Form

<input type="text" />

<input type="number" />

<input type="range" />

<input type="email" />

<input type="password" />

<input type="tel" />

<input type="checkbox" />

<input type="radio" />

<input type="color" />

<input type="url" />

<input type="image" />

<input type="file" />

<input type="hidden" />

<input type="date" />

<input type="datetime-local" />

<input type="month" />

<input type="week" />

<input type="time" />

<input type="reset" />

<input type="search" />

<input type="submit" />

<input type="button" />

<textarea>Please write your comment ...</textarea>

<select name="country">

<option value="">Select your country</option>

<option value="finland">Finland</option>

<option value="sweden">Sweden</option>

<option value="denmark">Denmark</option>

<option value="norway">Norway</option>

<option value="iceland">Iceland</option>

</select>

React form example

import { useState } from "react";

import { useForm } from "react-hook-form";

import Header from "./Header";

export function App() {

const { register, handleSubmit } = useForm();

const [data, setData] = useState("");

return (

<form onSubmit={handleSubmit((data) => setData(JSON.stringify(data)))}>

<Header />

<input {...register("firstName")} placeholder="First name" />

<select {...register("category", { required: true })}>

<option value="">Select...</option>

<option value="A">Option A</option>

<option value="B">Option B</option>

</select>

<textarea {...register("aboutYou")} placeholder="About you" />

<p>{data}</p>

<input type="submit" />

</form>

);

}

**Handling Forms**

Handling forms is about how you handle the data when it changes value or gets submitted.

In HTML, form data is usually handled by the DOM.

In React, form data is usually handled by the components.

When the data is handled by the components, all the data is stored in the component state.

You can control changes by adding event handlers in the onChange attribute.

We can use the useState Hook to keep track of each inputs value and provide a "single source of truth" for the entire application.

import { useState } from 'react';

import ReactDOM from 'react-dom/client';

function MyForm() {

const [name, setName] = useState("");

return (

<form>

<label>Enter your name:

<input

type="text"

value={name}

onChange={(e) => setName(e.target.value)}

/>

</label>

</form>

)

}

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<MyForm />);

**Getting data from an input field**

The input element has many attributes such as value, name, id, placeholder, type and event handler. In addition, we can link a label and an input field using an id of input field and htmlFor of the label.If label and input are linked it will focus the input when we click on label. Look at the example give below.

import React, { Component } from 'react'

import ReactDOM from 'react-dom'

class App extends Component {

// declaring state

// initial state

state = {

firstName: '',

}

handleChange = (e) => {

const value = e.target.value

this.setState({ firstName: value })

}

render() {

/\*

accessing the state value and

this value will injected to the input in the value attribute

\*/

const firstName = this.state.firstName

return (

<div className='App'>

<label htmlFor='firstName'>First Name: </label>

<input

type='text'

id='firstName'

name='firstName'

placeholder='First Name'

value={firstName}

onChange={this.handleChange}

/>

<h1>{this.state.firstName}</h1>

</div>

)

}

}

const rootElement = document.getElementById('root')

ReactDOM.render(<App />, rootElement)

Getting multiple input data from form

In this section we will develop a small form which collect user information. Our user is a student. We use a parent form element and certain number of input elements to collect user information. In addition to that we will have event listener for the form (onSubmit) and for the inputs (onChange). See the following example try to see the commonts too. You can also check the live demo.

import React, { Component } from 'react'

import ReactDOM from 'react-dom'

class App extends Component {

// declaring initial state

state = {

firstName: '',

lastName: '',

country: '',

title: '',

}

handleChange = (e) => {

/\*

we can get the name and value like this: e.target.name, e.target.value

but we can also destructure name and value from e.target

const name = e.target.name

const value = e.target.value

\*/

const { name, value } = e.target

// [variablename] to use a variable name as a key in an object

// name refers to the name attribute of the input elements

this.setState({ [name]: value })

}

 handleSubmit = (e) => {

/\*

e.preventDefault()

stops the default behavior of form element

specifically refreshing of page

\*/

e.preventDefault()

/\*

the is the place where we connect backend api

to send the data to the database

\*/

console.log(this.state)

}

render() {

// accessing the state value by destrutcturing the state

const { firstName, lastName, title, country } = this.state

return (

<div className='App'>

<h3>Add Student</h3>

<form onSubmit={this.handleSubmit}>

<div>

<input

type='text'

name='firstName'

placeholder='First Name'

value={firstName}

onChange={this.handleChange}

/>

</div>

<div>

<input

type='text'

name='lastName'

placeholder='Last Name'

value={lastName}

onChange={this.handleChange}

/>

</div>

<div>

<input

type='text'

name='country'

placeholder='Country'

value={country}

onChange={this.handleChange}

/>

</div>

<div>

<input

type='text'

name='title'

placeholder='Title'

value={title}

onChange={this.handleChange}

/>

</div>

<button class='btn btn-success'>Submit</button>

</form>

</div>

)

}

}

const rootElement = document.getElementById('root')

ReactDOM.render(<App />, rootElement)

The above form handles only text types but do have different input field types. Let's do another form which handle all the different input field types.

Get data from different input field types

// index.js

import React, { Component } from 'react'

import ReactDOM from 'react-dom'

const options = [

{

value: '',

label: '-- Select Country--',

},

{

value: 'Finland',

label: 'Finland',

},

{

value: 'Sweden',

label: 'Sweden',

},

{

value: 'Norway',

label: 'Norway',

},

{

value: 'Denmark',

label: 'Denmark',

},

]

// mapping the options to list(array) of JSX options

const selectOptions = options.map(({ value, label }) => (

<option value={value}> {label}</option>

))

class App extends React.Component {

// declaring state

state = {

firstName: '',

lastName: '',

email: '',

country: '',

tel: '',

dateOfBirth: '',

favoriteColor: '',

weight: '',

gender: '',

file: '',

bio: '',

skills: {

html: false,

css: false,

javascript: false,

},

}

handleChange = (e) => {

/\*

we can get the name and value like: e.target.name, e.target.value

Wwe can also destructure name and value from e.target

const name = e.target.name

const value = e.target.value

\*/

const { name, value, type, checked } = e.target

/\*

[variablename] we can make a value stored in a certain variable could be a key for an object, in this case a key for the state

\*/

if (type === 'checkbox') {

this.setState({

skills: { ...this.state.skills, [name]: checked },

})

} else if (type === 'file') {

console.log(type, 'cehck here')

this.setState({ [name]: e.target.files[0] })

} else {

this.setState({ [name]: value })

}

}

handleSubmit = (e) => {

/\*

e.preventDefault()

stops the default behavior of form element

specifically refreshing of page

\*/

e.preventDefault()

const {

firstName,

lastName,

email,

tel,

dateOfBirth,

favoriteColor,

weight,

country,

gender,

bio,

file,

skills,

} = this.state

const formattedSkills = []

for (const key in skills) {

console.log(key)

if (skills[key]) {

formattedSkills.push(key.toUpperCase())

}

}

const data = {

firstName,

lastName,

email,

tel,

dateOfBirth,

favoriteColor,

weight,

country,

gender,

bio,

file,

skills: formattedSkills,

}

/\*

the is the place where we connect backend api

to send the data to the database

\*/

console.log(data)

}

render() {

// accessing the state value by destrutcturing the state

const {

firstName,

lastName,

email,

tel,

dateOfBirth,

favoriteColor,

weight,

country,

gender,

bio,

} = this.state

return (

<div className='App'>

<h3>Add Student</h3>

<form onSubmit={this.handleSubmit}>

<div className='row'>

<div className='form-group'>

<label htmlFor='firstName'>First Name </label>

<input

type='text'

name='firstName'

value={firstName}

onChange={this.handleChange}

placeholder='First Name'

/>

</div>

<div className='form-group'>

<label htmlFor='lastName'>Last Name </label>

<input

type='text'

name='lastName'

value={this.state.lastName}

onChange={this.handleChange}

placeholder='Last Name'

/>

</div>

<div className='form-group'>

<label htmlFor='email'>Email </label>

<input

type='email'

name='email'

value={email}

onChange={this.handleChange}

placeholder='Email'

/>

</div>

</div>

<div className='form-group'>

<label htmlFor='tel'>Telephone </label>

<input

type='tel'

name='tel'

value={tel}

onChange={this.handleChange}

placeholder='Tel'

/>

</div>

<div className='form-group'>

<label htmlFor='dateOfBirth'>Date of birth </label>

<input

type='date'

name='dateOfBirth'

value={dateOfBirth}

onChange={this.handleChange}

placeholder='Date of Birth'

/>

</div>

<div className='form-group'>

<label htmlFor='favoriteColor'>Favorite Color</label>

<input

type='color'

id='color'

name='color'

value={favoriteColor}

onChange={this.handleChange}

placeholder='Favorite Color'

/>

</div>

<div className='form-group'>

<label htmlFor='weight'>Weight </label>

<input

type='number'

id='weight'

name='weight'

value={weight}

onChange={this.handleChange}

placeholder='Weight in Kg'

/>

</div>

<div>

<label htmlFor='country'>Country</label> <br />

<select name='country' onChange={this.handleChange} id='country'>

{selectOptions}

</select>

</div>

<div>

<p>Gender</p>

<div>

<input

type='radio'

id='female'

name='gender'

value='Female'

onChange={this.handleChange}

checked={gender === 'Female'}

/>

<label htmlFor='female'>Female</label>

</div>

<div>

<input

id='male'

type='radio'

name='gender'

value='Male'

onChange={this.handleChange}

checked={gender === 'Male'}

/>

<label htmlFor='male'>Male</label>

</div>

<div>

<input

id='other'

type='radio'

name='gender'

value='Other'

onChange={this.handleChange}

checked={gender === 'Other'}

/>

<label htmlFor='other'>Other</label>

</div>

</div>

<div>

<p>Select your skills</p>

<div>

<input

type='checkbox'

id='html'

name='html'

onChange={this.handleChange}

/>

<label htmlFor='html'>HTML</label>

</div>

<div>

<input

type='checkbox'

id='css'

name='css'

onChange={this.handleChange}

/>

<label htmlFor='css'>CSS</label>

</div>

<div>

<input

type='checkbox'

id='javascript'

name='javascript'

onChange={this.handleChange}

/>

<label htmlFor='javascript'>JavaScript</label>

</div>

</div>

<div>

<label htmlFor='bio'>Bio</label> <br />

<textarea

id='bio'

name='bio'

value={bio}

onChange={this.handleChange}

cols='120'

rows='10'

placeholder='Write about yourself ...'

/>

</div>

<div>

<input type='file' name='file' onChange={this.handleChange} />

</div>

<div>

<button>Submit</button>

</div>

</form>

</div>

)

}

}

const rootElement = document.getElementById('root')

ReactDOM.render(<App />, rootElement)

Form Validation

What is validation?

The action or process of checking or proving the validity or accuracy of something in this case data.

What is the purpose of validation

The main purpose to validation is to get a desired data from users. In addition, to prevent malicious users and data.

Validation Types

Validation can be done in client side or sever side. At the moment, we are using React which is a front end technology and we use client side validation.A validation can implement using HTML5 built-in validation or using JavaScript(using regular expression).

In the following snippet of code, a validation has been implemented the first field. Try to understand how it works. The onBlur event has been used to check validity when the input is not focused.

// index.js

import React, { Component } from 'react'

import ReactDOM from 'react-dom'

const options = [

{

value: '',

label: '-- Select Country--',

},

{

value: 'Finland',

label: 'Finland',

},

{

value: 'Sweden',

label: 'Sweden',

},

{

value: 'Norway',

label: 'Norway',

},

{

value: 'Denmark',

label: 'Denmark',

},

]

// mapping the options to list(array) of JSX options

const selectOptions = options.map(({ value, label }) => (

<option value={value}> {label}</option>

))

class App extends Component {

// declaring state

state = {

firstName: '',

lastName: '',

email: '',

country: '',

tel: '',

dateOfBirth: '',

favoriteColor: '',

weight: '',

gender: '',

file: '',

bio: '',

skills: {

html: false,

css: false,

javascript: false,

},

touched: {

firstName: false,

lastName: false,

},

}

handleChange = (e) => {

/\*

we can get the name and value like: e.target.name, e.target.value

Wwe can also destructure name and value from e.target

const name = e.target.name

const value = e.target.value

\*/

const { name, value, type, checked } = e.target

/\*

[variablename] we can make a value stored in a certain variable could be a key for an object, in this case a key for the state

\*/

if (type === 'checkbox') {

this.setState({

skills: { ...this.state.skills, [name]: checked },

})

} else if (type === 'file') {

this.setState({ [name]: e.target.files[0] })

} else {

this.setState({ [name]: value })

}

}

handleBlur = (e) => {

const { name, value } = e.target

this.setState({ touched: { ...this.state.touched, [name]: true } })

}

validate = () => {

// Object to collect error feedback and to display on the form

const errors = {

firstName: '',

}

if (

(this.state.touched.firstName && this.state.firstName.length < 3) ||

(this.state.touched.firstName && this.state.firstName.length > 12)

) {

errors.firstName = 'First name must be between 2 and 12'

}

return errors

}

handleSubmit = (e) => {

/\*

e.preventDefault()

stops the default behavior of form element

specifically refreshing of page

\*/

e.preventDefault()

const {

firstName,

lastName,

email,

country,

gender,

tel,

dateOfBirth,

favoriteColor,

weight,

bio,

file,

skills,

} = this.state

const formattedSkills = []

for (const key in skills) {

console.log(key)

if (skills[key]) {

formattedSkills.push(key.toUpperCase())

}

}

const data = {

firstName,

lastName,

email,

country,

gender,

tel,

dateOfBirth,

favoriteColor,

weight,

bio,

file,

skills: formattedSkills,

}

/\*

the is the place where we connect backend api

to send the data to the database

\*/

console.log(data)

}

render() {

// accessing the state value by destrutcturing the state

// the noValidate attribute on the form is to stop the HTML5 built-in validation

const { firstName } = this.validate()

return (

<div className='App'>

<h3>Add Student</h3>

<form onSubmit={this.handleSubmit} noValidate>

<div className='row'>

<div className='form-group'>

<label htmlFor='firstName'>First Name </label>

<input

type='text'

name='firstName'

value={this.state.firstName}

onChange={this.handleChange}

onBlur={this.handleBlur}

placeholder='First Name'

/> <br />

<small>{firstName}</small>

</div>

<div className='form-group'>

<label htmlFor='lastName'>Last Name </label>

<input

type='text'

name='lastName'

value={this.state.lastName}

onChange={this.handleChange}

placeholder='Last Name'

/>

</div>

<div className='form-group'>

<label htmlFor='email'>Email </label>

<input

type='email'

name='email'

value={this.state.email}

onChange={this.handleChange}

placeholder='Email'

/>

</div>

</div>

<div className='form-group'>

<label htmlFor='tel'>Telephone </label>

<input

type='tel'

name='tel'

value={this.state.tel}

onChange={this.handleChange}

placeholder='Tel'

/>

</div>

<div className='form-group'>

<label htmlFor='dateOfBirth'>Date of birth </label>

<input

type='date'

name='dateOfBirth'

value={this.state.dateOfBirth}

onChange={this.handleChange}

placeholder='Date of Birth'

/>

</div>

<div className='form-group'>

<label htmlFor='favoriteColor'>Favorite Color</label>

<input

type='color'

id='favoriteColor'

name='favoriteColor'

value={this.state.favoriteColor}

onChange={this.handleChange}

placeholder='Favorite Color'

/>

</div>

<div className='form-group'>

<label htmlFor='weight'>Weight </label>

<input

type='number'

id='weight'

name='weight'

value={this.state.weight}

onChange={this.handleChange}

placeholder='Weight in Kg'

/>

</div>

<div>

<label htmlFor='country'>Country</label> <br />

<select name='country' onChange={this.handleChange} id='country'>

{selectOptions}

</select>

</div>

<div>

<p>Gender</p>

<div>

<input

type='radio'

id='female'

name='gender'

value='Female'

onChange={this.handleChange}

checked={this.state.gender === 'Female'}

/>

<label htmlFor='female'>Female</label>

</div>

<div>

<input

id='male'

type='radio'

name='gender'

value='Male'

onChange={this.handleChange}

checked={this.state.gender === 'Male'}

/>

<label htmlFor='male'>Male</label>

</div>

<div>

<input

id='other'

type='radio'

name='gender'

value='Other'

onChange={this.handleChange}

checked={this.state.gender === 'Other'}

/>

<label htmlFor='other'>Other</label>

</div>

</div>

<div>

<p>Select your skills</p>

<div>

<input

type='checkbox'

id='html'

name='html'

onChange={this.handleChange}

/>

<label htmlFor='html'>HTML</label>

</div>

<div>

<input

type='checkbox'

id='css'

name='css'

onChange={this.handleChange}

/>

<label htmlFor='css'>CSS</label>

</div>

<div>

<input

type='checkbox'

id='javascript'

name='javascript'

onChange={this.handleChange}

/>

<label htmlFor='javascript'>JavaScript</label>

</div>

</div>

<div>

<label htmlFor='bio'>Bio</label> <br />

<textarea

id='bio'

name='bio'

value={this.state.bio}

onChange={this.handleChange}

cols='120'

rows='10'

placeholder='Write about yourself ...'

/>

</div>

<div>

<input type='file' name='file' onChange={this.handleChange} />

</div>

<div>

<button>Submit</button>

</div>

</form>

</div>

)

}

}

const rootElement = document.getElementById('root')

ReactDOM.render(<App />, rootElement)