",

birthdayError: "",

contactError: "",

infoError: "",

},

formIsValid: false,

};

//event object: created automatically

handleChange = (e) => {

console.log(e);

console.log(e.target.value);

console.log(e.target.id);

// Validation

if (e.target.id === "name") {

this.validateName(e.target.value);

}

if (e.target.id === "email") {

this.validateEmail(e.target.value);

}

if (e.target.id === "birthday") {

this.validateBirthday(e.target.value);

}

if (e.target.id === "contact") {

this.validateContact(e.target.value);

}

if (e.target.id === "info") {

this.validateInfo(e.target.value);

}

// this.setState({

// [e.target.id]: e.target.value

// });

};

validateName = (name) => {

let nameError = this.state.error.nameError;

let formIsValid = this.state.formIsValid;

if (name.trim() === "") {

nameError = "Please enter name";

formIsValid = false;

} else if (name.trim().length <= 3) {

nameError = "Please enter a name having more 3 letters";

formIsValid = false;

} else if (!/^[A-Za-z\s]\*$/.test(name)) {

nameError = "Please key only alphabets";

formIsValid = false;

}

// else if(!patten.test(name)){}

else {

nameError = "";

formIsValid = true;

}

this.setState({

name: name,

error: { ...this.state.error, nameError },

formIsValid,

});

return formIsValid;

};

validateEmail = (email) => {

let formIsValid = this.state.formIsValid;

let emailError = this.state.error.emailError;

if (email.trim() === "") {

formIsValid = false;

emailError = "This field can't be empty";

} else if (!/\S+@\S+\.\S+/.test(email)) {

formIsValid = false;

emailError = "Please enter email address again.";

}

// else if(!/^\d+$/.test(contact)) {

// }

else {

formIsValid = true;

emailError = "";

}

this.setState({

email,

formIsValid,

error: { ...this.state.error, emailError },

});

return formIsValid;

};

validateBirthday = (birthday) => {

let birthdayError = this.state.error.birthdayError;

let formIsValid = this.state.formIsValid;

if (birthday.trim() === "") {

formIsValid = false;

birthdayError = "Please enter birthday";

} else {

formIsValid = true;

birthdayError = "";

}

this.setState({

birthday,

formIsValid,

error: { ...this.state.error, birthdayError },

});

return formIsValid;

};

validateContact = (contact) => {

let formIsValid = this.state.formIsValid;

let contactError = this.state.error.contactError;

if (contact.trim() === "") {

formIsValid = false;

contactError = "This field can't be empty";

} else if (contact.trim().length != 10) {

formIsValid = false;

contactError = "Contact number should be of 10 characters";

} else if (!/^[0-9]{10}$/.test(contact)) {

formIsValid = false;

contactError = "Contact number should be only numbers";

} else {

formIsValid = true;

contactError = "";

}

this.setState({

contact,

formIsValid,

error: { ...this.state.error, contactError },

});

return formIsValid;

};

validateInfo = (info) => {

let formIsValid = this.state.formIsValid;

let infoError = this.state.error.infoError;

if (info.trim() === "") {

formIsValid = false;

contactError = "This field can't be empty";

} else {

formIsValid = true;

infoError = "";

}

this.setState({

info,

formIsValid,

error: { ...this.state.error, infoError },

});

return formIsValid;

};

handleSubmit = (e) => {

e.preventDefault(); //restrict and prevent the default behavior of any event

// let nameBool = this.validateName();

if (

this.validateName(this.state.name) &&

this.validateEmail(this.state.email) &&

this.validateBirthday(this.state.birthday) &&

this.validateContact(this.state.contact) &&

this.validateInfo(this.state.info)

) {

alert("form is submited");

this.props.addData(this.state);

this.setState({

name: "",

email: "",

birthday: "",

contact: "",

info: "",

});

} else {

alert("Please check all the fields");

}

};

render() {

const labelStyle = {

textAlign: "center",

paddingRight: "20px",

paddingBottom: "20px",

};

const inputStyle = {

padding: "15px",

paddingLeft: "20px",

paddingTop: "20px",

};

const infoStyle = {

padding: "30px",

paddingLeft: "20px",

};

return (

<Grid

item

container

direction="column"

sx={{ marginTop: 2, paddingTop: 2 }}

justifyContent="center"

alignItems="center"

>

<form onSubmit={this.handleSubmit}>

<label style={labelStyle}>Name</label>

<br></br>

<input

style={inputStyle}

type="text"

placeholder="Please enter name"

id="name"

onChange={this.handleChange}

value={this.state.name}

/>

<p style={{ color: "red" }}>{this.state.error.nameError}</p>

<label style={labelStyle}>Email</label>

<br></br>

<input

style={inputStyle}

type="text"

onChange={this.handleChange}

placeholder="Please enter email address"

id="email"

value={this.state.email}

/>

<p style={{ color: "red" }}>{this.state.error.emailError}</p>

<label style={labelStyle}>Date of Birth</label>

<br></br>

<input

style={inputStyle}

type="date"

onChange={this.handleChange}

id="birthday"

value={this.state.birthday}

/>

<p style={{ color: "red" }}>{this.state.error.birthdayError}</p>

<label style={labelStyle}>Contact</label>

<br></br>

<input

style={inputStyle}

type="number"

onChange={this.handleChange}

placeholder="Please enter contact number"

id="contact"

value={this.state.contact}

/>

<p style={{ color: "red" }}>{this.state.error.contactError}</p>

<label style={labelStyle}>Tell me about yourself</label>

<br></br>

<input

style={infoStyle}

type="text"

onChange={this.handleChange}

placeholder="Write here"

id="info"

value={this.state.info}

/>

<p style={{ color: "red" }}>{this.state.error.infoError}</p>

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

</form>

</Grid>

);

}

}

export default FormInput;

CardTem.js

import React from "react";

// 1 rem = 16px

const CardTem = (props) => {

const { personData } = props;

const personList = personData.length ? (

personData.map((info) => {

return (

<div

className="card"

style={{ width: "200px", padding: "20px" }}

key={info.contact}

>

<div className="card-body">

<h5 className="card-title">{info.name}</h5>

<p className="card-text-email">{info.email}</p>

<p className="card-text-birthday">{info.birthday}</p>

<p className="card-text-birthday">{info.contact}</p>

<p className="card-text-info">{info.info}</p>

</div>

</div>

);

})

) : (

<></>

);

return <>{personList}</>;

};

export default CardTem;