import PropTypes from 'prop-types';

class Greeting extends React.Component {

render() {

return (

<h1>Привет, {this.props.name}</h1>

);

}

}

Greeting.propTypes = {

name: PropTypes.string

};

import PropTypes from 'prop-types';

MyComponent.propTypes = {

// Можно объявить проп на соответствие определённому JS-типу.

// По умолчанию это не обязательно.

optionalArray: PropTypes.array,

optionalBool: PropTypes.bool,

optionalFunc: PropTypes.func,

optionalNumber: PropTypes.number,

optionalObject: PropTypes.object,

optionalString: PropTypes.string,

optionalSymbol: PropTypes.symbol,

// Все, что может быть отрендерено:

// числа, строки, элементы или массивы

// (или фрагменты) содержащие эти типы

optionalNode: PropTypes.node,

// React-элемент

optionalElement: PropTypes.element,

// Тип React-элемент (например, MyComponent).

optionalElementType: PropTypes.elementType,

// Можно указать, что проп должен быть экземпляром класса

// Для этого используется JS-оператор instanceof.

optionalMessage: PropTypes.instanceOf(Message),

// Вы можете задать ограничение конкретными значениями

// при помощи перечисления

optionalEnum: PropTypes.oneOf(['News', 'Photos']),

// Объект, одного из нескольких типов

optionalUnion: PropTypes.oneOfType([

PropTypes.string,

PropTypes.number,

PropTypes.instanceOf(Message)

]),

// Массив объектов конкретного типа

optionalArrayOf: PropTypes.arrayOf(PropTypes.number),

// Объект со свойствами конкретного типа

optionalObjectOf: PropTypes.objectOf(PropTypes.number),

// Объект с определённой структурой

optionalObjectWithShape: PropTypes.shape({

color: PropTypes.string,

fontSize: PropTypes.number

}),

// При наличии необъявленных свойств в объекте будут вызваны предупреждения

optionalObjectWithStrictShape: PropTypes.exact({

name: PropTypes.string,

quantity: PropTypes.number

}),

// Можно добавить`isRequired` к любому приведённому выше типу,

// чтобы показывать предупреждение,

// если проп не передан

requiredFunc: PropTypes.func.isRequired,

// Обязательное значение любого типа

requiredAny: PropTypes.any.isRequired,

// Можно добавить собственный валидатор.

// Он должен возвращать объект `Error` при ошибке валидации.

// Не используйте `console.warn` или `throw`

// - это не будет работать внутри `oneOfType`

customProp: function(props, propName, componentName) {

if (!/matchme/.test(props[propName])) {

return new Error(

'Проп `' + propName + '` компонента' +

' `' + componentName + '` имеет неправильное значение'

);

}

},

// Можно задать свой валидатор для `arrayOf` и `objectOf`.

// Он должен возвращать объект Error при ошибке валидации.

// Валидатор будет вызван для каждого элемента в массиве

// или для каждого свойства объекта.

// Первые два параметра валидатора

// - это массив или объект и ключ текущего элемента

customArrayProp: PropTypes.arrayOf(function(propValue, key, componentName, location, propFullName) {

if (!/matchme/.test(propValue[key])) {

return new Error(

'Проп `' + propFullName + '` компонента' +

' `' + componentName + '` имеет неправильное значение'

);

}

})

};

<Router>

<div>

<nav>

<ul>

<li>

<Link to="/">Home</Link>

</li>

<li>

<Link to="/about">About</Link>

</li>

<li>

<Link to="/users">Users</Link>

</li>

</ul>

</nav>

{/\* A <Switch> looks through its children <Route>s and

renders the first one that matches the current URL. \*/}

<Switch>

<Route path="/about">

<About />

</Route>

<Route path="/users">

<Users />

</Route>

<Route path="/">

<Home />

</Route>

</Switch>

</div>

</Router>

let match = useRouteMatch();

<Route path={`${match.path}/:topicId`}>

<Link to={`${match.url}/components`}>Components</Link>

import React from 'react'

import { useFormik } from 'formik'

export function RegisterForm () {

// A custom validation function. This must return an object

// which keys are symmetrical to our values/initialValues

const validate = values => {

const errors = {}

if (!values.email) {

errors.email = 'Required'

} else if (values.email.length < 4) {

errors.email = 'Must be 5 characters or more'

}

if (!values.password) {

errors.password = 'Required'

} else if (values.password.length < 8) {

errors.password = 'Must be 8 characters or more'

} else if (values.password === '12345678') {

errors.password = 'Must not be 12345678 !!!'

}

if (!values.repassword) {

errors.repassword = 'Required'

} else if (values.repassword !== values.password) {

errors.repassword = 'Second password doesn\'t match'

}

return errors

}

const formik = useFormik({

initialValues: {

email: '',

password: '',

repassword: ''

},

validate,

onSubmit: values => {

alert(JSON.stringify(values, null, 2))

}

})

return (<div>

<h1>Register </h1>

<form onSubmit={formik.handleSubmit}>

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

<input

id="email"

name="email"

type="email"

onChange={formik.handleChange}

onBlur={formik.handleBlur}

value={formik.values.email}/>

{formik.touched.email && formik.errors.email ? <div className='error'>{formik.errors.email}</div> : null}

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

<input

id="password"

name="password"

type="password"

onChange={formik.handleChange}

onBlur={formik.handleBlur}

value={formik.values.password}/>

{formik.touched.password && formik.errors.password ? <div className='error'>{formik.errors.password}</div> : null}

<label htmlFor="repassword">Password again</label>

<input

id="repassword"

name="repassword"

type="password"

onChange={formik.handleChange}

onBlur={formik.handleBlur}

value={formik.values.repassword}/>

{formik.touched.repassword && formik.errors.repassword ? <div className='error'>{formik.errors.repassword}</div> : null}

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

</form>

</div>)

}

JSS is an authoring tool for CSS which allows you to use JavaScript to describe styles in a declarative, conflict-free and reusable way. It can compile in the browser, server-side or at build time in Node.

const Button = styled.button`

background: transparent;

border-radius: 3px;

border: 2px solid palevioletred;

color: palevioletred;

margin: 0.5em 1em;

padding: 0.25em 1em;

${props => props.primary && css`

background: palevioletred;

color: white;

`}

`;

const Container = styled.div`

text-align: center;

`

render(

<Container>

<Button>Normal Button</Button>

<Button primary>Primary Button</Button>

</Container>

);