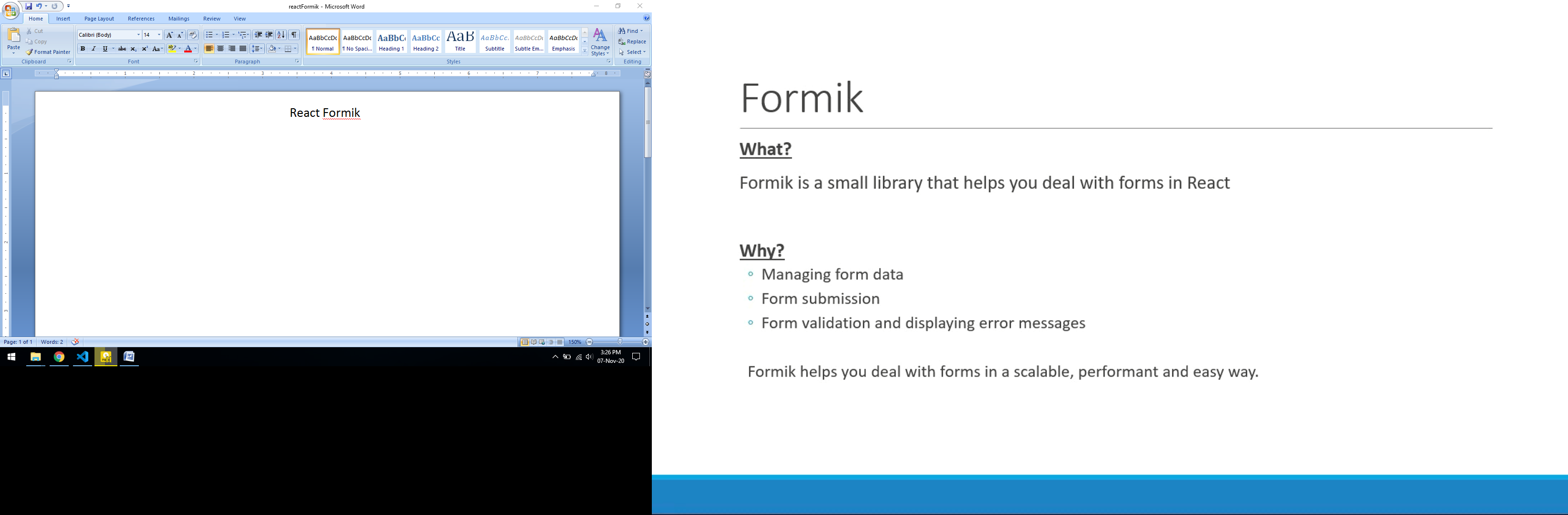
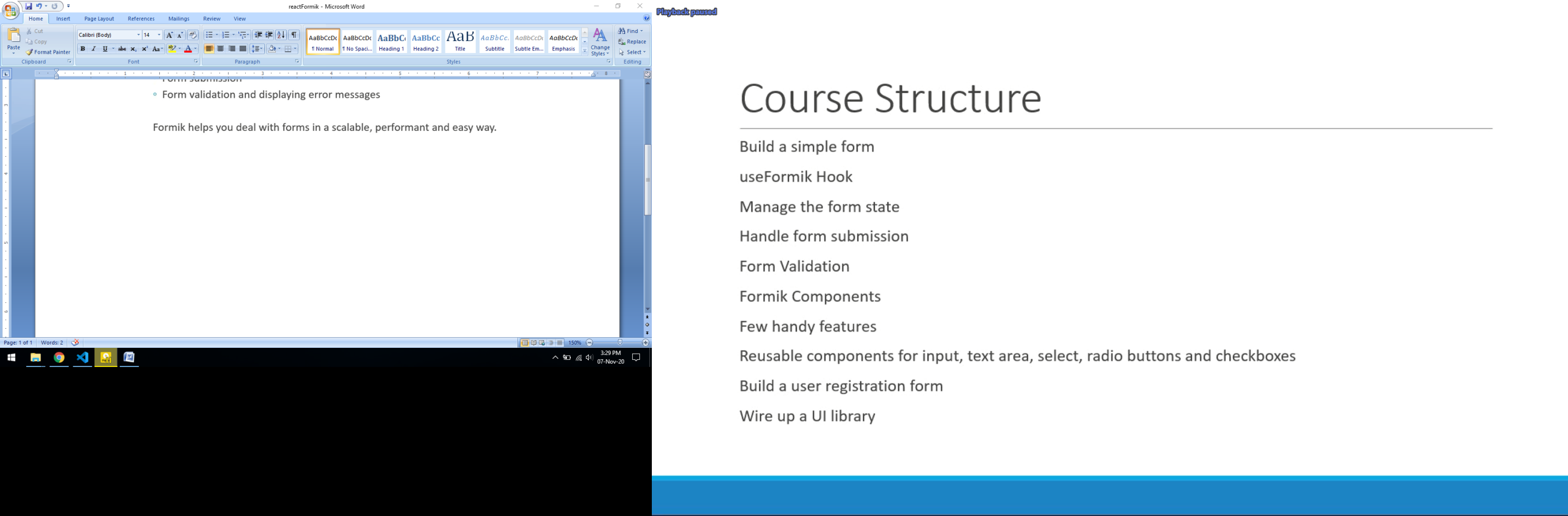
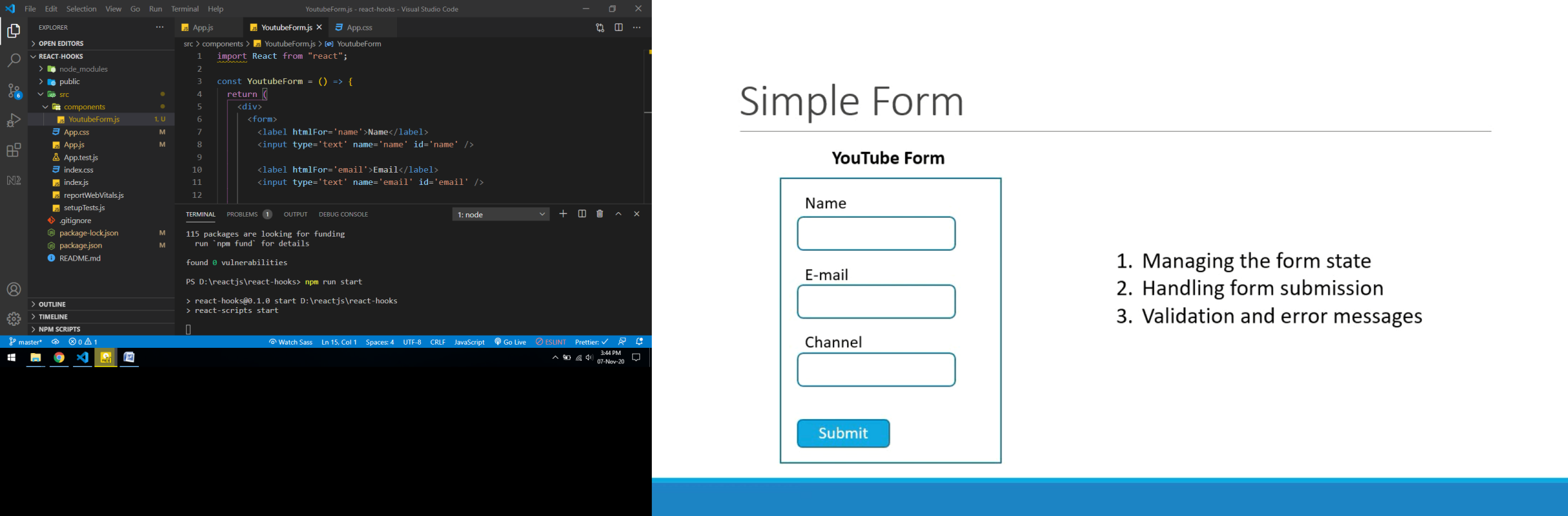
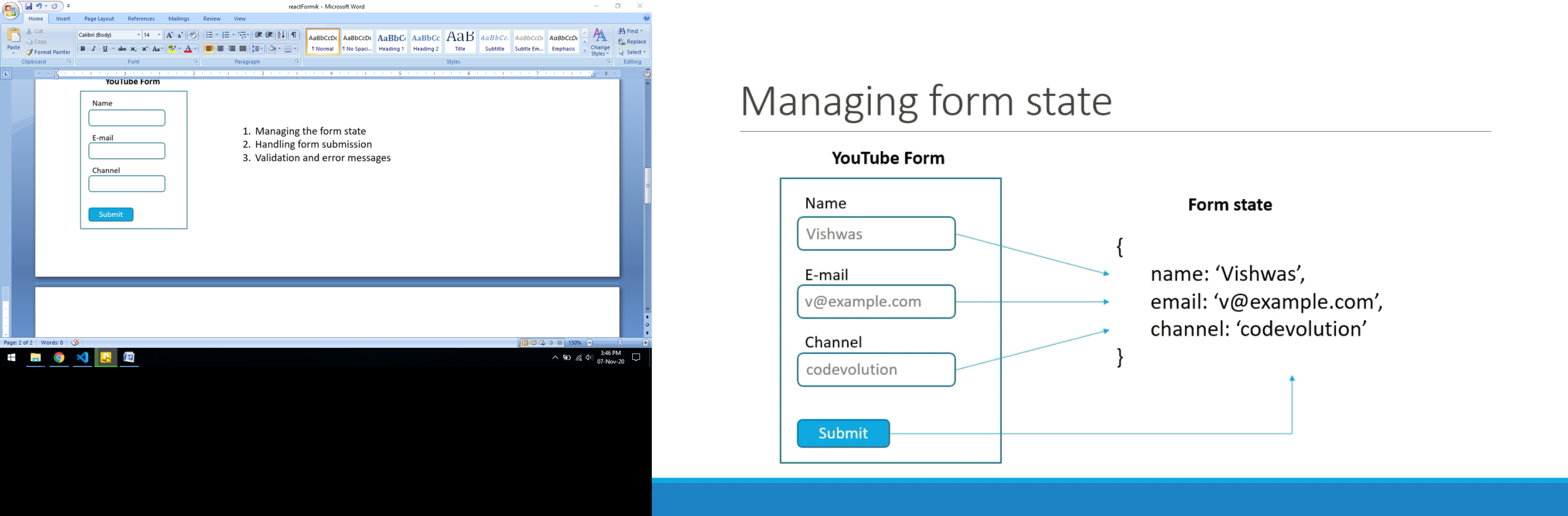
React Formik





Install formik - npm i formik





Manage Form State

import { useFormik } from "formik";

import React from "react";

const YoutubeForm = () => {

  const formik = useFormik({

    initialValues: {

      name: "", //field must be equal to input name attribute

      email: "",

      channel: "",

    },

  });

  console.log(formik.values);

  return (

    <div>

      <form>

        <label htmlFor='name'>Name</label>

        <input

          type='text'

          name='name'

          onChange={formik.handleChange}

          value={formik.values.name}

          id='name'

        />

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

        <input

          type='text'

          name='email'

          onChange={formik.handleChange}

          value={formik.values.email}

          id='email'

        />

        <label htmlFor='channel'>Channel</label>

        <input

          type='text'

          name='channel'

          onChange={formik.handleChange}

          value={formik.values.channel}

          id='channel'

        />

        <button>Submit</button>

      </form>

    </div>

  );

};

export default YoutubeForm;

Handle Form Submission

import { useFormik } from "formik";

import React from "react";

const YoutubeForm = () => {

  const formik = useFormik({

    initialValues: {

      name: "",

      email: "",

      channel: "",

    },

    onSubmit: (values) => {

      console.log("Form Data", values);

    },

  });

  return (

    <div>

      <form onSubmit={formik.handleSubmit}>

        <label htmlFor='name'>Name</label>

        <input

          type='text'

          name='name'

          onChange={formik.handleChange}

          value={formik.values.name}

          id='name'

        />

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

        <input

          type='text'

          name='email'

          onChange={formik.handleChange}

          value={formik.values.email}

          id='email'

        />

        <label htmlFor='channel'>Channel</label>

        <input

          type='text'

          name='channel'

          onChange={formik.handleChange}

          value={formik.values.channel}

          id='channel'

        />

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

      </form>

    </div>

  );

};

export default YoutubeForm;

Form Validation

import { useFormik } from "formik";

import React from "react";

const YoutubeForm = () => {

  const initialValues = {

    name: "",

    email: "",

    channel: "",

  };

  const onSubmit = (values) => {

    console.log("Form Data", values);

  };

  const validate = (values) => {

    let errors = {};

    //values.name ,values.email, values.channel

    //errors.name,errors.email,errors.channel

    if (!values.name) {

      errors.name = "Required";

    }

    if (!values.email) {

      errors.email = "Required";

    } else if (

      !/^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)\*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/.test(

        String(values.email).toLowerCase()

      )

    ) {

      errors.email = "Invalid Email Id";

    }

    if (!values.channel) {

      errors.channel = "Required";

    }

    return errors;

  };

  const formik = useFormik({

    initialValues,

    onSubmit,

    validate,

  });

  console.log("formik Data", formik.values);

  console.log("formik Errors", formik.errors);

  console.log("formik touched", formik.touched);

  return (

    <div>

      <form onSubmit={formik.handleSubmit}>

        <div className='form-control'>

          <label htmlFor='name'>Name</label>

          <input

            type='text'

            name='name'

            onChange={formik.handleChange}

            value={formik.values.name}

            id='name'

          />

          {formik.errors.name && (

            <div className='error'>{formik.errors.name}</div>

          )}

        </div>

        <div className='form-control'>

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

          <input

            type='text'

            name='email'

            onChange={formik.handleChange}

            value={formik.values.email}

            id='email'

          />

          {formik.errors.email && (

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

          )}

        </div>

        <div className='form-control'>

          <label htmlFor='channel'>Channel</label>

          <input

            type='text'

            name='channel'

            onChange={formik.handleChange}

            value={formik.values.channel}

            id='channel'

          />

          {formik.errors.channel && (

            <div className='error'>{formik.errors.channel}</div>

          )}

        </div>

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

      </form>

    </div>

  );

};

export default YoutubeForm;

Yup: validation package- npm i yup

import { useFormik } from "formik";

import React from "react";

import \* as Yup from "yup";

const YoutubeForm = () => {

  const initialValues = {

    name: "",

    email: "",

    channel: "",

  };

  const onSubmit = (values) => {

    console.log("Form Data", values);

  };

  const validationSchema = Yup.object({

    name: Yup.string().required("Name is required"),

    email: Yup.string()

      .required("Email is required")

      .email("Invalid E-mail Address"),

    channel: Yup.string().required("Channel is required"),

  });

  const formik = useFormik({

    initialValues,

    onSubmit,

    validationSchema,

  });

  return (

    <div>

      <form onSubmit={formik.handleSubmit}>

        <div className='form-control'>

          <label htmlFor='name'>Name</label>

          <input

            type='text'

            name='name'

            onChange={formik.handleChange}

            value={formik.values.name}

            id='name'

          />

          {formik.errors.name && (

            <div className='error'>{formik.errors.name}</div>

          )}

        </div>

        <div className='form-control'>

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

          <input

            type='text'

            name='email'

            onChange={formik.handleChange}

            value={formik.values.email}

            id='email'

          />

          {formik.errors.email && (

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

          )}

        </div>

        <div className='form-control'>

          <label htmlFor='channel'>Channel</label>

          <input

            type='text'

            name='channel'

            onChange={formik.handleChange}

            value={formik.values.channel}

            id='channel'

          />

          {formik.errors.channel && (

            <div className='error'>{formik.errors.channel}</div>

          )}

        </div>

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

      </form>

    </div>

  );

};

export default YoutubeForm;

Reduce code by formik

<input

            type='text'

            name='name'

            onChange={formik.handleChange}

            value={formik.values.name}

            id='name'

          />

Can replace with

 <input

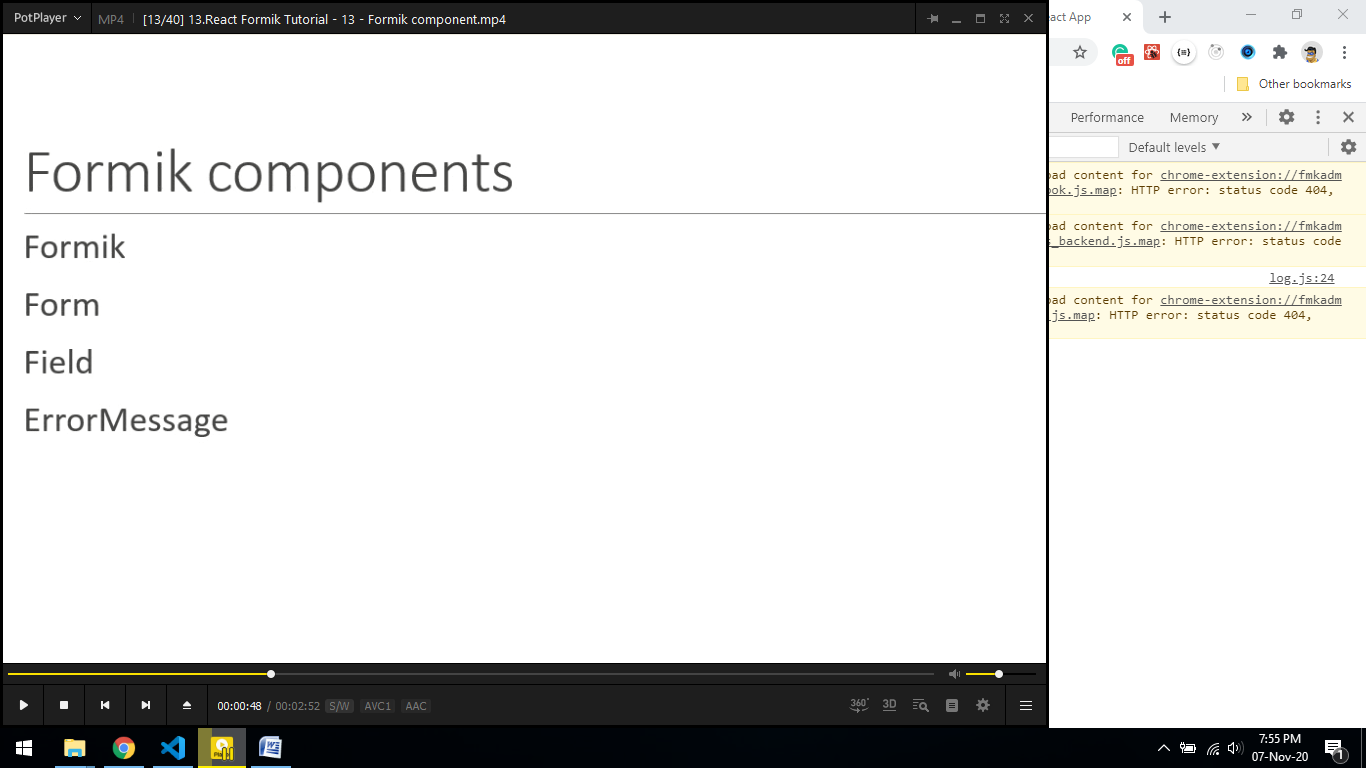
            type='text'

            name='name'

            {...formik.getFieldProps("name")}

            id='name'

          />



Form validation with Formik Component

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

import React from "react";

import \* as Yup from "yup";

const YoutubeForm = () => {

  const initialValues = {

    name: "",

    email: "",

    channel: "",

  };

  const onSubmit = (values) => {

    console.log("Form Data", values);

  };

  const validationSchema = Yup.object({

    name: Yup.string().required("Name is required"),

    email: Yup.string()

      .required("Email is required")

      .email("Invalid E-mail Address"),

    channel: Yup.string().required("Channel is required"),

  });

  return (

    <Formik

      initialValues={initialValues}

      onSubmit={onSubmit}

      validationSchema={validationSchema}

    >

      <Form>

        <div className='form-control'>

          <label htmlFor='name'>Name</label>

          <Field type='text' name='name' id='name' />

          <ErrorMessage name='name' />

        </div>

        <div className='form-control'>

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

          <Field type='text' name='email' id='email' />

          <ErrorMessage name='email' />

        </div>

        <div className='form-control'>

          <label htmlFor='channel'>Channel</label>

          <Field type='text' name='channel' id='channel' />

          <ErrorMessage name='channel' />

        </div>

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

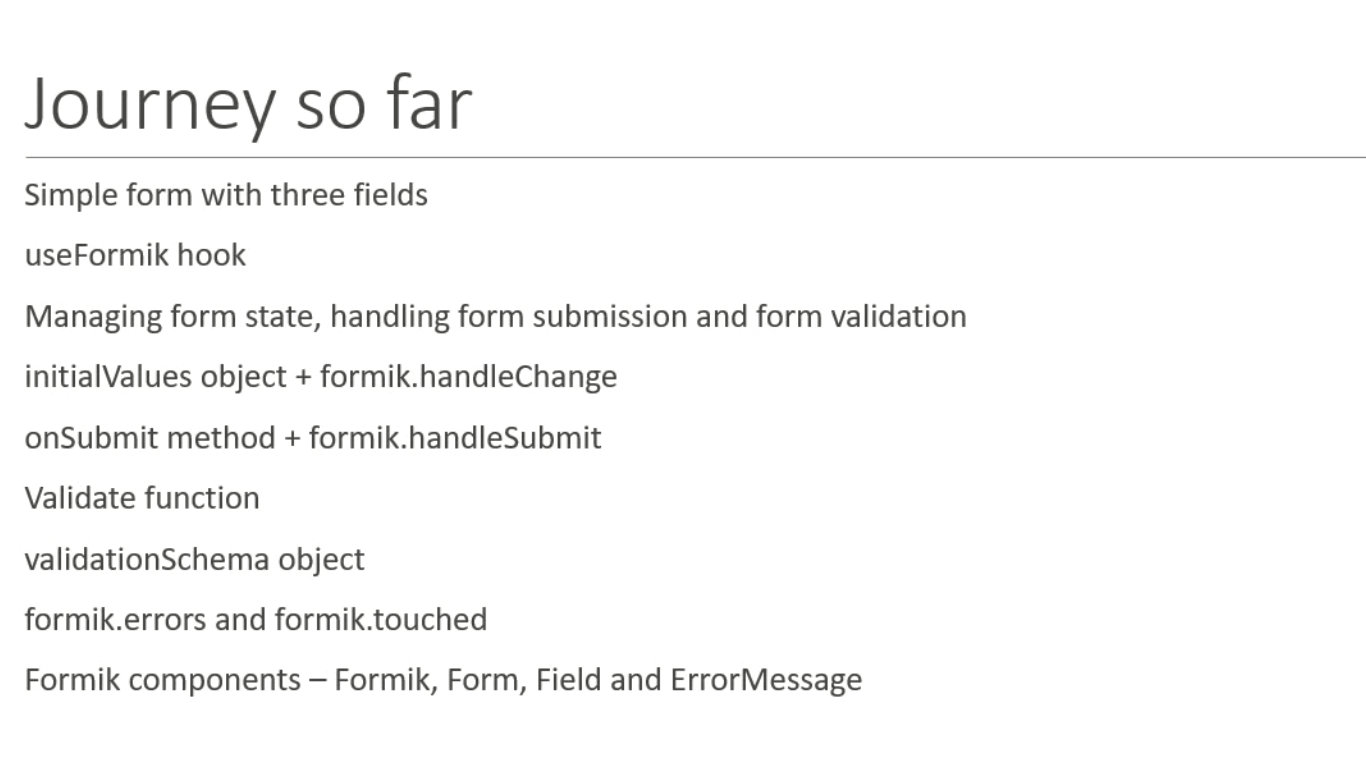
      </Form>

    </Formik>

  );

};

export default YoutubeForm;



 <Field

            as='textarea'

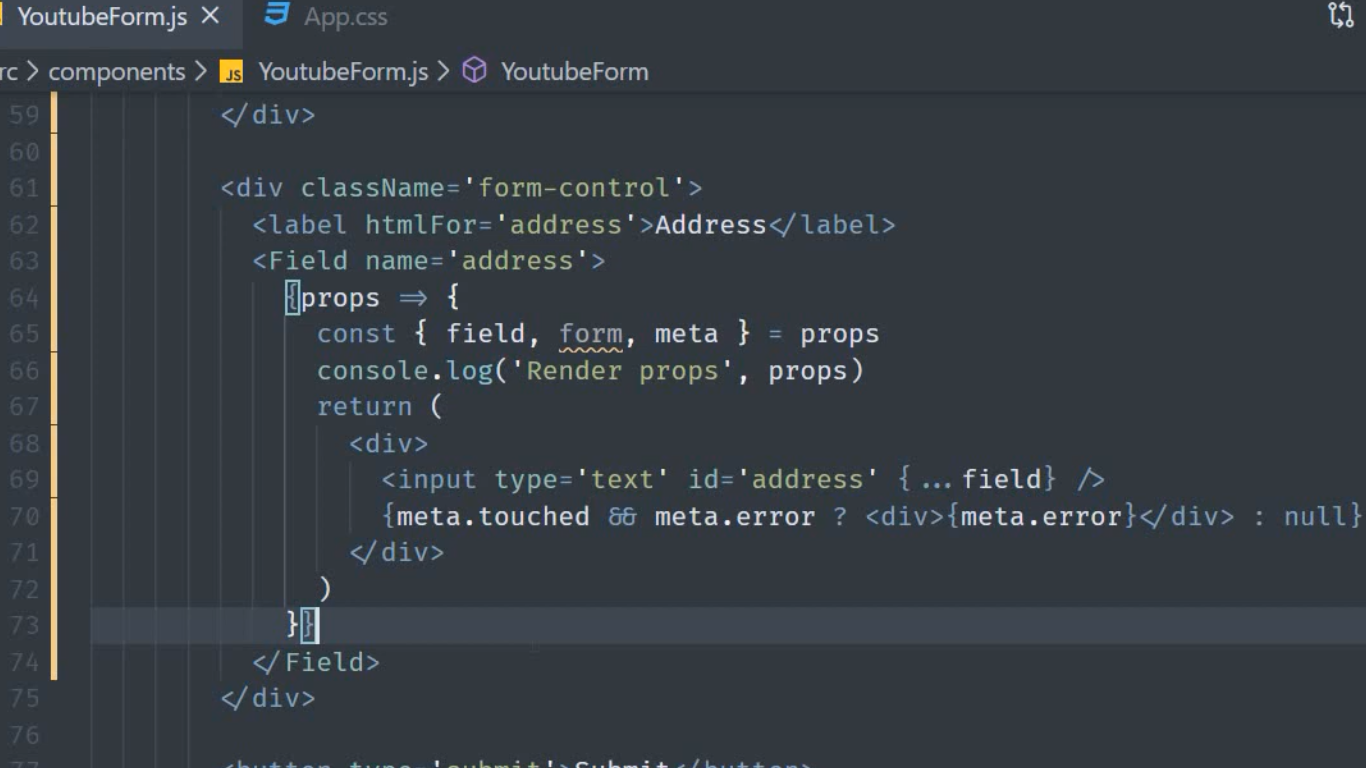
            placeholder='Enter you comment'

            name='comments'

            id='comments'

          />

Field Component



ErrorMessage

TextError.js

import React from "react";

const TextError = (props) => {

  return <div className='error'>{props.children}</div>;

};

export default TextError;

YoutubeForm.js

<div className='form-control'>

          <label htmlFor='name'>Name</label>

          <Field type='text' name='name' id='name' />

          <ErrorMessage name='name' component={TextError} />

        </div>

        <div className='form-control'>

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

          <Field type='text' name='email' id='email' />

          <ErrorMessage name='email'>

            {(errorMsg) => <div className='error'>{errorMsg}</div>}

          </ErrorMessage>

        </div>

Nested Object

const initialValues = {

    name: "",

    email: "",

    channel: "",

    comments: "",

    social: {

      facebook: "",

      twitter: "",

    },

  };

 <div className='form-control'>

          <label htmlFor='facebook'>Facebook Profile</label>

          <Field type='text' name='social.facebook' id='facebook' />

        </div>

        <div className='form-control'>

          <label htmlFor='twitter'>Twitter Profile</label>

          <Field type='text' name='social.twitter' id='twitter' />

        </div>

FieldArray

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

import React from "react";

import \* as Yup from "yup";

import TextError from "./TextError";

const YoutubeForm = () => {

  const initialValues = {

    name: "",

    email: "",

    channel: "",

    comments: "",

    social: {

      facebook: "",

      twitter: "",

    },

    phoneNumbers: ["", ""],

    phNumbers: [""],

  };

 <div className='form-control'>

          <label htmlFor='phone1'>Phone Number 1 static</label>

          <Field type='text' name='phoneNumbers[0]' id='phone1' />

        </div>

        <div className='form-control'>

          <label htmlFor='phone2'>Phone Number 2 static</label>

          <Field type='text' name='phoneNumbers[1]' id='phone2' />

        </div>

        <FieldArray name='phNumbers'>

          {(props) => {

            const { push, remove, form } = props;

            const { values } = form;

            const { phNumbers } = values;

            return (

              <div>

                {phNumbers.map((phNum, index) => (

                  <div key={index}>

                    <Field name={`phNumbers[${index}]`} />

                    {index > 0 && (

                      <button type='button' onClick={() => remove(index)}>

                        -

                      </button>

                    )}

                    <button type='button' onClick={() => push("")}>

                      +

                    </button>

                  </div>

                ))}

              </div>

            );

          }}

        </FieldArray>

FastField

FastField is an optimized version of field. It don’t allow multiple re render. But there is some conditions in FastField

<FastField /> is an optimized version of <Field /> meant to be used on large forms (~30+ fields) or when a field has very expensive validation requirements.

**When to use <FastField />**

**If a <Field /> is "independent" of all other <Field />'s in your form, then you can use <FastField />**.

<div className='form-control'>

          <label htmlFor='address'>Address</label>

          <Field name='address'>

            {(props) => {

              const { field, form, meta } = props;

              return (

                <div>

                  <textarea id='address' {...field}></textarea>

                  {meta.touched && meta.error ? <div>{meta.error}</div> : null}

                </div>

              );

            }}

          </Field>

        </div>

<div className='form-control'>

          <label htmlFor='address'>Address</label>

          <FastField name='address'>

            {(props) => {

              const { field, form, meta } = props;

              return (

                <div>

                  <textarea id='address' {...field}></textarea>

                  {meta.touched && meta.error ? <div>{meta.error}</div> : null}

                </div>

              );

            }}

          </FastField>

        </div>

When validation occur

1. On change
2. On Blur
3. On submit

 <Formik

      initialValues={initialValues}

      onSubmit={onSubmit}

      validationSchema={validationSchema}

      validateOnBlur={false}

      validateOnChange={false}

    >

      <Form>

<div className='form-control'>

          <label htmlFor='name'>Name</label>

          <Field type='text' name='name' id='name' />

          <ErrorMessage name='name' component={TextError} />

        </div>

<button type='submit'>Submit</button>

      </Form>

    </Formik>

Validtion on field level

const validateComment = (value) => {

    let error;

    if (!value) {

      error = "Comment is required";

    }

    return error;

  };

<div className='form-control'>

          <label htmlFor='comments'>Comments</label>

          <Field

            as='textarea'

            placeholder='Enter you comment'

            name='comments'

            id='comments'

            validate={validateComment}

          />

          <ErrorMessage name='comments' component={TextError} />

        </div>

Manual Form validation triggering

import {

  ErrorMessage,

  Field,

  Form,

  Formik,

  FieldArray,

  FastField,

} from "formik";

import React from "react";

import \* as Yup from "yup";

import TextError from "./TextError";

const YoutubeForm = () => {

  const initialValues = {

    name: "",

    email: "",

    channel: "",

    comments: "",

    address: "",

    social: {

      facebook: "",

      twitter: "",

    },

    phoneNumbers: ["", ""],

    phNumbers: [""],

  };

  const onSubmit = (values) => {

    console.log("Form Data", values);

  };

  const validationSchema = Yup.object({

    name: Yup.string().required("Name is required"),

    email: Yup.string()

      .required("Email is required")

      .email("Invalid E-mail Address"),

    channel: Yup.string().required("Channel is required"),

    address: Yup.string().required("Address is required"),

  });

  const validateComment = (value) => {

    let error;

    if (!value) {

      error = "Comment is required";

    }

    return error;

  };

  return (

    <Formik

      initialValues={initialValues}

      onSubmit={onSubmit}

      validationSchema={validationSchema}

    >

      {(formik) => {

        console.log(formik);

        return (

          <Form>

            <div className='form-control'>

              <label htmlFor='name'>Name</label>

              <Field type='text' name='name' id='name' />

              <ErrorMessage name='name' component={TextError} />

            </div>

            <div className='form-control'>

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

              <Field type='text' name='email' id='email' />

              <ErrorMessage name='email'>

                {(errorMsg) => <div className='error'>{errorMsg}</div>}

              </ErrorMessage>

            </div>

            <div className='form-control'>

              <label htmlFor='channel'>Channel</label>

              <Field type='text' name='channel' id='channel' />

              <ErrorMessage name='channel' />

            </div>

            <div className='form-control'>

              <label htmlFor='comments'>Comments</label>

              <Field

                as='textarea'

                placeholder='Enter you comment'

                name='comments'

                id='comments'

                validate={validateComment}

              />

              <ErrorMessage name='comments' component={TextError} />

            </div>

            <div className='form-control'>

              <label htmlFor='address'>Address</label>

              <FastField name='address'>

                {(props) => {

                  const { field, form, meta } = props;

                  return (

                    <div>

                      <textarea id='address' {...field}></textarea>

                      {meta.touched && meta.error ? (

                        <div>{meta.error}</div>

                      ) : null}

                    </div>

                  );

                }}

              </FastField>

            </div>

            <div className='form-control'>

              <label htmlFor='facebook'>Facebook Profile</label>

              <Field type='text' name='social.facebook' id='facebook' />

            </div>

            <div className='form-control'>

              <label htmlFor='twitter'>Twitter Profile</label>

              <Field type='text' name='social.twitter' id='twitter' />

            </div>

            <div className='form-control'>

              <label htmlFor='phone1'>Phone Number 1 static</label>

              <Field type='text' name='phoneNumbers[0]' id='phone1' />

            </div>

            <div className='form-control'>

              <label htmlFor='phone2'>Phone Number 2 static</label>

              <Field type='text' name='phoneNumbers[1]' id='phone2' />

            </div>

            <FieldArray name='phNumbers'>

              {(props) => {

                const { push, remove, form } = props;

                const { values } = form;

                const { phNumbers } = values;

                return (

                  <div>

                    {phNumbers.map((phNum, index) => (

                      <div key={index}>

                        <Field name={`phNumbers[${index}]`} />

                        {index > 0 && (

                          <button type='button' onClick={() => remove(index)}>

                            -

                          </button>

                        )}

                        <button type='button' onClick={() => push("")}>

                          +

                        </button>

                      </div>

                    ))}

                  </div>

                );

              }}

            </FieldArray>

            <button

              type='button'

              onClick={() => formik.validateField("channel")}

            >

              Validate channel Field

            </button>

            <button type='button' onClick={() => formik.validateForm()}>

              Validate All

            </button>

            <button

              type='button'

              onClick={() => formik.setFieldTouched("channel")}

            >

              Touched channel Field

            </button>

            <button

              type='button'

              onClick={() => {

                formik.setTouched({

                  name: true,

                  email: true,

                  channel: true,

                  comments: true,

                });

              }}

            >

              Touched All

            </button>

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

          </Form>

        );

      }}

    </Formik>

  );

};

export default YoutubeForm;

Disable Submit Button

1. On invalid form

 <button type='submit' disabled={!formik.isValid}>

              Submit

            </button>

Disable button on load (when few number of inputs )

 <Formik

      initialValues={initialValues}

      onSubmit={onSubmit}

      validationSchema={validationSchema}

      validateOnMount //validate form on mount/load

    >

      {(formik) => {

 return (

          <Form>

<button type='submit' disabled={!formik.isValid}>

              Submit

            </button>

          </Form>

        );

      }}

    </Formik>

Disable form in edit mode when no field is changed (not use validateOnMount)

<Formik

      initialValues={initialValues}

      onSubmit={onSubmit}

      validationSchema={validationSchema}

    >

      {(formik) => {

<button type='submit' disabled={!(formik.dirty && formik.isValid)}>

              Submit

            </button>

          </Form>

        );

      }}

    </Formik>

Load Save Data

import {

  ErrorMessage,

  Field,

  Form,

  Formik,

  FieldArray,

  FastField,

} from "formik";

import React, { useState } from "react";

import \* as Yup from "yup";

import TextError from "./TextError";

const YoutubeForm = () => {

  const [formValues, setFormValues] = useState(null); //step1

  const initialValues = {

    name: "",

    email: "",

    channel: "",

    comments: "",

    address: "",

    social: {

      facebook: "",

      twitter: "",

    },

    phoneNumbers: ["", ""],

    phNumbers: [""],

  };

//step2

  const savedValues = {

    name: "Aamir",

    email: "aamir@gmail.com",

    channel: "CodexKing",

    comments: "Welcome to formik",

    address: "H 37,Gupta Colony",

    social: {

      facebook: "",

      twitter: "",

    },

    phoneNumbers: ["", ""],

    phNumbers: [""],

  };

  const onSubmit = (values) => {

    console.log("Form Data", values);

  };

  const validationSchema = Yup.object({

    name: Yup.string().required("Name is required"),

    email: Yup.string()

      .required("Email is required")

      .email("Invalid E-mail Address"),

    channel: Yup.string().required("Channel is required"),

    address: Yup.string().required("Address is required"),

  });

  const validateComment = (value) => {

    let error;

    if (!value) {

      error = "Comment is required";

    }

    return error;

  };

  return (

    <Formik

      initialValues={formValues || initialValues} //step3

      onSubmit={onSubmit}

      validationSchema={validationSchema}

      enableReinitialize //step3

    >

      {(formik) => {

        console.log(formik);

        return (

          <Form>

            <div className='form-control'>

              <label htmlFor='name'>Name</label>

              <Field type='text' name='name' id='name' />

              <ErrorMessage name='name' component={TextError} />

            </div>

            <div className='form-control'>

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

              <Field type='text' name='email' id='email' />

              <ErrorMessage name='email'>

                {(errorMsg) => <div className='error'>{errorMsg}</div>}

              </ErrorMessage>

            </div>

            <div className='form-control'>

              <label htmlFor='channel'>Channel</label>

              <Field type='text' name='channel' id='channel' />

              <ErrorMessage name='channel' />

            </div>

            <div className='form-control'>

              <label htmlFor='comments'>Comments</label>

              <Field

                as='textarea'

                placeholder='Enter you comment'

                name='comments'

                id='comments'

                validate={validateComment}

              />

              <ErrorMessage name='comments' component={TextError} />

            </div>

            <div className='form-control'>

              <label htmlFor='address'>Address</label>

              <FastField name='address'>

                {(props) => {

                  const { field, form, meta } = props;

                  return (

                    <div>

                      <textarea id='address' {...field}></textarea>

                      {meta.touched && meta.error ? (

                        <div>{meta.error}</div>

                      ) : null}

                    </div>

                  );

                }}

              </FastField>

            </div>

            <div className='form-control'>

              <label htmlFor='facebook'>Facebook Profile</label>

              <Field type='text' name='social.facebook' id='facebook' />

            </div>

            <div className='form-control'>

              <label htmlFor='twitter'>Twitter Profile</label>

              <Field type='text' name='social.twitter' id='twitter' />

            </div>

            <div className='form-control'>

              <label htmlFor='phone1'>Phone Number 1 static</label>

              <Field type='text' name='phoneNumbers[0]' id='phone1' />

            </div>

            <div className='form-control'>

              <label htmlFor='phone2'>Phone Number 2 static</label>

              <Field type='text' name='phoneNumbers[1]' id='phone2' />

            </div>

            <FieldArray name='phNumbers'>

              {(props) => {

                const { push, remove, form } = props;

                const { values } = form;

                const { phNumbers } = values;

                return (

                  <div>

                    {phNumbers.map((phNum, index) => (

                      <div key={index}>

                        <Field name={`phNumbers[${index}]`} />

                        {index > 0 && (

                          <button type='button' onClick={() => remove(index)}>

                            -

                          </button>

                        )}

                        <button type='button' onClick={() => push("")}>

                          +

                        </button>

                      </div>

                    ))}

                  </div>

                );

              }}

            </FieldArray>

            <br />

            <button type='button' onClick={() => setFormValues(savedValues)}>

              Load Data

            </button> {/\*step4\*/}

            <button type='submit' disabled={!formik.isValid}>

              Submit

            </button>

          </Form>

        );

      }}

    </Formik>

  );

};

export default YoutubeForm;

Reset Form on Submit

 const onSubmit = (values, onSubmitProps) => {

    console.log("Form Data", values);

    console.log("Submit Props", onSubmitProps);

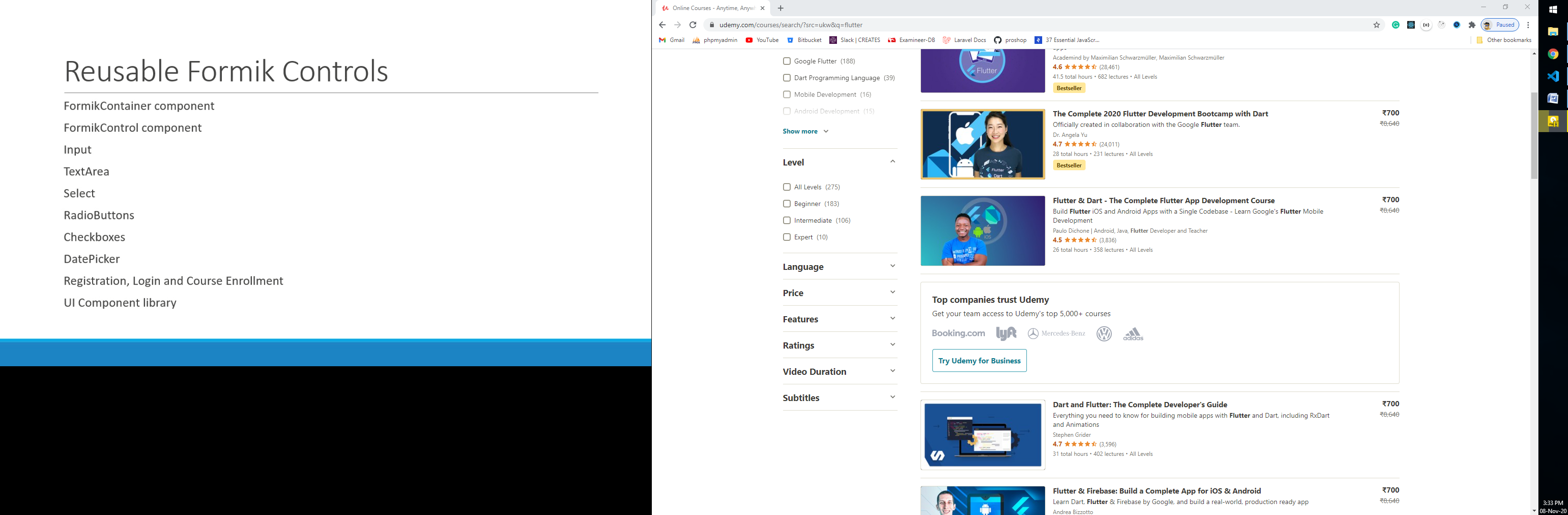
    onSubmitProps.setSubmitting(false);

    onSubmitProps.resetForm();

  };

Reset form by reset button

<button type='reset'>Reset</button>



Custom Form elements

App.js

import "./App.css";

import FormikContainer from "./components/FormikContainer";

function App() {

  return (

    <div className='App'>

      <FormikContainer />

    </div>

  );

}

export default App;

FormikContaier.js

import React from "react";

import { Form, Formik } from "formik";

import \* as Yup from "yup";

import FormikControl from "./FormikControl";

const FormikContainer = () => {

  const initialValues = {

    email: "",

    description: "",

    selectOption: "",

    radioOption: "",

    checkboxOption: [],

    birthDate: null,

  };

  const dropDownOptions = [

    { key: "Select Option", value: "" },

    { key: "option 1", value: "Option1" },

    { key: "option 2", value: "Option2" },

    { key: "option 3", value: "Option3" },

  ];

  const radioOptions = [

    { key: "roption 1", value: "rOption1" },

    { key: "roption 2", value: "rOption2" },

    { key: "roption 3", value: "rOption3" },

  ];

  const checkboxOptions = [

    { key: "roption 1", value: "cOption1" },

    { key: "roption 2", value: "cOption2" },

    { key: "roption 3", value: "cOption3" },

  ];

  const validationSchema = Yup.object({

    email: Yup.string()

      .required("Required")

      .email("Enter Valid E-mail Address"),

    description: Yup.string().required("Required"),

    selectOption: Yup.string().required("Required"),

    radioOption: Yup.string().required("Required"),

    checkboxOption: Yup.array().required("Required"),

    birthDate: Yup.date().required("Required").nullable(),

  });

  const onSubmit = (values) => {

    console.log(`Form Data `);

    console.log(values);

    console.log("save Data");

    console.log(JSON.parse(JSON.stringify(values)));

  };

  return (

    <Formik

      initialValues={initialValues}

      validationSchema={validationSchema}

      onSubmit={onSubmit}

    >

      {(formik) => {

        return (

          <Form>

            <FormikControl

              control='input'

              label='Email'

              name='email'

              type='email'

            />

            <FormikControl

              control='textarea'

              label='Description'

              name='description'

            />

            <FormikControl

              control='select'

              label='Select Option'

              name='selectOption'

              options={dropDownOptions}

            />

            <FormikControl

              control='radio'

              label='Radio Option'

              name='radioOption'

              options={radioOptions}

            />

            <FormikControl

              control='checkbox'

              label='Checkbox Option'

              name='checkboxOption'

              options={checkboxOptions}

            />

            <FormikControl control='date' label='Select DOB' name='birthDate' />

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

          </Form>

        );

      }}

    </Formik>

  );

};

export default FormikContainer;

FormControl.js

import React from "react";

import CheckboxButton from "./CheckboxButton";

import DatePickerInput from "./DatePickerInput";

import Input from "./Input";

import RadioButton from "./RadioButton";

import Select from "./Select";

import Textarea from "./Textarea";

const FormikControl = (props) => {

  const { control, ...rest } = props;

  switch (control) {

    case "input":

      return <Input {...rest} />;

    case "textarea":

      return <Textarea {...rest} />;

    case "select":

      return <Select {...rest} />;

    case "radio":

      return <RadioButton {...rest} />;

    case "checkbox":

      return <CheckboxButton {...rest} />;

    case "date":

      return <DatePickerInput {...rest} />;

    default:

      return null;

  }

};

export default FormikControl;

Input.js

import { ErrorMessage, Field } from "formik";

import React from "react";

import TextError from "./TextError";

const Input = (props) => {

  const { label, name, ...rest } = props;

  return (

    <div className='form-control'>

      <label htmlFor={name}>{label}</label>

      <Field id={name} name={name} {...rest} />

      <ErrorMessage name={name} component={TextError} />

    </div>

  );

};

export default Input;

TextArea.js

import { ErrorMessage, Field } from "formik";

import React from "react";

import TextError from "./TextError";

const Textarea = (props) => {

  const { label, name, ...rest } = props;

  return (

    <div className='form-control'>

      <label htmlFor={name}>{label}</label>

      <Field as='textarea' name={name} id={name} {...rest} />

      <ErrorMessage name={name} component={TextError} />

    </div>

  );

};

export default Textarea;

Select.js

import { ErrorMessage, Field } from "formik";

import React from "react";

import TextError from "./TextError";

const Select = (props) => {

  const { label, name, options, ...rest } = props;

  return (

    <div className='form-control'>

      <label htmlFor={name}>{label}</label>

      <Field as='select' name={name} id={name} {...rest}>

        {options.map((op) => (

          <option key={op.value} value={op.value}>

            {op.key}

          </option>

        ))}

      </Field>

      <ErrorMessage name={name} component={TextError} />

    </div>

  );

};

export default Select;

RadioButton.js

import { ErrorMessage, Field } from "formik";

import React, { Fragment } from "react";

import TextError from "./TextError";

const RadioButton = (props) => {

  const { label, name, options, ...rest } = props;

  return (

    <div className='form-control'>

      <label>{label}</label>

      <Field name={name} {...rest}>

        {({ field }) => {

          return options.map((op) => {

            return (

              <Fragment key={op.value}>

                <input

                  type='radio'

                  id={op.value}

                  {...field}

                  {...rest}

                  checked={field.value === op.value}

                  value={op.value}

                />

                <label htmlFor={op.value}>{op.key}</label>

              </Fragment>

            );

          });

        }}

      </Field>

      <ErrorMessage name={name} component={TextError} />

    </div>

  );

};

export default RadioButton;

CheckboxButton.js

import { ErrorMessage, Field } from "formik";

import React, { Fragment } from "react";

import TextError from "./TextError";

const CheckboxButton = (props) => {

  const { label, name, options, ...rest } = props;

  return (

    <div className='form-control'>

      <label>{label}</label>

      <Field name={name} {...rest}>

        {({ field }) => {

          console.log(field);

          return options.map((op) => {

            const isCheck = field.value.includes(op.value);

            return (

              <Fragment key={op.value}>

                <input

                  type='checkbox'

                  id={op.value}

                  {...field}

                  {...rest}

                  value={op.value}

                  checked={isCheck}

                />

                <label htmlFor={op.value}>{op.key}</label>

              </Fragment>

            );

          });

        }}

      </Field>

      <ErrorMessage name={name} component={TextError} />

    </div>

  );

};

export default CheckboxButton;

DatePickerInput.js

import { ErrorMessage, Field } from "formik";

import React from "react";

import DatePicker from "react-datepicker";

import "react-datepicker/dist/react-datepicker.css";

import TextError from "./TextError";

const DatePickerInput = (props) => {

  const { name, label, ...rest } = props;

  return (

    <div className='form-control'>

      <label htmlFor={name}>{label}</label>

      <Field name={name}>

        {({ field, form }) => {

          const { setFieldValue } = form;

          const { value } = field;

          return (

            <DatePicker

              id={name}

              {...field}

              {...rest}

              selected={value}

              onChange={(val) => setFieldValue(name, val)}

            />

          );

        }}

      </Field>

      <ErrorMessage name={name} component={TextError} />

    </div>

  );

};

export default DatePickerInput;

TextError.js

import React from "react";

const TextError = (props) => {

  return <div className='error'>{props.children}</div>;

};

export default TextError;

Yup validation

1. Email
2. Confirm password
3. Conditional Validation

