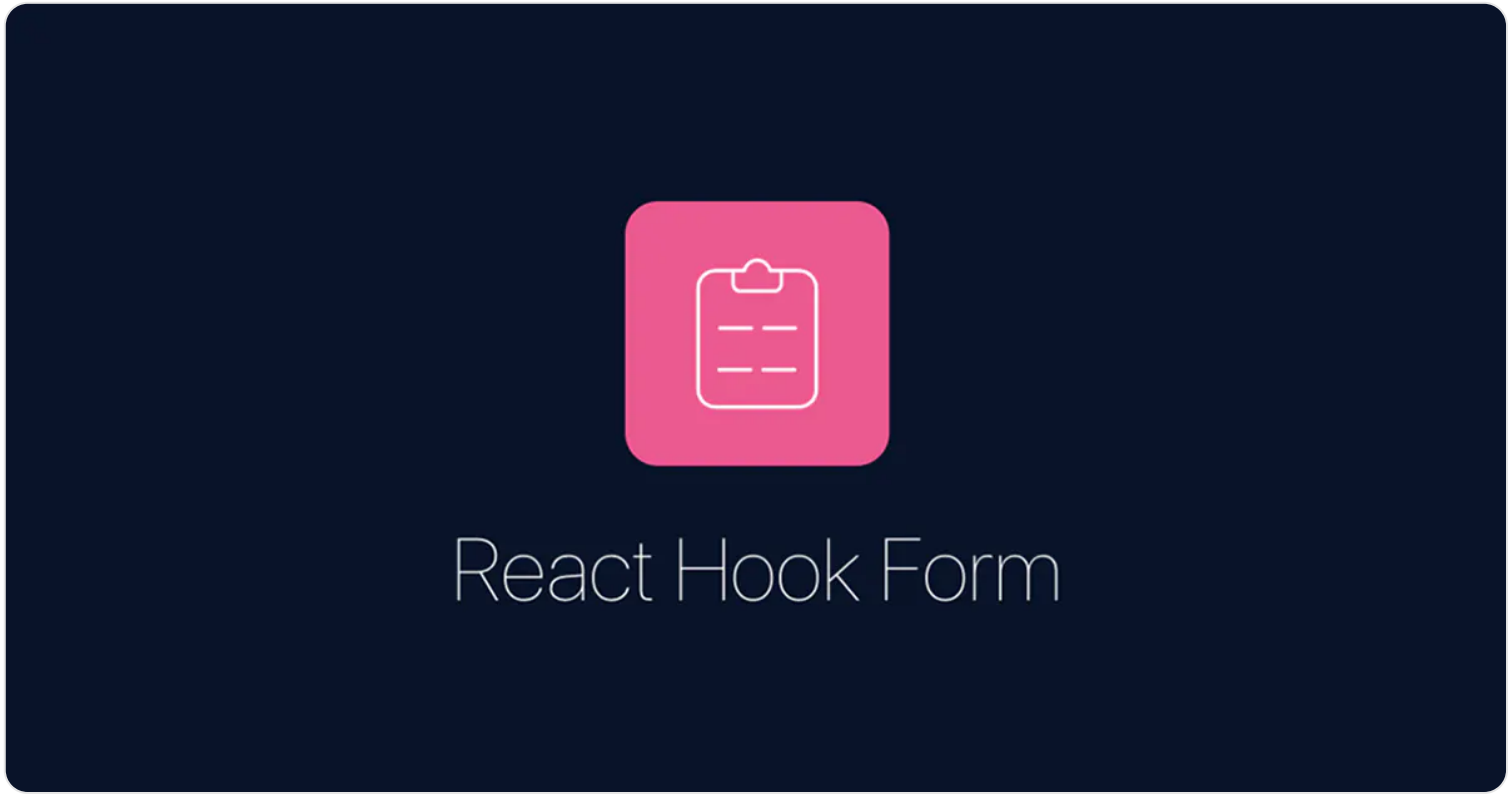


Building a simple login form with Material UI and React Hook Form

Published on Jul 23, 2020 · 7 min read

Prerequisites

This tutorial will use Next.js as the application framework. I will assume that you already read my previous article about using Material UI in Next.js because we will use it as a starting point. Still, you should be able to follow along if you already have a basic understanding of React.

Install dependencies

Install `react-hook-form` in your project directory:

```
yarn add -E react-hook-form
```

Get started

In your `pages` directory, create a new file called `login.tsx`. And, write the following code:

```
import React from 'react';
// Modules
import { NextPage } from 'next/types';

const LoginPage: NextPage = () => {
  return <div>Login Page</div>;
```

Then, save your file and run the application in the development server by running the following command in your terminal:

```
yarn dev
```

By executing the command above, your application will run on `localhost:3000`. You can try to visit `localhost:3000/login` and you will see a login page.

writing some codes to build the user interface. Open `login.tsx` and import these modules:

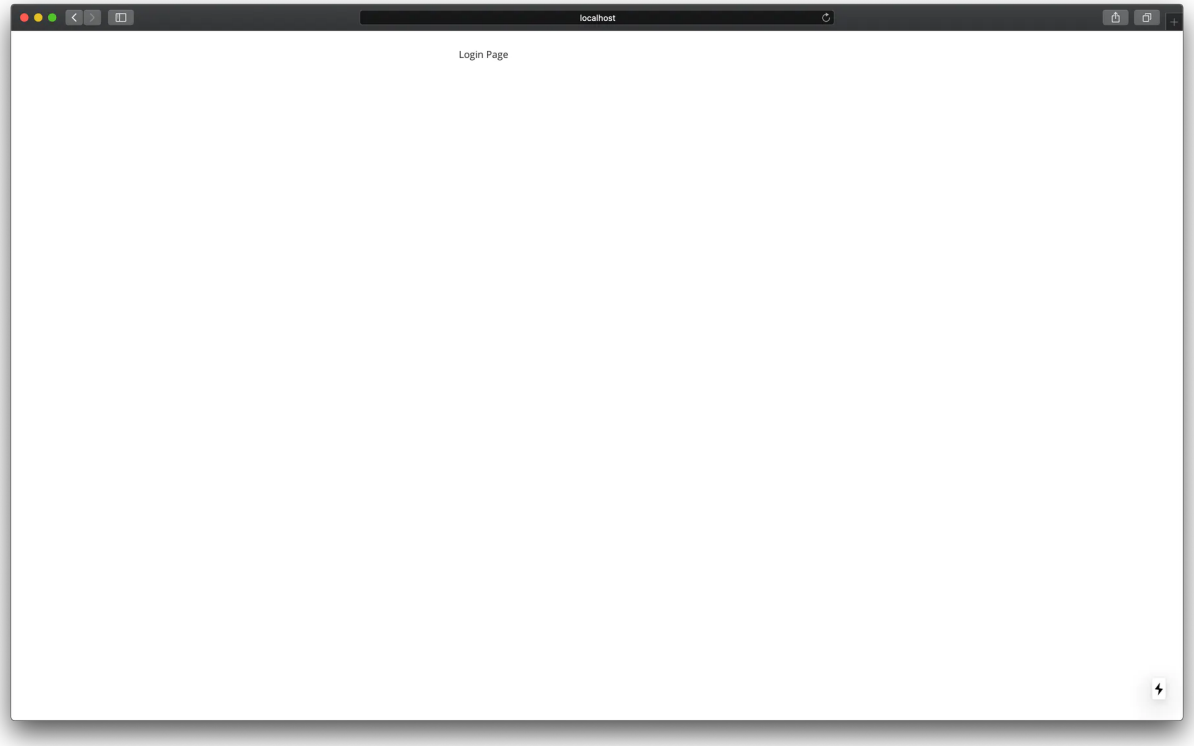
```
import Button from '@material-ui/core/Button';
import Container from '@material-ui/core/Container';
import Grid from '@material-ui/core/Grid';
import TextField from '@material-ui/core/TextField';
import { makeStyles } from '@material-ui/core/styles';
```

By default, Material UI uses JSS-based styling to style your application. However, they allow you to use other styling solutions, such as Plain CSS, Global CSS, Styled Components, CSS Modules, and Emotion. You can learn more about each styling solution from their official documentation, but, in this tutorial, I will use the default styling solution provided by Material UI to keep it simple. Open `login.tsx` and write the following code:

```
// ... imports

const useStyles = makeStyles((theme) => ({
  container: {
    padding: theme.spacing(3),
  },
}));

const LoginPage: NextPage = () => {
  const classes = useStyles();

  return (
    <Container className={classes.container} maxWidth="xs">
```


```
export default LoginPage;
```

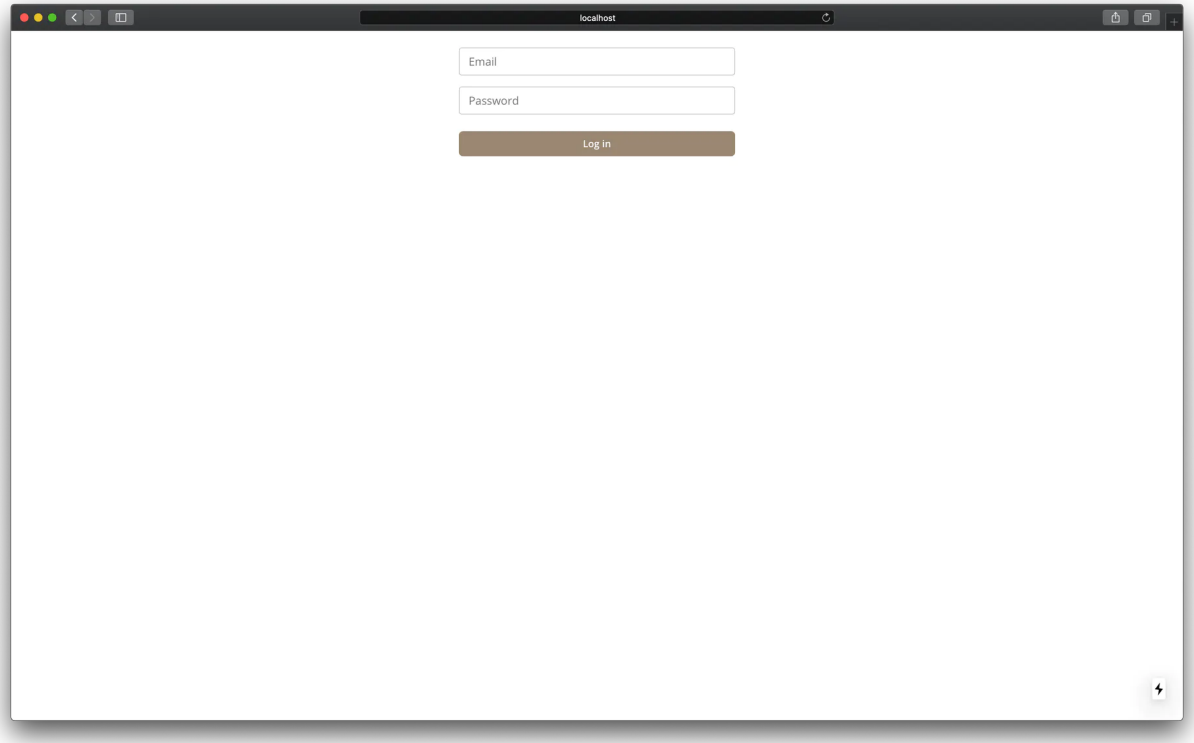
Then, save your file, and you will see the changes you just made in the browser. Thanks to Next.js for enabling Fast Refresh which helps us to see changes reflected directly to the browser without having to re-run the development server.




```
// ... rest of your codes

const LoginPage: NextPage = () => {
  const classes = useStyles();

  return (
    <Container className={classes.container} maxWidth="xs">
      <form>
        <Grid container spacing={3}>
          <Grid item xs={12}>
            <Grid container spacing={2}>
              <Grid item xs={12}>
                <TextField fullWidth label="Email" name="email" size="small" variant="out
              </Grid>
              <Grid item xs={12}>
                <TextField
                  fullWidth
                  label="Password"
                  name="password"
                  size="small"
                  type="password"
                  variant="outlined"
                />
              </Grid>
            </Grid>
          </Grid>
          <Grid item xs={12}>
            <Button color="secondary" fullWidth type="submit" variant="contained">
              Log in
            </Button>
          </Grid>
```


```
// ... rest of your codes
```

Then, save your file, and you will see the changes you just made in the browser.



Now, we finish building the user interface of our login form. By the end of this section, your code should look like this:


```
import Button from '@material-ui/core/Button';
import Container from '@material-ui/core/Container';
import Grid from '@material-ui/core/Grid';
import TextField from '@material-ui/core/TextField';
import { makeStyles } from '@material-ui/core/styles';

const useStyles = makeStyles((theme) => ({
  container: {
    padding: theme.spacing(3),
  },
}));

const LoginPage: NextPage = () => {
  const classes = useStyles();

  return (
    <Container className={classes.container} maxWidth="xs">
      <form>
        <Grid container spacing={3}>
          <Grid item xs={12}>
            <Grid container spacing={2}>
              <Grid item xs={12}>
                <TextField fullWidth label="Email" name="email" size="small" variant="ou
              </Grid>
              <Grid item xs={12}>
                <TextField
                  fullWidth
                  label="Password"
                  name="password"
                  size="small"
                  type="password"
                  variant="outlined"
                />
```


```
            Log in
          </Button>
        </Grid>
      </Grid>
    </form>
  </Container>
 );
};

export default LoginPage;
```

Implementing React Hook Form

In this section, we are going to implement react hook form to handle the login functionality. To use React Hook Form, we need to import `react-hook-form`. Write the following code in your `login.tsx`:

```
import { useForm } from 'react-hook-form';
```

Since we are using TypeScript, we need to define an interface to make our form strongly typed. Write the following code in your `login.tsx`:

```
// ... rest of your codes
```


```
const useStyles = makeStyles((theme) => ({
  container: {
    padding: theme.spacing(3),
  },
}));

// ... rest of your codes
```

Then, write the following code to implement `react-hook-form` in our component:

```
// ... rest of your codes

const LoginPage: NextPage = () => {
  const { handleSubmit, register } = useForm<FormData>();

  const onSubmit = handleSubmit((data) => {
    console.log(data);
  });

  // ... rest of your codes
};

// ... rest of your codes
```

One of the key concepts in React Hook Form is to **register** your uncontrolled component into the hook. This will make its value available for both the form validation and submission. Each field is


```
<Grid item xs={12}>
  <TextField
    fullWidth
    inputRef={register}
    label="Email"
    name="email"
    size="small"
    variant="outlined"
  />
</Grid>
<Grid item xs={12}>
  <TextField
    fullWidth
    inputRef={register}
    label="Password"
    name="password"
    size="small"
    type="password"
    variant="outlined"
  />
</Grid>
```

Then, pass `onSubmit` into the `onSubmit` props on `<form />` component:

```
<form onSubmit={onSubmit}>
  {/* ... rest of your codes */}
</form>
```

By the end of this section, your code will look like this:

```
import React from 'react';
// Modules
import { NextPage } from 'next/types';
import { useForm } from 'react-hook-form';
// MUI Core
import Button from '@material-ui/core/Button';
import Container from '@material-ui/core/Container';
import Grid from '@material-ui/core/Grid';
import TextField from '@material-ui/core/TextField';
import { makeStyles } from '@material-ui/core/styles';

interface FormData {
  email: string;
  password: string;
}

const useStyles = makeStyles((theme) => ({
  container: {
    padding: theme.spacing(3),
  },
}));

const LoginPage: NextPage = () => {
  const { handleSubmit, register } = useForm<FormData>();

  const classes = useStyles();

  const onSubmit = handleSubmit((data) => {
```


```
<form onSubmit={onSubmit}>
  <Grid container spacing={3}>
    <Grid item xs={12}>
      <Grid container spacing={2}>
        <Grid item xs={12}>
          <TextField
            fullWidth
            inputRef={register}
            label="Email"
            name="email"
            size="small"
            variant="outlined"
          />
        </Grid>
        <Grid item xs={12}>
          <TextField
            fullWidth
            inputRef={register}
            label="Password"
            name="password"
            size="small"
            type="password"
            variant="outlined"
          />
        </Grid>
      </Grid>
    </Grid>
    <Grid item xs={12}>
      <Button color="secondary" fullWidth type="submit" variant="contained">
        Log in
      </Button>
    </Grid>
```


```
export default LoginPage;
```

Summary

Hopefully, this article has helped you to understand how to implement React Hook Form in Material UI. This article only covers basic concept of React Hook Form. If you want to learn more about React Hook Form, you can visit its official documentation here.

Articles you might like

improve their productivity and prevent them from producing bugs.

Published on Mar 10, 2021 · 2 min read

unknown now that may become known in the future, in other words an asynchronous value.

Published on Mar 10, 2021 · 2 min read

React Native over-the-air app update with CodePush

CodePush is a tool developed by Microsoft which allows developers to quickly and easily manage and

Building a simple login form in React Native using React Hook Form

A short step-by-step tutorial about how to build a simple login form in

Building floating logo bubbles on Stripe.com using React Hooks

In this tutorial, we are going to clone animating logo bubbles from Stripe.com using React Hooks.

Published on Aug 11, 2020 · 9 min read

Building custom React Hooks to fetch data from Firebase Firestore

In this article, I want to share my experience of building custom React Hooks to fetch data from Firebase Firestore.

Published on Aug 5, 2020 · 5 min read

Introduction to React Hooks

Hooks solve a wide variety of seemingly unconnected problems in React. Whether you're new to React or use it daily, you might recognise some of these problems.

Published on Jul 21, 2020 · 3 min read

N+1 problem in GraphQL

The n+1 problem occurs because GraphQL executes a separate function – called resolver – for each field.

Published on Jul 18, 2020 · 2 min read

GraphQL vs REST

GraphQL has been introduced as a revolutionary alternative to REST APIs. However, it has its pros and cons.

Published on Jul 16, 2020 · 3 min read

Data fetching in Next.js 9.3+

There are two forms of pre-rendering: Static Generation and Server-side Rendering.

Published on Jul 15, 2020 · 6 min read

When should we consider using third-party libraries?

Step-by-step guidelines to implement Material UI in

you to save time because you do not need to develop the functionality that the library gives.

Published on Jul 13, 2020 · 4 min read

integrate Material UI in Next.js applications. We will build a very basic website using Material UI and Next.js.

Published on Jul 12, 2020 · 4 min read