Gas testing new form pdf

I'm not robot

reCAPTCHA

Continue

Software Engineer from Costa Rica, lives in BerlinFormik is a nice library to speed up the process of creating forms in React. It covers all basic functions such as form status, validation, and submission. First, create an empty create-react-app create-react-app my-appInstall the formik libraryUse formik componentsRemove the generated sample code and import the formik library and some components into App.js. We use the react useState hook only to display the values of the form in this example.import React, 'useState' from 'react' import' Formik, Form, Field ' from 'formik' function App() ' const [result, setResult] = useState('') return (* Formik is the main component that processes all the logic ' ' ' ' ' ' & lt;form> Wrapper <Formik> s() => (<Form></Form> • Export standard AppLet components with the Field formik component to create some fields. Check the documents to see all sorts of props, default field an input type=Text <Field name=fieldName></Field> For the selection we specify the options as child elements. <Field as=select name=color value=none><option value=none>Choose a color<option value=red>Red<option value=green>Green<option value=blue>Blue </Field>In this example, we use a range of text to display the values. after the props of the <Field as=textarea value='result's rows={5}></Field> form that have been propred to child elements that provide many options for custom fields.<Field name=email> => (<div><label>E-Mail :</label><input type=email required= placeholder=Email. . . . . field'=></div> We </Field>can go further and simply set a component support. Form props are attached to this component function CustomInput(' Field, Form, ... Props - Returns <input . . . . . field_= . . . . . props-=> A <Field name=name required= placeholder=Name component='CustomInput'></Field> typ=submit automatically handles form on Submit<button type=submit>Submit<Button>All TogetherThis is the last sample form created with formik. Formik needs the initialValues prop to function properly.import React, -useState - from 'react' import - Formik, Form, Field ' of 'formik' function CustomInput(' Field, Form, ... Props - Returns <input . . . . . field_= . . . . . props-=> Function App() - const [result, setResult] = useState('') return (<Formik initialvalues=' email:= ",= name:= ",,=color:= }}= onsubmit={(values, actions)=> - setResult(JSON.stringify(werte)) > {() => (<Form><Field as=select name=color value=none><option value=none>Wählen Sie eine Farbe<option value=red>Rot<option value=green>Grün<option value=blue>Blau</Field>
<Field name=email> => (<div><label>E-Mail:</label><input type=email required type=email required=></input type=email required ></div></Field></Form></Formik></form> ></div></Field></Form></Formik></form> {... Field - />)
<Field name=name required= placeholder=Name component='CustomInput'></Field>
<button type=submit>Submit&lt ;/button>

<Field style='width:= '100%"= as=textarea value='result rows={2}></Field>)) - Export Standard AppNot so failed, but I didn't want to pollute the code with styling. Unit tests with React Testing LibraryNormally, it is sufficient to use Jest and enzymes to test components, but with Formik, internal react state and events under the hood become more complex and a simple input.simulate ('change') does not work. Glücklicherweise gibt es eine nette Software namens Testing Library, die Unterstützung für viele Frontend-Bibliotheken und Frameworks von allen hat, Installieren Sie die Bibliothek als dev dependencyyarn add --dev '@testing-library/react'Test, wenn die App nicht zuerst abstürzt('rendert ohne Abstürze', () => 'const div = document.createElement('div') ReactDOM.render(<App></App> , div) () => ,,const container = render(<App></App>) const name = container.querySelector('input[name=name]') const email = container.querySelector('input[name=email]') const color = container.querySelector('input[name=color]') const submit = container.querySelector('button[type=email]') const color = container.querySelector('input]') const color = container.querySelector('input[name=color]') const submit = container.querySelector('button • Ziel: 'mockname' - fireEvent.change(email, ' target: 'mockemail' ' ' ' '