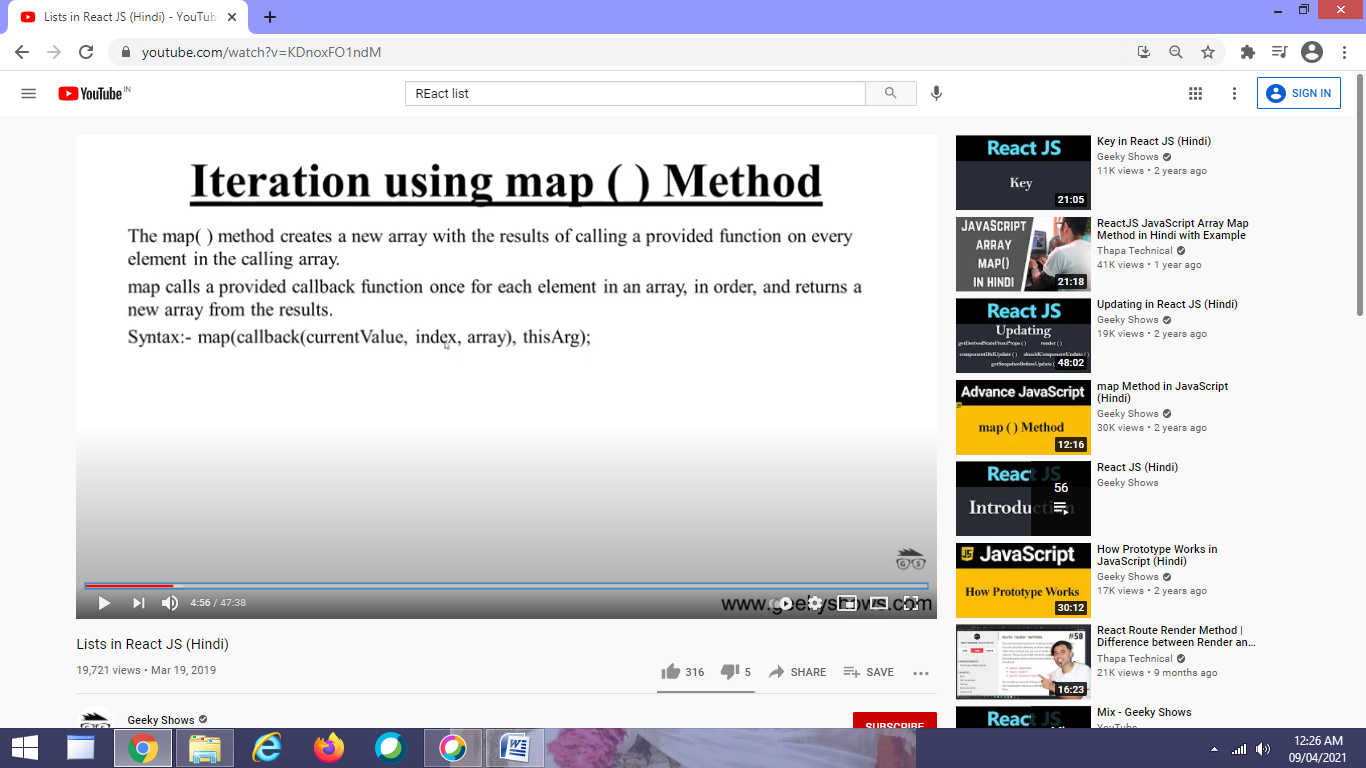


State is an object and users is a property , it is containing an array.

Second property is isLoggedIn

map() :- is basically used as an iteration in React.



ex :- map((num,index)=>{return num})

//declaration and initialization of Array

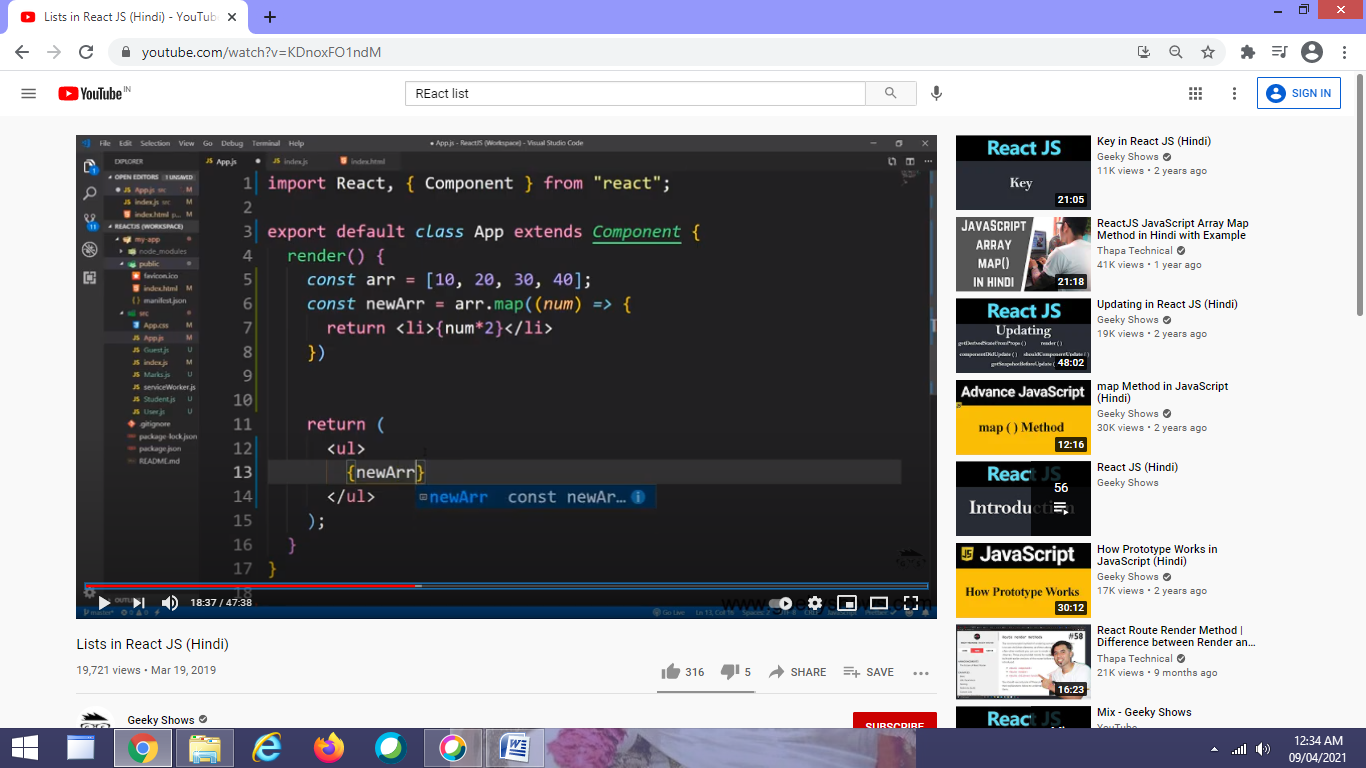
const arr=[10,20,30,40];

// Using Array Map method

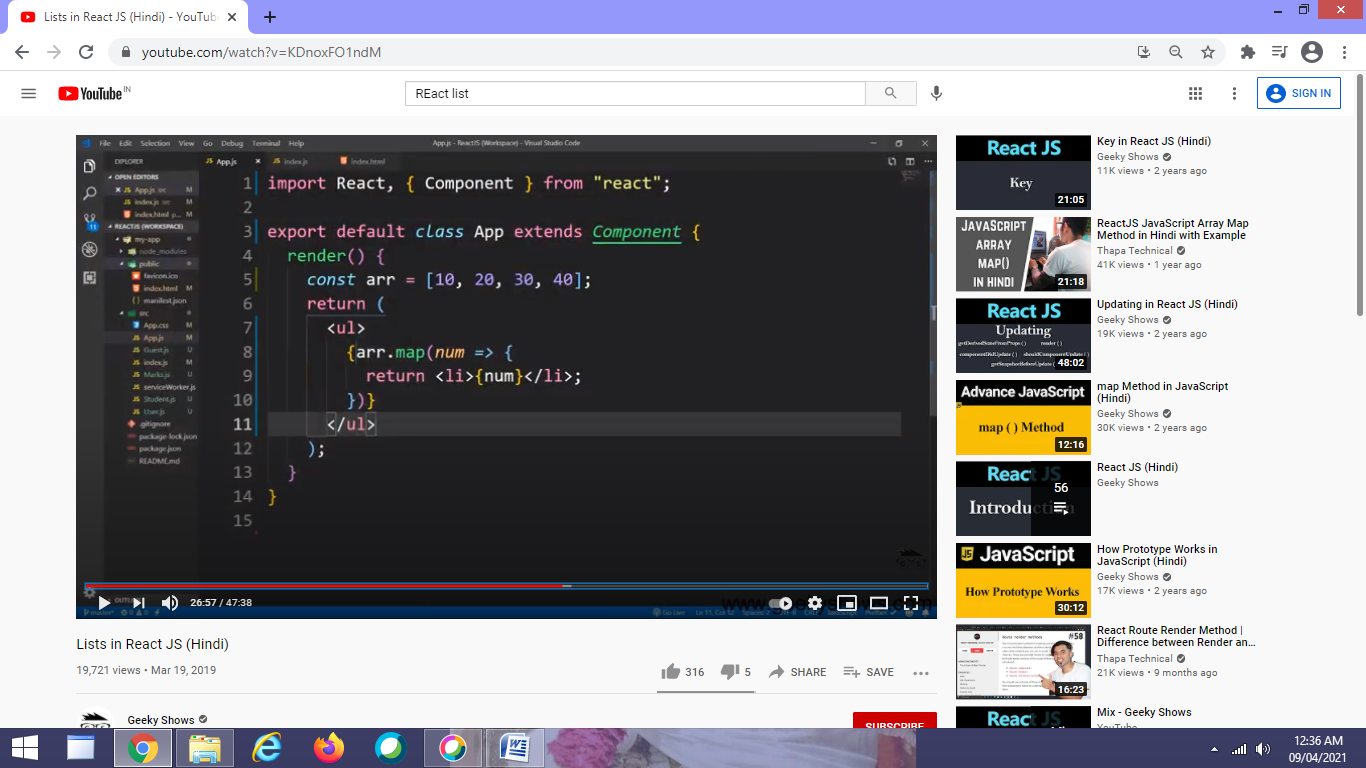
const newArr=arr.map(num=>{

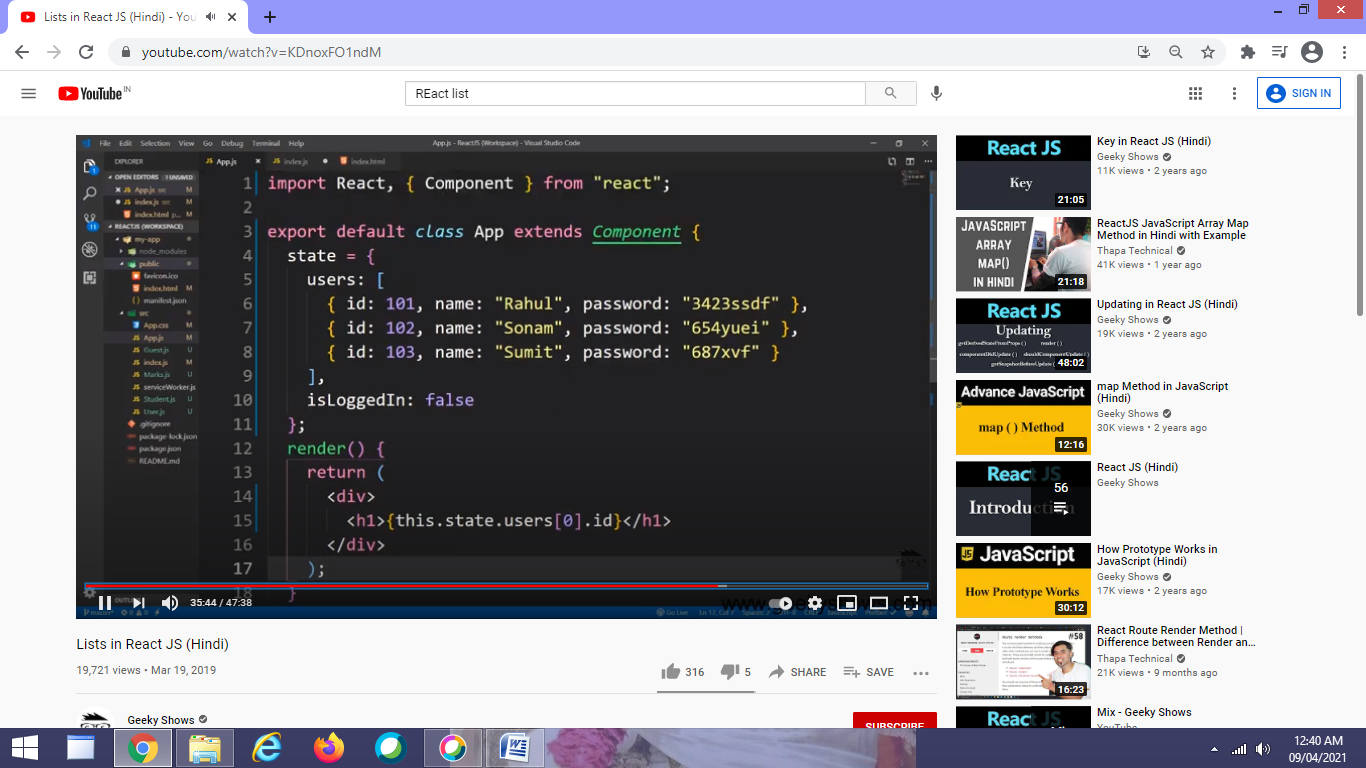
return <li> {num\*2}</li>;

});

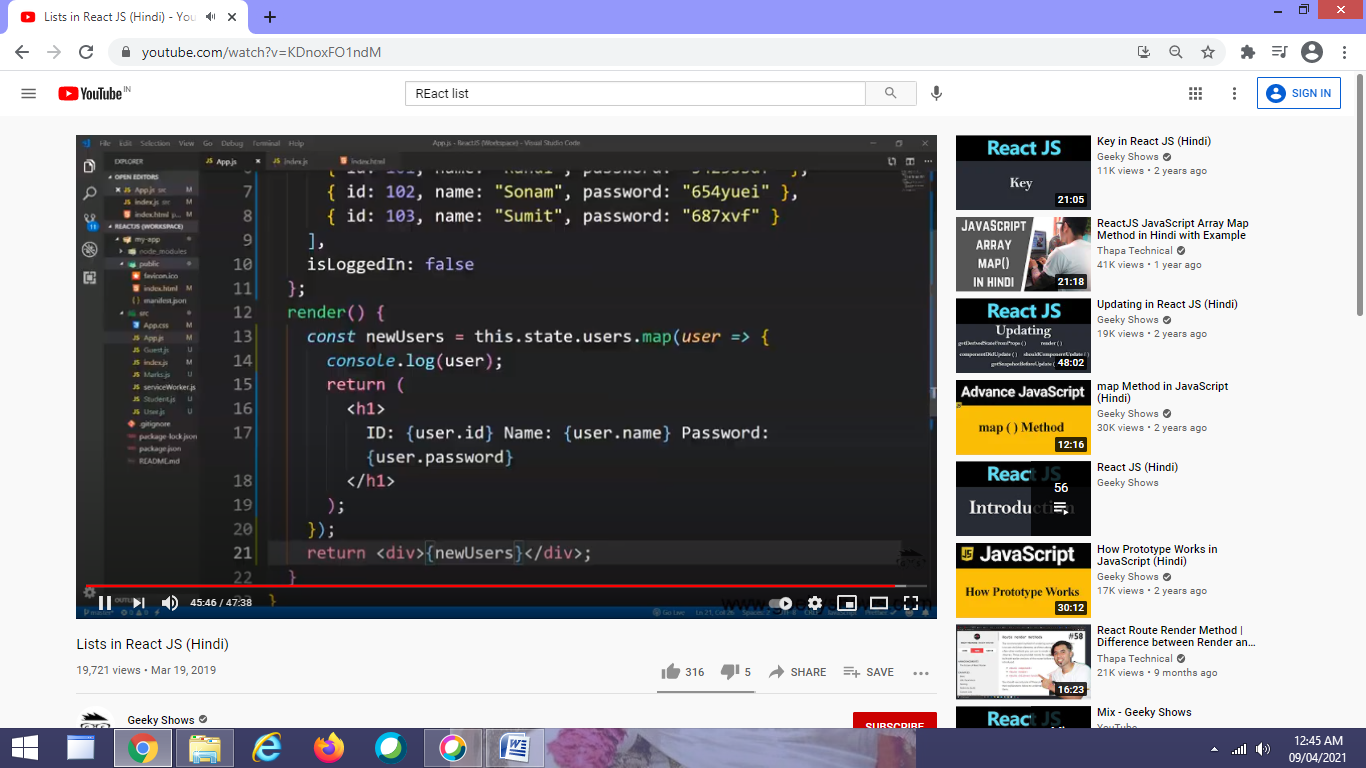


Another way :- Embedded map in jsx





Or



React JS Forms

App.js

import React, {useState} from “react”;

const App=()=>{

const [name,setName]=useState(“”);

const inputEvent =(event)=>{

console.log(“clicked”);

console.log(event.target.value);

setName(event.target.value);

};

return(

<div>

<h1> Hello {name}</h1>

<input type=’text’ placeholder=”Enter your Name” value={name}/>

<button> Click Me </button>

);

}

export default App;

In React we have Controlled and Vs Uncontrolled Component

In HTML, form elements such as <input>, <textarea>, and <select> typically maintain their own state and update it based on user input. In React, mutable state is typically kept in the state property of components, and only updated with [setState()](https://reactjs.org/docs/react-component.html#setstate).

We can combine the two by making the React state be the “single source of truth”. Then the React component that renders a form also controls what happens in that form on subsequent user input. An input form element whose value is controlled by React in this way is called a “controlled component”.

**Implementing Hooks in Functional component**

Hooks are new addition in React 16.8. They let us use state and other React features without writing a class. A Hook is a special function that let us “hook” into React features. For example, useState is a Hook that lets us add React state to function components.

Syntax

const[variablename, functionName]=useState();

Example on Hooks

Import React, {useState} from “react”;

const App=()=>{

const [myText,setMyText]=useState(‘Hooks Demo’);

const changeText=()=>{

setMyText(‘Learning with React – Hooks’);

}

console.log(myText);

return(

<div>

<h1>{myText} </h1>

<button onClick={changeText}> Click Here </button>

</div>

)

}

export default App;

Instead use defaultValue

<input type=’text’ placeholder=”Enter your Name” onChange={inputEvent} value={name}/>

Single source of truth :- {name}

On buttons click event -- Form event

const [name,setName]=useState(“”);

const[fullName,setFullName]=useState(“”);

<h1> Hello {fullname}</h1>

<button onClick={onSubmit}> Click Me </button>

const onSubmit=()=>{

setFullName(name);

}

Change the code and see the behaviour of the react component

const[lastName,setLastName]=useState();

const[newFullName,setNewFullName]=useState(“”);

const onSubmits=(event)=>{

event.preventDefault();

setFullName(name);

setNewFullName (lastName);

};

const inputEvent =(event)=>{

console.log(event.target.value);

setName(event.target.value);

};

const inputEventTwo =(event)=>{

console.log(event.target.value);

setLastName(event.target.value);

};

<div>

<form onSubmit={onSubmits}>

<div>

<h1> Hello {fullname} {newFullName}</h1>

<input type=’text’ placeholder=”Enter your Name” onChange={inputEvent} value={name}/>

<br/>

<input type=’text’ placeholder=”Enter your Last Name” onChange={inputEventTwo} value={lastName}/>

<br/>

<button onClick={onSubmit}> Click Me </button>

</div>

</form>

</div>

Creating Basic Structure or the Layout for other React components

<Router/> :- this can include many routes that render inclusively the matching url path

<Router>

<Route path=”/” component={Home}/>

<Route path=”/Overview” component={Overview}/>

<Route path=”/Contact” component={Contact}/>

</Router>

Routing in React

<Switch/> :- will only render the first component(route) that matches /includes the path.

<Switch>

<Route exact path=”/” component={Home}/>

<Route path=”/Overview” component={Overview}/>

<Route path=”/Contact” component={Contact}/>

</Switch>

http://localhost:3000/Overview

React JS Forms

Steps

import React, {useState} from 'react';

const App=()=>{

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

 const inputEvent =(event)=>{

    console.log('clicked');

    console.log(event.target.value);

    setName(event.target.value);

    };

    return(

    <div>

    <h1> Hello {name}</h1>

    <input type='text' placeholder='Enter your Name' onChange={inputEvent}  value={name}/>

    <button> Click Me </button>

    </div>

    );

    }

    export default App;

Complete code

import React, {useState} from 'react';

const App=()=>{

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

    const[fullName,setFullName]=useState('');

    const[lastName,setLastName]=useState('');

    const[newFullName,setNewFullName]=useState('');

const inputEvent =(event)=>{

console.log('clicked');

console.log(event.target.value);

setName(event.target.value);

};

const inputEventTwo =(event)=>{

    console.log(event.target.value);

    setLastName(event.target.value);

    };

const onSubmits=(event)=>{

    event.preventDefault();

    setFullName(name);

    setNewFullName (lastName);

};

const onSubmit=()=>{

    setFullName(name);

    };

return(

<div>

<form onSubmit={onSubmits}>

<div>

<h1> Hello {fullName} {newFullName}</h1>

<input type='text' placeholder='Enter your Name' onChange={inputEvent} value={name}/>

<br/>

<input type='text' placeholder='Enter your Last Name' onChange={inputEventTwo} value={lastName}/>

<br/>

<button onClick={onSubmit}> Click Me </button>

</div>

</form>

</div>

)

}

export default App;

**React form validation**

**Form validation** in React allows an error message to be displayed if the user has not correctly filled out the form with the expected type of input.

There are several ways to validate forms in React; however, in this example we will focus on creating a validator function with validation rules.

The form validation rules are applied in the handleChange function that handles input from users.

import React from 'react';

const validEmailRegex = RegExp(

  /^(([^<>()\[\]\.,;:\s@\"]+(\.[^<>()\[\]\.,;:\s@\"]+)\*)|(\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^<>()[\]\.,;:\s@\"]{2,})$/i

);

const validateForm = errors => {

  let valid = true;

  Object.values(errors).forEach(val => val.length > 0 && (valid = false));

  return valid;

};

export default class FormValidation extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      fullName: null,

      email: null,

      password: null,

      errors: {

        fullName: '',

        email: '',

        password: '',

      }

    };

  }

  handleChange = (event) => {

    event.preventDefault();

    const { name, value } = event.target;

    let errors = this.state.errors;

    switch (name) {

      case 'fullName':

        errors.fullName =

          value.length < 5

            ? 'Full Name must be at least 5 characters long!'

            : '';

        break;

      case 'email':

        errors.email =

          validEmailRegex.test(value)

            ? ''

            : 'Email is not valid!';

        break;

      case 'password':

        errors.password =

          value.length < 8

            ? 'Password must be at least 8 characters long!'

            : '';

        break;

      default:

        break;

    }

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

  }

  handleSubmit = (event) => {

    event.preventDefault();

    if(validateForm(this.state.errors)) {

      console.info('Valid Form')

    }else{

      console.error('Invalid Form')

    }

  }

  render() {

    const {errors} = this.state;

    return (

      <div className='wrapper'>

        <div className='form-wrapper'>

          <h2>Create Account</h2>

          <form onSubmit={this.handleSubmit} noValidate>

            <div className='fullName'>

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

              <input type='text' name='fullName' onChange={this.handleChange} noValidate />

              {errors.fullName.length > 0 &&

                <span className='error'>{errors.fullName}</span>}

            </div>

            <div className='email'>

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

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

              {errors.email.length > 0 &&

                <span className='error'>{errors.email}</span>}

            </div>

            <div className='password'>

              <label htmlFor="password">Password</label>

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

              {errors.password.length > 0 &&

                <span className='error'>{errors.password}</span>}

            </div>

            <div className='submit'>

              <button>Create</button>

            </div>

          </form>

        </div>

      </div>

    );

  }

}

## Explanation

Upon clicking the Create button, the console tab shows whether or not the form is valid. The lines of code that implement this functionality are explained below.

* The **validEmailRegex variable** holds the [regex](https://www.educative.io/shoteditor/4879445779283968/preview) rules to check whether or not the given input is a valid email.
* The **validateForm function** checks whether or not any of the input fields have any errors. If there are any errors, the function outputs it on the console.
* The actual rules of validation are applied in the **handleChange function**. The function has a switch-case implementation in it ​which applies rules specific to the input field. For example, if the input field is fullName , then the function checks the length of the value in that​ field. If the length is less than 5, it sets the error message of the associated field. Otherwise, the error message of the associated field is set to an empty field.