## Page:1

# Formik and Yup

### What is Formik?

Formik is a popular open-source library for managing forms in React. It helps simplify form handling by providing:

* **State Management**: Manages form state (values, errors, touched fields).
* **Validation**: Integrates easily with validation libraries like Yup.
* **Submission Handling**: Simplifies form submission processes.
* **Field Management**: Provides hooks and components to manage form fields and their states.

### What is Yup?

Yup is a JavaScript schema builder for value parsing and validation. It helps in:

* **Schema Definition**: Defining schemas for data validation.
* **Validation Rules**: Creating complex validation rules for various data types.
* **Error Messages**: Customizing validation error messages.
* **Integration**: Integrates seamlessly with Formik for form validation.

## Code Breakdown

# Installation

To get started with Formik and Yup, you need to install them in your React project.

## **Formik Installation**

npm install formic

## **Yup Installation**

npm install yup

## Initial Setup

1. **Import Libraries**

javascriptCopy code

import React from "react";

import { useFormik } from 'formik';

import \* as Yup from 'yup';

## Page:2

## Formik Setup

**Initial Values**

Define the initial values for the form fields.

javascriptCopy code

const initialValues = {

title: "",

description: "",

price: 0,

image: "",

category: ""

};

**Formik Configuration**

Setup Formik using the useFormik hook.

javascriptCopy code

const { values, handleSubmit, handleChange, errors } = useFormik({

initialValues: initialValues,

validationSchema: ProductSchema,

onSubmit: (value, action) => {

console.log(value);

action.resetForm();

}

});

const { title, description, price, image, category } = values;

### Yup Schema

1. **Validation Schema** Define the validation schema using Yup.

javascriptCopy code

const ProductSchema = Yup.object({

title: Yup.string().required("📝 Title is required"),

description: Yup.string().required("📝 Description is required"),

## Page:3

price: Yup.number().required("💵 Price is required"),

image: Yup.string().required("📷 Image is required"),

category: Yup.string().required("📂 Category is required")

});

### Form Component

1. **Form Structure** Create the form component with fields and error messages.

javascriptCopy code

const Form = () => {

return (

<form onSubmit={handleSubmit}>

<input

type="text"

onChange={handleChange}

name="image"

value={image}

placeholder="📷 Image"

/>

{errors.image && <div style={{ color: "red" }}>{errors.image}</div>}

<br />

<input

type="text"

onChange={handleChange}

name="title"

value={title}

placeholder="📝 Title"

/>

{errors.title && <div style={{ color: "red" }}>{errors.title}</div>}

<br />

## Page:4

<select name="category" value={category} onChange={handleChange}>

<option value="">📂 Select Your Category</option>

<option value="men's clothing">👔 Men's Clothing</option>

<option value="jewelery">💍 Jewelry</option>

<option value="electronics">📱 Electronics</option>

<option value="women's clothing">👗 Women's Clothing</option>

</select>

{errors.category && <div style={{ color: "red" }}>{errors.category}</div>}

<br />

<input

name="price"

value={price}

onChange={handleChange}

type="number"

placeholder="💵 Price"

/>

{errors.price && <div style={{ color: "red" }}>{errors.price}</div>}

<br />

<input

name="description"

value={description}

onChange={handleChange}

type="text"

placeholder="📝 Description"

/>

{errors.description && <div style={{ color: "red" }}>{errors.description}</div>}

<br />

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

</form>

## Page:5

);

};

export default Form;

> For password

>

>

> const validationSchema = Yup.object({

> password: Yup.string()

> .min(8, "Password must be at least 8 characters")

> .required("Password is required"),

> confirmPassword: Yup.string()

> .oneOf([Yup.ref('password'), null], "Passwords must match")

> .required("Confirm Password is required")

> });

>