Form handling:

Form handling with plain java script:

```
import  './mystyle.css';
export default function Main(){


   const handlesubmit = (e)=>{
      e.preventDefault();
      let inpt = document.getElementsByTagName('input');
      let username = inpt[0].value;
      let password = inpt[0].value;
      console.log(username,password)
   }
    return(
      <>
        <form onSubmit={handlesubmit}>
          <label>Username</label>
          <input type="text" placeholder="enter username here"/>
          <label>password</label>
          <input type="text" placeholder="enter password here" />
          <button type="submit">Login</button>
        </form>
      </>
   )
}
```

But in the above program we are printing just username and password. But while sending to the backend API, we need to pass a javascript object. So we have to create a js object and send it back to the backend API as given below:

```
import  './mystyle.css';
```

```
export default function Main(){


  const handlesubmit = (e)=>{
     e.preventDefault();
     let inpt = document.getElementsByTagName('input');
     let username = inpt[0].value;
     let password = inpt[0].value;
     let loginObj = {'username':username,'password':password};
     console.log('Sending this object to backend API :: ',loginObj);
  }
   return(
     <>
      <form onSubmit={handlesubmit}>
        <label>Username</label>
        <input type="text" placeholder="enter username here"/>
        <label>password</label>
        <input type="text" placeholder="enter password here" />
        <button type="submit">Login</button>
      </form>
     </>
  )
}
```

Form handling with React way:

```
import React from 'react';
import  './mystyle.css';
export default function Main(){


  const [login,setLogin] = React.useState({
     username:'',
```

Here, if we run this program, we can not type anything in input. Because we are setting username state to the input and even though we are typing anything, the username will be still empty so it will not reflect in the input. That's why we have to change your state(username and password) on input

```
    password:''
  })
   const handlesubmit = ()=>{
    console.log(login)
  }
  return(
    <>
      <form onSubmit={handlesubmit}>
        <label>Username</label>
        <input type="text" placeholder="enter
username here" value={login.username}/>
        <label>password</label>
        <input type="text" placeholder="enter
password here" value={login.password}/>
        <button type="submit">Login</button>
      </form>
    </>
  )
}
```

change.

```
import React from 'react';
import  './mystyle.css';
export default function Main(){

  const [login,setLogin] = React.useState({
    username:'',
    password:''
  })
   const handlechange = (e)=>{
    let selectedField = e.target.name;
    let enteredValue = e.target.value;
```

```
      if(selectedField == 'username'){
          setLogin({
              username:enteredValue,
              password:login.password
          })
      }
      if(selectedField == 'password'){
          setLogin({
              username:login.username,
              password:enteredValue
          })
      }
    }
    const handlesubmit = (e)=>{
        e.preventDefault();
        console.log(login)
    }
    return(
        <>
          <form onSubmit={handlesubmit}>
            <label>Username</label>
            <input type="text" placeholder="enter username here" name="username" value={login.username}
onChange={handlechange}/>
            <label>password</label>
            <input type="text" placeholder="enter password here" name="password" value={login.password}
onChange={handlechange}/>
            <button type="submit">Login</button>
          </form>
        </>
    )
}
```

But if you observe the above code(highlighted part), we are checking the condition for each field in the form.

But the form has a lot of fields so checking for each field is not a good idea. So the solution is in next example:

```
import React from 'react';
import  './mystyle.css';
export default function Main(){


  const [login,setLogin] = React.useState({
     username:'',
     password:''
  })
   const handlechange = (e)=>{
     let selectedField = e.target.name;
     let enteredValue = e.target.value;
     setLogin({...login, [selectedField]:enteredValue})
  }
  const handlesubmit = (e)=>{
     e.preventDefault();
     console.log(login)
  }
  return(
     <>
       <form onSubmit={handlesubmit}>
         <label>Username</label>
         <input type="text" placeholder="enter username here" name="username" value={login.username}
onChange={handlechange}/>
         <label>password</label>
         <input type="text" placeholder="enter password here" name="password" value={login.password}
onChange={handlechange}/>
         <button type="submit">Login</button>
       </form>
     </>
```

```
  )
}
```

Registration form:

```
import React from 'react';
import  './mystyle.css';
export default function Main(){


  const [user,setUser] = React.useState({
      firstName:'',
      lastName:'',
      age:''
  })
   const handlechange = (e)=>{
      let selectedField = e.target.name;
      let enteredValue = e.target.value;
      setUser({...user, [selectedField]:enteredValue})
  }
  const handlesubmit = (e)=>{
      e.preventDefault();
      console.log(user)
  }
  return(
      <>
        <form onSubmit={handlesubmit}>
          <label>Firstname</label>
          <input type="text" placeholder="enter firstname here" name="firstName" value={user.firstName}
onChange={handlechange}/>
          <label>Lastname</label>
          <input type="text" placeholder="enter lastname here" name="lastName" value={user.lastName}
```

```
onChange={handlechange}/>
          <label>Age</label>
          <input type="text" placeholder="enter age here" name="age" value={user.age}
onChange={handlechange}/>
          <button type="submit">Login</button>
        </form>
      </>
  )
}
```

Register form with Select options

```
import React from 'react';
import  './mystyle.css';
export default function Main(){

  const [user,setUser] = React.useState({
     firstName:'',
     lastName:'',
     age:'',
     state:''
  })
   const handlechange = (e)=>{
     let selectedField = e.target.name;
     let enteredValue = e.target.value;
     setUser({...user, [selectedField]:enteredValue})
  }
  const handlesubmit = (e)=>{
     e.preventDefault();
     console.log(user)
  }
  return(
     <>
```

```
        <form onSubmit={handlesubmit}>
          <label>Firstname</label>
          <input type="text" placeholder="enter firstname here" name="firstName" value={user.firstName}
onChange={handlechange}/>
          <label>Lastname</label>
          <input type="text" placeholder="enter lastname here" name="lastName" value={user.lastName}
onChange={handlechange}/>
          <label>Age</label>
          <input type="text" placeholder="enter age here" name="age" value={user.age}
onChange={handlechange}/>
          <label>Select state</label>
          <select onChange={handlechange} name="state">
             <option value="MH">Maharashtra</option>
             <option value="AP">Andra pradesh</option>
             <option value="TN">Tenlagna</option>
          </select>
          <button type="submit">Login</button>
        </form>
      </>
   )
}
```

RegisterForm with Radio button

```
import React from 'react';
import  './mystyle.css';
export default function Main(){

   const [user,setUser] = React.useState({
      firstName:'',
      lastName:'',
```

```
    age:'',
    gender:''
  })
   const handlechange = (e)=>{
    let selectedField = e.target.name;
    let enteredValue = e.target.value;
    console.log(enteredValue)
    setUser({...user, [selectedField]:enteredValue})
  }
  const handlesubmit = (e)=>{
    e.preventDefault();
    console.log(user)
  }
  return(
    <>
      <form onSubmit={handlesubmit}>
        <label>Firstname</label>
        <input type="text" placeholder="enter firstname here" name="firstName" value={user.firstName}
onChange={handlechange}/>
        <label>Lastname</label>
        <input type="text" placeholder="enter lastname here" name="lastName" value={user.lastName}
onChange={handlechange}/>
        <label>Age</label>
        <input type="text" placeholder="enter age here" name="age" value={user.age}
onChange={handlechange}/>
        <label>Select Gender</label>
        <input type="radio" name="gender" value="male" onChange={handlechange}/>Male
        <input type="radio" name="gender" value="female" onChange={handlechange}/>Female
        <button type="submit">Login</button>
      </form>
    </>
  )
```

```
}
```

Register form with Date:

```
import React from 'react';
import  './mystyle.css';
export default function Main(){

  const [user,setUser] = React.useState({
     firstName:'',
     lastName:'',
     age:'',
     joinDate:''
  })
   const handlechange = (e)=>{
     let selectedField = e.target.name;
     let enteredValue = e.target.value;
     console.log(enteredValue)
     setUser({...user, [selectedField]:enteredValue})
  }
  const handlesubmit = (e)=>{
     e.preventDefault();
     console.log(user)
  }
  return(
     <>
       <form onSubmit={handlesubmit}>
         <label>Firstname</label>
         <input type="text" placeholder="enter firstname here" name="firstName" value={user.firstName}
onChange={handlechange}/>

         <label>Lastname</label>
```

```
        <input type="text" placeholder="enter lastname here" name="lastName" value={user.lastName}
onChange={handlechange}/>

        <label>Age</label>
        <input type="text" placeholder="enter age here" name="age" value={user.age}
onChange={handlechange}/>

        <label>Joining Date</label>
        <input type="date" placeholder="enter age here" name="joinDate" value={user.joinDate}
onChange={handlechange}/>
        <button type="submit">Login</button>
      </form>
    </>
  )
}
```

Ref in React:

Ref is used to access any DOM element. We can directly manipulate DOM elements using ref.

```
import React from 'react';
import  './mystyle.css';
export default function Main(){
  const myref = React.useRef(null);
  const check = () =>{
    console.log(myref.current.innerHTML)
  }
  return(
    <>
      <h1 ref={myref}>This is text</h1>
      <button onClick={check}>Click here</button>
    </>
  )
}
```

Here we are reading innerHTML or content of h1 tag.

```
import React from 'react';
import  './mystyle.css';
export default function Main(){
   const myref = React.useRef(null);
   const check = () =>{
      myref.current.innerHTML = 'This text added by ref'
   }
   return(
      <>
         <h1 ref={myref}>This is text</h1>
         <button onClick={check}>Click here</button>
      </>
   )
}
```

Here, we are changing text of h1

In the below program we are changing the color of the h1 text.

```
import React from 'react';
import  './mystyle.css';
export default function Main(){
   const myref = React.useRef(null);
   const check = () =>{
      myref.current.style.color="red"
   }
   return(
      <>
         <h1 ref={myref}>This is text</h1>
         <button onClick={check}>Click here</button>
      </>
   )
}
```

So, using ref we can manipulate any element in a DOM.

Now we are going to use ref to read form data.

```
import React from 'react';
import  './mystyle.css';
export default function Main(){

   const firstNameRef = React.useRef(null);
   const lastNameRef = React.useRef(null);
    const handlesubmit = (e)=>{
       e.preventDefault();
       let firstName = firstNameRef.current.value;
       let lastName = lastNameRef.current.value;
       console.log(firstName,lastName)
   }
   return(
       <>
         <form onSubmit={handlesubmit}>
           <label>Firstname</label>
           <input type="text" placeholder="enter firstname here" name="firstName" ref={firstNameRef}/>

           <label>Lastname</label>
           <input type="text" placeholder="enter lastname here" name="lastName" ref={lastNameRef}/>

           <button type="submit">Login</button>
         </form>
       </>
   )
}
```

Controlled vs Uncontrolled component:

Uncontrolled Components: Uncontrolled Components are the components that are not controlled by the React state and are handled by the DOM (Document Object Model). So in order to access any value that has been entered we take the help of refs.

```
import React from 'react';
import  './mystyle.css';
export default function Main(){

  const firstNameRef = React.useRef(null);
  const lastNameRef = React.useRef(null);
   const handlesubmit = (e)=>{
     e.preventDefault();
     let firstName = firstNameRef.current.value;
     let lastName = lastNameRef.current.value;
     console.log(firstName,lastName)
  }
  return(
     <>
       <form onSubmit={handlesubmit}>
         <label>Firstname</label>
         <input type="text" placeholder="enter firstname here" name="firstName" ref={firstNameRef}/>

         <label>Lastname</label>
         <input type="text" placeholder="enter lastname here" name="lastName" ref={lastNameRef}/>

         <button type="submit">Login</button>
       </form>
     </>
  )
}
```

Controlled Components: In React, Controlled Components are those in which form's data is handled by the component's state. It takes its current value through props and makes changes through callbacks like onClick, onChange, etc. A parent component manages its own state and passes the new values as props to the controlled component.

```
import React from 'react';
import  './mystyle.css';
export default function Main(){

  const [user,setUser] = React.useState({
      firstName:'',
      lastName:'',
      age:'',
      joinDate:''
  })
   const handlechange = (e)=>{
      let selectedField = e.target.name;
      let enteredValue = e.target.value;
      console.log(enteredValue)
      setUser({...user, [selectedField]:enteredValue})
  }
  const handlesubmit = (e)=>{
      e.preventDefault();
      console.log(user)
  }
  return(
      <>
        <form onSubmit={handlesubmit}>
          <label>Firstname</label>
          <input type="text" placeholder="enter firstname here" name="firstName" value={user.firstName}
onChange={handlechange}/>

          <label>Lastname</label>
          <input type="text" placeholder="enter lastname here" name="lastName" value={user.lastName}
onChange={handlechange}/>

          <label>Age</label>
```

```
        <input type="text" placeholder="enter age here" name="age" value={user.age}
onChange={handlechange}/>

        <label>Joining Date</label>
        <input type="date" placeholder="enter age here" name="joinDate" value={user.joinDate}
onChange={handlechange}/>
        <button type="submit">Login</button>
      </form>
    </>
  )
}
```

React form validation without using any library

```
import React from 'react';
import  './mystyle.css';
export default function Main(){

  const [user,setUser] = React.useState({
    firstName:'',
    lastName:''
  })
  const [fnameError,setFnameError] = React.useState('');
  const [lnameError,setLnameError] = React.useState('');
   const handlechange = (e)=>{
    let selectedField = e.target.name;
    let enteredValue = e.target.value;
    setUser({...user, [selectedField]:enteredValue})
  }
  const handlesubmit = (e)=>{
    e.preventDefault();
    setFnameError('')
```

```
        setLnameError('')
        if(user.firstName == ''){
            setFnameError('first name can not be blank')
        }
        if(user.lastName == ''){
            setLnameError('last name can not be blank')
        }
        console.log(user)
    }
    return(
        <>
          <form onSubmit={handlesubmit}>
            <label>Firstname</label>
            <input type="text" placeholder="enter firstname here" name="firstName" value={user.firstName}
onChange={handlechange}/>
            <p className='error-p'>{fnameError}</p>

            <label>Lastname</label>
            <input type="text" placeholder="enter lastname here" name="lastName" value={user.lastName}
onChange={handlechange}/>
            <p className='error-p'>{lnameError}</p>

            <button type="submit">Login</button>
          </form>
        </>
    )
}
```

Formik Validation:

Formik is a third party React form library. It provides basic form programming and validation. It is based on a controlled component and greatly reduces the time to do form programming.

```
npm i formik
npm i yup
```

Example:

```
import {  Field, Form, Formik } from "formik";
import React from "react";
import * as Yup from "yup";
export default function AddUser() {

  return (
    <>
      <h1>Validation with Formik</h1>
      <Formik
        enableReinitialize={true}
        initialValues={{
          firstName: '',
          lastName: ''
        }}
        validationSchema={
          Yup.object().shape({
            firstName: Yup.string().trim().min(2, 'min required').required('need to assign'),
            lastName: Yup.string().trim().min(3, 'min add').required('required here')
          })
        }
        onSubmit={(data) => {
          console.log(data)
        }}
      >
        {({}) => (
          <Form>
            <Field type="firstName" name="firstName" placeholder="enter firstname here" />

            <Field type="lastName" name="lastName" placeholder="enter lastname here" />

            <input type="submit" value="Add" />
          </Form>
```

```
            )}
        </Formik>
      </>
  )
}
```

Explanation:
initialValues: whatever values we are providing here will be set to the inputs on page load. It is mostly useful in update forms.

ValidationSchema: here we provide actual validations of our form. It will be applied to inputs of our form. In order to apply this validationSchema, instead of input we have to use Field tag.

onSubmit : this function will get called only when we follow all the validations.

In the above program, the form will not submit until we follow all the validations. But we are not showing error message then how user will know the issue, so we have to show error message as given below:

```
import { ErrorMessage, Field, Form, Formik } from "formik";
import React from "react";
import * as Yup from "yup";
export default function AddUser() {

  return (
    <>
      <h1>Validation with Formik</h1>
      <Formik
        initialValues={{
          firstName: '',
          lastName: ''
        }}
        validationSchema={
          Yup.object().shape({
            firstName: Yup.string().trim().min(2, 'min required').required('need to assign'),
            lastName: Yup.string().trim().min(3, 'min add').required('required here')
          })
```

```
            }
            onSubmit={(data) => {
                console.log(data)
            }}
        >
            {({
                errors
            }) => (
                <Form>
                    <Field type="firstName" name="firstName" placeholder="enter firstname here" /><br></br>
                    <span style={{ color: 'red' }}>{errors.firstName}</span><br></br>
                    <Field type="lastName" name="lastName" placeholder="enter lastname here" /><br></br>
                    <span style={{ color: 'red' }}>{errors.lastName}</span><br></br>
                    <input type="submit" value="Add" />
                </Form>
            )}
        </Formik>
    </>
  )
}
```

Formik has provided one more feature to display error messages, which is the **ErrorMessage** tag.

```
import { ErrorMessage, Field, Form, Formik } from "formik";
import React from "react";
import * as Yup from "yup";
export default function AddUser() {

  return (
    <>
      <h1>Validation with Formik</h1>
```

```
<Formik
  initialValues={{
    firstName: '',
    lastName: ''
  }}
  validationSchema={
    Yup.object().shape({
      firstName: Yup.string().trim().min(2, 'min required').required('need to assign'),
      lastName: Yup.string().trim().min(3, 'min add').required('required here')
    })
  }
  onSubmit={(data) => {
    console.log(data)
  }}
>
  {({
    errors
  }) => (
    <Form>
      <Field type="firstName" name="firstName" placeholder="enter firstname here" /><br></br>
      <ErrorMessage
        id="firstName-error"
        name="firstName"
        component="div"
        className="invalid-data"
      />
      <Field type="lastName" name="lastName" placeholder="enter lastname here" /><br></br>
      <ErrorMessage
        id="lastName-error"
        name="firstName"
        component="div"
        className="invalid-data"
```

```
                    />
                        <input type="submit" value="Add" />
                    </Form>
                )}
            </Formik>
        </>
    )
}
```

Here **in ErrorMessage tag name attribute should match with the key in validationSchema.**

Another case where we want to keep submit button disable until you enter correct data then we can do following way:

```
import { ErrorMessage, Field, Form, Formik } from "formik";
import React from "react";
import * as Yup from "yup";
export default function AddUser() {

  return (
    <>
      <h1>Validation with Formik</h1>
      <Formik
        initialValues={{
          firstName: '',
          lastName: ''
        }}
        validationSchema={
          Yup.object().shape({
            firstName: Yup.string().trim().min(2, 'min required').required('need to assign'),
            lastName: Yup.string().trim().min(3, 'min add').required('required here')
          })
```

```
            }
            onSubmit={(data) => {
                console.log(data)
            }}
        >
            {({
                errors,
                dirty
            }) => (
                <Form>
                    <Field type="firstName" name="firstName" placeholder="enter firstname here" /><br></br>
                    <ErrorMessage
                        id="firstName-error"
                        name="firstName"
                        component="div"
                        className="invalid-data"
                    />
                    <Field type="lastName" name="lastName" placeholder="enter lastname here" /><br></br>
                    <ErrorMessage
                        id="lastName-error"
                        name="firstName"
                        component="div"
                        className="invalid-data"
                    />
                    <input type="submit" value="Add" style={{ background: 'green', border: 'none', padding: '8px'
}} disabled={!dirty} />
                </Form>
            )}
        </Formik>
    </>
  )
}
```

Email validation:

```
import { ErrorMessage, Field, Form, Formik } from "formik";
import React from "react";
import * as Yup from "yup";
export default function AddUser() {

  return (
    <>
      <h1>Validation with Formik</h1>
      <Formik
        initialValues={{
          email: ''
        }}
        validationSchema={
          Yup.object().shape({
            email: Yup.string().trim().min(2, 'min required').email('enter valid email')
          })
        }
        onSubmit={(data) => {
          console.log(data)
        }}
      >
        {({
          errors
        }) => (
          <Form>
            <Field type="text" name="email" placeholder="enter email here" /><br></br>
            <ErrorMessage
              id="firstName-error"
              name="email"
              component="div"
```

```
                    className="invalid-data"
                />
                <input type="submit" value="Add" style={{ background: 'green', border: 'none', padding: '8px' }} />
            </Form>
        )}
    </Formik>
  </>
 )
}
```

Working with select with Formik

```
import { ErrorMessage, Field, Form, Formik } from "formik";
import React from "react";
import * as Yup from "yup";
export default function AddUser() {

  return (
    <>
      <h1>Validation with Formik</h1>
      <Formik
        initialValues={{
          state: ''
        }}
        validationSchema={
          Yup.object().shape({
            state: Yup.string().trim().required('at least select one')
          })
        }
        onSubmit={(data) => {
          console.log(data)
```

```
            }}
        >
            {({
                errors
            }) => (
                <Form>
                    <Field as="select" name="state">
                        <option>Select state</option>
                        <option value="MH">MH</option>
                        <option value="AP">AP</option>
                        <option value="TN">TN</option>
                    </Field>
                    <br></br>
                    <ErrorMessage
                        id="firstName-error"
                        name="state"
                        component="div"
                        className="invalid-data"
                    />
                    <input type="submit" value="Add" style={{ background: 'green', border: 'none', padding: '8px',
marginTop: '15px' }} />
                </Form>
            )}
        </Formik>
    </>
  )
}
```

Working with Radio with Formik

```
import { ErrorMessage, Field, Form, Formik } from "formik";
```

```
import React from "react";
import * as Yup from "yup";
export default function AddUser() {

  return (
    <>
      <h1>Validation with Formik</h1>
      <Formik
        initialValues={{
          gender: ''
        }}
        validationSchema={
          Yup.object().shape({
            gender: Yup.string().trim().required('at least select one')
          })
        }
        onSubmit={(data) => {
          console.log(data)
        }}
      >
        {({
          errors
        }) => (
          <Form>
            <Field type="radio" name="gender" value="Male" />Male
            <Field type="radio" name="gender" value="Female" />FeMale
            <br></br>
            <ErrorMessage
              id="firstName-error"
              name="gender"
              component="div"
              className="invalid-data"
```

```
                />
                <input type="submit" value="Add" style={{ background: 'green', border: 'none', padding: '8px',
marginTop: '15px' }} />
            </Form>
          )}
        </Formik>
      </>
  )
}
```

Working with phone Number

To add phone input which supports different countries, use following npm:

```
npm i react-phone-input-2
```

```
import React from "react";
import PhoneInput from "react-phone-input-2";
export default function AddUser ()
{

  return (
    <>
      <form>
        <PhoneInput
          name="contact"
          country="us"
          onChange={( e ) => { console.log( e ) }}
        />
      </form>
    </>
  )
}
```

PhoneInput with India number

```
import React from "react";
import PhoneInput from "react-phone-input-2";
export default function AddUser ()
{

  return (
    <>
      <form>
        <PhoneInput
          name="contact"
          country="in"
          onChange={( e ) => { console.log( e ) }}
        />
      </form>
    </>
  )
}
```

Phone number with Formik

```
import { ErrorMessage, Field, Form, Formik } from "formik";
import React from "react";
import PhoneInput from "react-phone-input-2";
import * as Yup from "yup";
export default function AddUser ()
{

  return (
    <>
```

```
<h1>Validation with Formik</h1>
<Formik
    initialValues={{
        contact: ''
    }}
    validationSchema={
        Yup.object().shape( {
            contact: Yup.string().trim()
                .min( 11, 'minimum 11 digit required' )
                .max( 12, 'max 12 digits required' )
                .required( 'at least select one' )
        } )
    }
    onSubmit={( data ) =>
    {
        console.log( data )
    }}
>
    {( {
        errors,
        setFieldValue
    } ) => (
        <Form>
            <PhoneInput
                id="contact-number"
                name="contact"
                country={"us"}
                onChange={( e ) =>
                {
                    setFieldValue( 'contact', e )
                }}
```

```
                />
                <br></br>
                <ErrorMessage
                    id="firstName-error"
                    name="contact"
                    component="div"
                    className="invalid-data"
                />
                <input type="submit" value="Add" style={{ background: 'green', border: 'none', padding: '8px',
marginTop: '15px' }} />
            </Form>
        )}
    </Formik>
  </>
 )
}
```

Formik with regex:

```
import { ErrorMessage, Field, Form, Formik } from "formik";
import React from "react";
import * as Yup from "yup";
export default function AddUser ()
{

 return (
  <>
    <h1>Validation with Formik</h1>
    <Formik
      initialValues={{
        firstName: ''
```

```
        }}
        validationSchema={
            Yup.object().shape( {
                firstName: Yup.string().trim()
                    .matches( /^[A-Za-z ]*$/, "special character not allowed" )
                    .required( 'at least select one' )
            } )
        }
        onSubmit={( data ) =>
        {
            console.log( data )
        }}
    >
        {( {
            errors,
            setFieldValue,
            handleChange
        } ) => (
            <Form>
                <Field name="firstName" />
                <br></br>
                <ErrorMessage
                    id="firstName-error"
                    name="firstName"
                    component="div"
                    className="invalid-data"
                />
                <input type="submit" value="Add" style={{ background: 'green', border: 'none', padding: '8px',
marginTop: '15px' }} />
            </Form>
        )}
    </Formik>
```

```
    </>
  )
}
```

Formik with update operation:

```
import { ErrorMessage, Field, Form, Formik } from "formik";
import React from "react";
import * as Yup from "yup";
export default function AddUser ()
{
  const [ user, setUser ] = React.useState( {
    'id': 1,
    'firstName': 'Riya',
    'lastName': 'Anuse',
    'age': 19,
    'contact': '+919090871671'
  } )

  return (
    <>
      <h1>Validation with Formik</h1>
      <Formik
        initialValues={{
          id: user.id,
          firstName: user.firstName,
          lastName: user.lastName,
          age: user.age

        }}
        validationSchema={
          Yup.object().shape( {
```

```
        id: Yup.number().required( 'id required' ),
        firstName: Yup.string().trim()
            .min( 2, 'minimum firstName is 2' )
            .required( 'first name required' ),
        lastName: Yup.string().trim()
            .min( 2, 'minimum lastName is 2' )
            .required( 'last name required' )
    } )
}
onSubmit={( data ) =>
{
    console.log( data )
}}>
{( {
    errors,
} ) => (
    <Form>
        <Field name="id" placeholder="add id here" />
        <br></br>
        <ErrorMessage
            id="id-error"
            name="id"
            component="div"
            className="invalid-data"
        />
        <br></br>
        <Field name="firstName" placeholder="add firstName here" />
        <br></br>
        <ErrorMessage
            id="firstName-error"
            name="firstName"
            component="div"
```

```
                    className="invalid-data"
                />
                <br></br>
                <Field name="lastName" placeholder="add lastName here" />
                <br></br>
                <ErrorMessage
                    id="lastName-error"
                    name="lastName"
                    component="div"
                    className="invalid-data"
                />
                <br></br>
                <Field name="age" placeholder="add age here" />
                <br></br>
                <ErrorMessage
                    id="age-error"
                    name="age"
                    component="div"
                    className="invalid-data"
                />
                <br></br>

                <input type="submit" value="Add" style={{ background: 'green', border: 'none', padding: '8px',
marginTop: '15px' }} />
            </Form>
        )}
    </Formik>
  </>
 )
}
```

With Phone Input

```
import { ErrorMessage, Field, Form, Formik } from "formik";
import React from "react";
import PhoneInput from "react-phone-input-2";
import * as Yup from "yup";
export default function AddUser ()
{
  const [ user, setUser ] = React.useState( {
    'id': 1,
    'firstName': 'Riya',
    'lastName': 'Anuse',
    'age': 19,
    'contact': '+919090871671'
  } )

  return (
    <>
      <h1>Validation with Formik</h1>
      <Formik
        initialValues={{
          id: user.id,
          firstName: user.firstName,
          lastName: user.lastName,
          age: user.age,
          contact: user.contact


        }}
        validationSchema={
          Yup.object().shape( {
            id: Yup.number().required( 'id required' ),
            firstName: Yup.string().trim()
              .min( 2, 'minimum firstName is 2' )
              .required( 'first name required' ),
```

```
        lastName: Yup.string().trim()
            .min( 2, 'minimum lastName is 2' )
            .required( 'last name required' ),
        contact: Yup.string().trim()
            .min( 11, 'minimum contact is 11' )
            .max( 13, 'max contact is 12' )
            .required( 'last name required' )
    } )
}
onSubmit={( data ) =>
{
    console.log( data )
}}>
{( {
    errors,
    setFieldValue,
    values
} ) => (
    <Form>
        <Field name="id" placeholder="add id here" />
        <br></br>
        <ErrorMessage
            id="id-error"
            name="id"
            component="div"
            className="invalid-data"
        />
        <br></br>
        <Field name="firstName" placeholder="add firstName here" />
        <br></br>
        <ErrorMessage
            id="firstName-error"
```

```
                    name="firstName"
                    component="div"
                    className="invalid-data"
                />
                <br></br>
                <Field name="lastName" placeholder="add lastName here" />
                <br></br>
                <ErrorMessage
                    id="lastName-error"
                    name="lastName"
                    component="div"
                    className="invalid-data"
                />
                <br></br>
                <Field name="age" placeholder="add age here" />
                <br></br>
                <ErrorMessage
                    id="age-error"
                    name="age"
                    component="div"
                    className="invalid-data"
                />
                <br></br>
                <PhoneInput specialLabel="" country="in" name="contact" value={values.contact}
placeholder="add contact here"
                    onChange={( e ) => { setFieldValue( 'contact', e ) }}
                />
                <br></br>
                <ErrorMessage
                    id="contact-error"
                    name="contact"
                    component="div"
```

```
                    className="invalid-data"
                />
                <input type="submit" value="Add" style={{ background: 'green', border: 'none', padding: '8px',
marginTop: '15px' }} />
            </Form>
        )}
    </Formik>
  </>
 )
}
```

Redux

Redux is an open-source JavaScript library used to manage application state. React uses Redux for building the user interface. It was first introduced by Dan Abramov and Andrew Clark in 2015.
React Redux is the official React binding for Redux. It allows React components to read data from a Redux Store, and dispatch Actions to the Store to update data. Redux helps apps to scale by providing a sensible way to manage state through a unidirectional data flow model. React Redux is conceptually simple. It subscribes to the Redux store, checks to see if the data which your component wants have changed, and re-renders your component.

Steps to create redux setup:

1. **Create a reducers** folder under src folder.
2. Create cartReducer.js file under reducers folder

```
const initialState = {
   numOfItems: 0,
};

const cartReducer = ( state = initialState, action ) => {
   switch ( action.type ) {
       case "ADD_ITEM":
           return {
               ...state,
               numOfItems: state.numOfItems + 1,
           }
       case "DELETE_ITEM":
```

```
            return {
                ...state,
                numOfItems: state.numOfItems - 1,
            }
        default:
            return state;
    }
};
export default cartReducer;
```

3. Create a store.js file under the src folder.

```
import { createStore } from "redux";
import cartReducer from "./reducers/cartReducer";

const store = createStore( cartReducer );

export default store;
```

4. Add store in App.js file

```
import logo from './logo.svg';
import './App.css';
import { Provider } from 'react-redux';
import store from './store';
import HelloWorld from './HelloWorld';

function App () {
 return (
   <Provider store={store}>
     <div className="App">
       <HelloWorld />
     </div>
   </Provider>
 );
}

export default App;
```

5. Use reducer in HelloWorld.js file in useSelector

```
import React from "react";
import { useSelector } from "react-redux";

export default function HelloWorld () {
   const state = useSelector( ( state ) => state );
   return (
       <h1>Hello {state.numOfItems}</h1>
   )
}
```

How can we modify store data?

To modify a reducer we have to compulsory use actions.

Create cartAction.js in the above project.

```
const addItem = () => {
   return {
       type: "ADD_ITEM",
   };
};

const deleteItem = () => {
   return {
       type: "DELETE_ITEM",
   };
};

export { addItem, deleteItem };
```

```
import React from "react";
import { useDispatch, useSelector } from "react-redux";
import { addItem, deleteItem } from "./actions/cartAction";

export default function HelloWorld () {
   const state = useSelector( ( state ) => state );
   const dispatch = useDispatch();
   console.log( state )
   const add = () => {
       dispatch( addItem() );
   }
   const remove = () => {
```

```
        dispatch( deleteItem() )
    }
    return (
        <>
            <h1>Hello {state.numOfItems}</h1>
            <button onClick={add}>add item</button>
            <button onClick={remove}>remove item</button>
        </>
    )
}
```

React memo:
React.memo is a higher-order component in React that is used for memoizing functional components. Memoization is a technique to optimize rendering performance by preventing unnecessary renders of a component when its props haven't changed. React.memo compares the previous and current props and only re-renders the component if there are differences.

```
import React from "react"
import Welcome from "./welcome";
const Hello = () => {
    const [ count, setCount ] = React.useState( 0 );
    return (
        <>
            <Welcome />
            <h1>Count value : {count}</h1>
            <button onClick={() => { setCount( count
+ 1 ) }}>Increment</button>
        </>
    )
}

export default Hello;

import React, { useEffect } from "react"
const Welcome = () => {
    console.log( 'welcome component rerender' );
    return (
        <h1>Welcome component</h1>
    )
```

Here if we observe, whenever we are clicking on increment button, Welcome component is also re rendering.

But it should not happen because there is no change in the state of the Welcome component. For this we have to use memo as given in next example.

```
}

export default Welcome;
```

Example 2:

```
import React from "react"
import Welcome from "./welcome";
const Hello = () => {
   const [ count, setCount ] = React.useState( 0 );
   return (
       <>
           <Welcome />
           <h1>Count value : {count}</h1>
           <button onClick={() => { setCount( count
+ 1 ) }}>Increment</button>
       </>
   )
}

export default Hello;
```

```
import React, { memo, useEffect } from "react"
const Welcome = () => {
   console.log( 'welcome component rerender' );
   return (
       <h1>Welcome component</h1>
   )
}

export default memo( Welcome );
```

If we change state of Welcome component then only it will re render Welcome component

React Hooks:

useContext

Suppose I want to pass value from to child, we have to pass as props as given below:

```
import React from "react"
import Welcome, { Component1 } from "./welcome";
const Hello = () => {
   return (
       <Component1 />
   )
}

export default Hello;
```

```
import { useState } from "react";

export function Component1 () {
   const [ user, setUser ] = useState( "Riya" );

   return (
       <>
           <h1>{`Hello ${user}!`}</h1>
           <Component2 user={user} />
```

Here, if we observe we are passing the user through every child component which is repeating.

We can we use useContext hook as given below example:

```
        </>
    );
}

function Component2 ( { user } ) {
    return (
        <>
            <h1>Component 2</h1>
            <Component3 user={user} />
        </>
    );
}

function Component3 ( { user } ) {
    return (
        <>
            <h1>Component 3</h1>
            <h2>{`Hello ${user} again!`}</h2>
        </>
    );
}
```

Example 2:

```
import React from "react"
import Welcome, { Component1 } from "./welcome";
const Hello = () => {
    return (
        <Component1 />
    )
}

export default Hello;
```

```
import { createContext, useContext, useState } from
"react";
export const UserContext = createContext();
export function Component1 () {

    const [ user, setUser ] = useState( "Riya" );

    return (
        <UserContext.Provider value={user}>
            <h1>{`Hello ${user}!`}</h1>
            <Component2 />
        </UserContext.Provider>
    );
```

```
}

function Component2 () {
    return (
        <>
            <h1>Component 2</h1>
            <Component3 />
        </>
    );
}

export function Component3 () {
    const user = useContext( UserContext );
    return (
        <>
            <h1>Component 3</h1>
            <h2>{`Hello ${user} again!`}</h2>
        </>
    );
}
```

Key:

In React, a "key" is a special string attribute that you need to include when creating lists of elements. The key prop helps React identify which items have changed, been added, or been removed in a list, and it improves the performance and efficiency of updating the user interface.

When you render a list of elements in React, each item in the list should have a unique key prop. This allows React to efficiently update and re-render only the components that have changed. The key should be stable and not change over time unless the identity of the item changes.

React uses keys during the process of reconciliation, which is the algorithm used to update the DOM efficiently. Keys help React identify which items have changed, been added, or been removed.

```
import React from 'react';

const MyList = () => {
  const items = [
    { id: 1, name: 'Item 1' },
```

```
    { id: 2, name: 'Item 2' },
    { id: 3, name: 'Item 3' },
  ];

  return (
    <ul>
      {items.map(item => (
        <li key={item.id}>{item.name}</li>
      ))}
    </ul>
  );
};

export default MyList;
```

React Router:
npm install react-router-dom

App.js

```
import { Provider } from 'react-redux';
import './App.css';
import { Hello } from './routers/hello';
import store from './store';
import { BrowserRouter, Route, Link, Routes } from
'react-router-dom';
import { Hello1 } from './routers/hello1';
function App () {
 return (
   <Provider store={store}>
     <BrowserRouter>
       <Routes>
         <Route path="/" element={<Hello
/>}></Route>
         <Route path='/hello1' element={<Hello1
/>}></Route>
       </Routes>
```

Hello.js

```
import React from "react";
import { Link, useNavigate } from
"react-router-dom";

export const Hello = () => {

   return (
       <>
           <a href="/hello1">Go to Hello1</a> |
           <Link to="/hello1">Goto Hello1 with
Link</Link>
       </>
   )
}
```

Hello1.js

```
import React from "react";
```

```
      </BrowserRouter>
    </Provider>
  );
}

export default App;
```

```
export const Hello1 = () => {

    return (
        <h1>This is Hello1 component</h1>
    )
}
```

Here when we use a tag it refreshes the page whereas when we use Link tag it just routes to another component without refresh.

Example 2:

```
import React from "react";
import { Link, useNavigate } from
"react-router-dom";

export const Hello = () => {
    const navigate = useNavigate();
    const gotoHello1 = () => {
        navigate( '/hello1' )
    }
    return (
        <>
            <button onClick={gotoHello1}>Goto Hello1
component</button>
        </>
    )
}
```

Here we can route to another component programmatically.