Gaurav Singhal

How to Use React Bootstrap with Redux

Gaurav Singhal

Sep 15, 2020 • 15 Min read • 5,184 Views

Sep 15, 2020 • 15 Min read • 5,184 Views

Web Development React

Introduction

28

Introduction

Bootstrap is the most popular CSS framework, used by over a million websites on the internet. So it's imperative to know how we can integrate this excellent CSS framework with our favorite JavaScript front-end stack—React and Redux.

One way to add Bootstrap into your app is by using CDN links. But, fortunately, there's also already an npm library out there that solves our problem.

React-Bootstrap

React-Bootstrap is the most popular front-end framework. It's rebuilt for React.js—that is, it completely replaces Bootstrap Javascript. React-Bootstrap is an npm package that completely re-implements Bootstrap components for React and has no dependency on `JQuery`. It uses the state as supposed to direct DOM manipulation, which makes this package more reliable in the React world.

Installing React-Bootstrap

To install `react-bootstrap`, run the following command in your terminal.

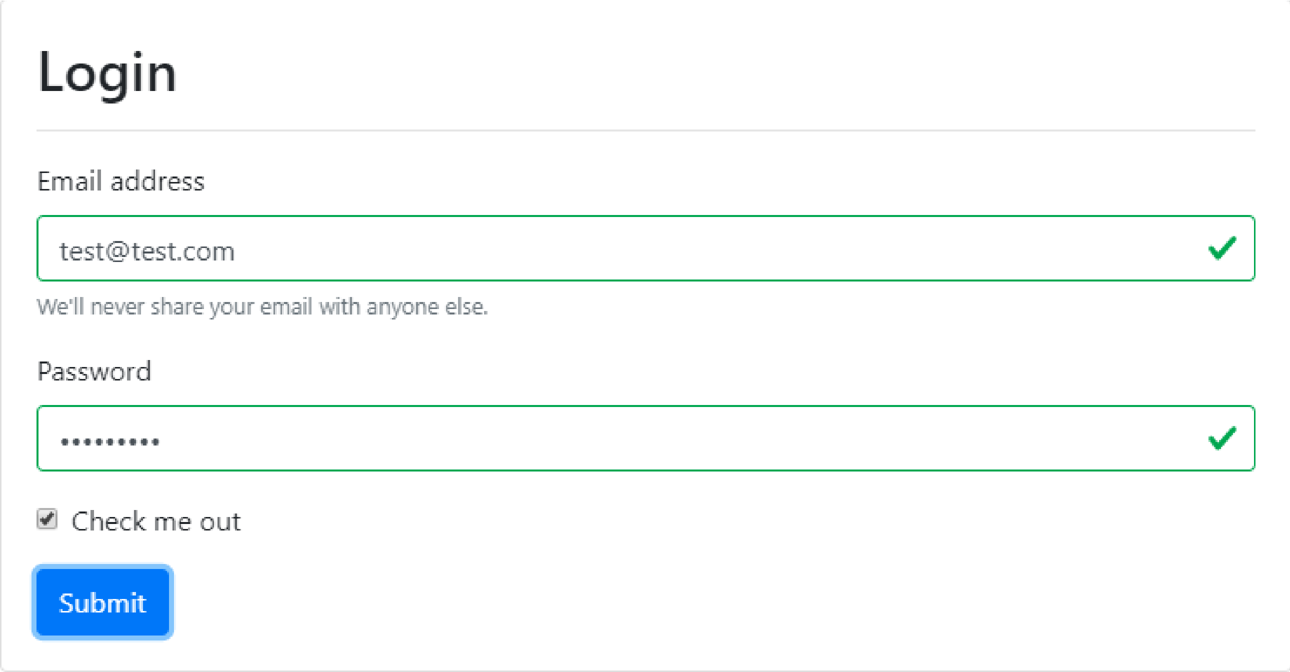
console

```console
npm install react-bootstrap bootstrap
```

Note: Please make sure you have node installed in your system before running the command.

Example

For this guide, we will use a `<LoginForm />` component from the React Bootstrap documentation and validate the fields using Redux.



Login

Email address

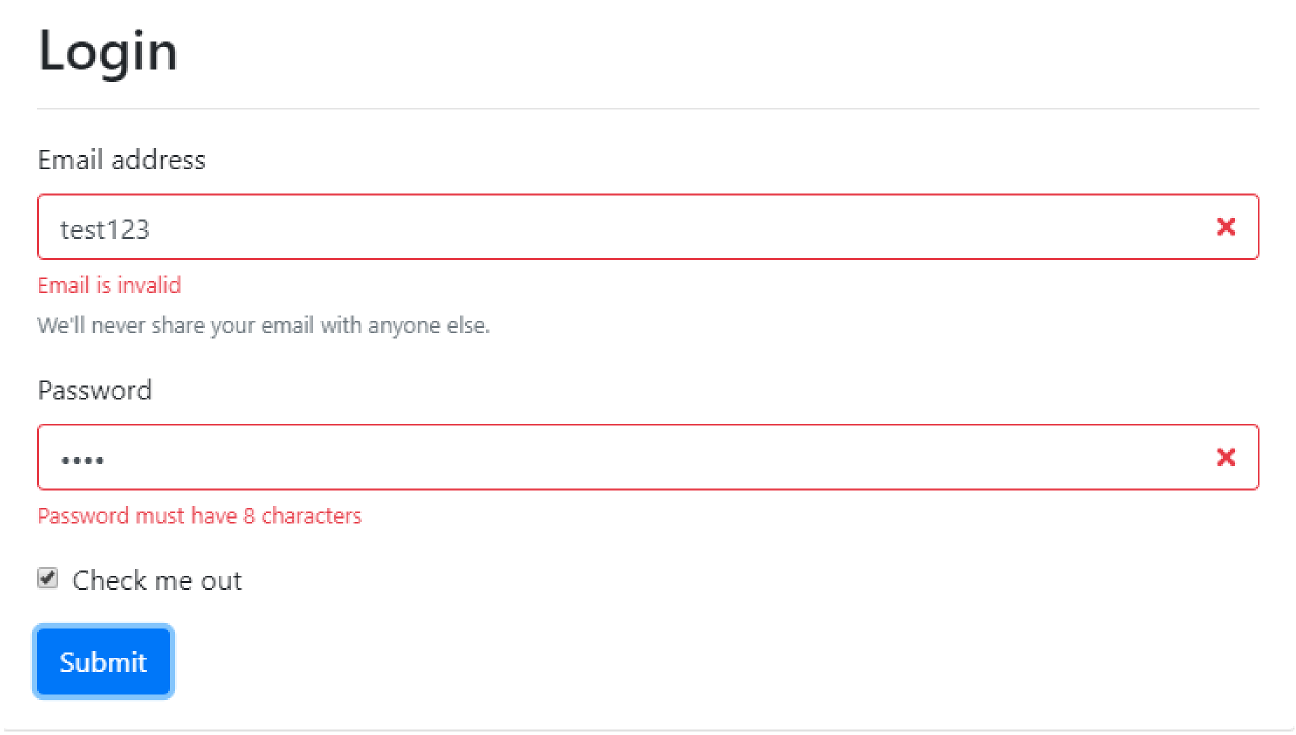
test@test.com ✓

We'll never share your email with anyone else.

Password

•••••••••• ✓

☑ Check me out



Login

Email address

test123 ✖

Email is invalid

We'll never share your email with anyone else.

Password

•••• ✖

Password must have 8 characters

☑ Check me out

LoginForm Component

We will import the `<Form />` and `<Button />` components from `react-bootstrap` and use them in our `<LoginForm />` component.

jsx

```jsx
import Form from "react-bootstrap/Form";
import Button from "react-bootstrap/Button";

const LoginForm = () => (
  <Form>
```

```
    <h2>Login</h2>
    <hr />
    <Form.Group controlId="formBasicEmail">
      <Form.Label>Email address</Form.Label>
      <Form.Control type="email" placeholder="Enter email" />
      <Form.Text className="text-muted">
        We'll never share your email with anyone else.
      </Form.Text>
    </Form.Group>

    <Form.Group controlId="formBasicPassword">
      <Form.Label>Password</Form.Label>
      <Form.Control type="password" placeholder="Password" />
    </Form.Group>
    <Form.Group controlId="formBasicCheckbox">
      <Form.Check type="checkbox" label="Check me out" />
    </Form.Group>
    <Button variant="primary" type="button">
      Submit
    </Button>
  </Form>
);

export default LoginForm;
```

Building the Store

In the global store object, we will store values and errors of the form.

js

```js
const initialState = {
  loginForm: {
    values: {
      email: "",
      password: ""
    },
    errors: {
      email: "",
      password: ""
    }
  }
};

export default (state = initialState, action) => {
  if (action.type === "FORM_SUBMIT") {
    const { email, password } = action.payload;
    const values = {
      email,
      password
    };
    const errors = {}; // validate fields
    return {
      loginForm: {
        values,
        errors
      }
    };
  }
  return state;
};
```

To validate the fields, we will write a `validateEmail()` function to check whether the email entered by the user is valid or not and whether the password has at least 8 characters.

js

```js
function validateEmail(email) {
  var re = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))
  return re.test(String(email).toLowerCase());
}

const setErrors = (email, password) => {
  let errors = { email: "", password: "" };
  if (!email && email.length === 0) {
    errors.email = "Email is required";
  } else if (!validateEmail(email)) {
    errors.email = "Email is invalid";
  }
  if (!password && password.length === 0) {
    errors.password = "Password is required";
  } else if (password.length < 8) {
    errors.password = "Password must have 8 characters";
  }
  return errors;
};

// ...

export default (state = initialState, action) => {
  if (action.type === "FORM_SUBMIT") {
    const { email, password } = action.payload;
    const values = {
      email,
      password
    };
    const errors = setErrors(email, password); // validate fields
    return {
      loginForm: {
        values,
        errors
      }
    };
  }
  return state;
};
```

Wrapping Root Component with Provider

To make the store accessible in all the components, we need to wrap the root component with the `<Provider />` component from the `react-redux` package.

Also, notice that it's essential to import bootstrap CSS in the root component so that the styles are applied to the component.

jsx

```jsx
import "bootstrap/dist/css/bootstrap.min.css";
import Card from "react-bootstrap/Card";
import { Provider } from "react-redux";
import { createStore } from "redux";

import reducer from "./reducer";

import LoginForm from "./LoginForm";

const store = createStore(reducer);

function App() {
  return (
    <Provider store={store}>
      <div className="App">
        <Card body>
          <LoginForm />
        </Card>
      </div>
    </Provider>
  );
}
```

Connecting <LoginForm /> Component with Redux

First, we have to make this a controlled component—that is, the form data must be controlled by the component's internal state.

```jsx
import React, { useState } from "react";

const LoginForm = props => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  return (
    <Form>
      <h2>Login</h2>
      <hr />
      <Form.Group controlId="formBasicEmail">
        <Form.Label>Email address</Form.Label>
        <Form.Control
          type="email"
          placeholder="Enter email"
          onChange={e => setEmail(e.target.value)}
        />
        <Form.Control.Feedback type="invalid">
          {props.loginForm.errors.email}
        </Form.Control.Feedback>
        <Form.Text className="text-muted">
          We'll never share your email with anyone else.
        </Form.Text>
      </Form.Group>

      <Form.Group controlId="formBasicPassword">
        <Form.Label>Password</Form.Label>
        <Form.Control
          type="password"
          placeholder="Password"
          onChange={e => setPassword(e.target.value)}
        />
        <Form.Control.Feedback type="invalid">
          {props.loginForm.errors.password}
        </Form.Control.Feedback>
      </Form.Group>
      <Form.Group controlId="formBasicCheckbox">
        <Form.Check type="checkbox" label="Check me out" />
      </Form.Group>
      <Button variant="primary" type="button">
```

```
        Submit
      </Button>
    </Form>
  );
};
```

We have discussed React hooks in an earlier (guide)/guides/change-page-background-color-each-route. Hooks are modern APIs provided by Redux to manage state in a functional component.

Next, we will connect the form to Redux using the `connect()` method.

jsx

```jsx
// ...
import { connect } from "react-redux";

const LoginForm = props => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  return (
    <Form>
      <h2>Login</h2>
      <hr />
      <Form.Group controlId="formBasicEmail">
        <Form.Label>Email address</Form.Label>
        <Form.Control
          type="email"
          placeholder="Enter email"
          isInvalid={props.loginForm.errors.email.length > 0}
          isValid={
            props.loginForm.values.email &&
            props.loginForm.errors.email.length === 0
          }
          onChange={e => setEmail(e.target.value)}
        />
        <Form.Control.Feedback type="invalid">
          {props.loginForm.errors.email}
        </Form.Control.Feedback>
        <Form.Text className="text-muted">
          We'll never share your email with anyone else.
        </Form.Text>
      </Form.Group>

      <Form.Group controlId="formBasicPassword">
        <Form.Label>Password</Form.Label>
        <Form.Control
          type="password"
          placeholder="Password"
          isInvalid={props.loginForm.errors.password.length > 0}
          isValid={
            props.loginForm.values.password &&
            props.loginForm.errors.password.length === 0
          }
          onChange={e => setPassword(e.target.value)}
        />
        <Form.Control.Feedback type="invalid">
          {props.loginForm.errors.password}
        </Form.Control.Feedback>
      </Form.Group>
      <Form.Group controlId="formBasicCheckbox">
        <Form.Check type="checkbox" label="Check me out" />
      </Form.Group>
      <Button
        variant="primary"
```

```
    type="button"
    onClick={() =>
     props.dispatch({ type: "FORM_SUBMIT", payload: { email, passwor
    }
   >
    Submit
   </Button>
  </Form>
 );
};

const mapStateToProps = state => ({
 loginForm: state.loginForm
});

export default connect(mapStateToProps)(LoginForm);
```

To display whether a field is valid or not, we have to pass a Boolean value to the `isValid` and `isInvalid` props of the `<Form.Control />` component.To display the error message, we will use the `<Form.Control.Feedback />` component with `type` prop as invalid.

jsx

```
<Form.Control.Feedback type="invalid">
 {props.loginForm.errors.email}
</Form.Control.Feedback>
```

Complete Source Code

LoginForm.js

jsx

```jsx
import React, { useState } from "react";
import Form from "react-bootstrap/Form";
import Button from "react-bootstrap/Button";
import { connect } from "react-redux";

const LoginForm = props => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  return (
    <Form>
      <h2>Login</h2>
      <hr />
      <Form.Group controlId="formBasicEmail">
        <Form.Label>Email address</Form.Label>
        <Form.Control
          type="email"
          placeholder="Enter email"
          isInvalid={props.loginForm.errors.email.length > 0}
          isValid={
            props.loginForm.values.email &&
            props.loginForm.errors.email.length === 0
          }
          onChange={e => setEmail(e.target.value)}
        />
        <Form.Control.Feedback type="invalid">
          {props.loginForm.errors.email}
        </Form.Control.Feedback>
        <Form.Text className="text-muted">
```

```
          We'll never share your email with anyone else.
        </Form.Text>
      </Form.Group>

      <Form.Group controlId="formBasicPassword">
        <Form.Label>Password</Form.Label>
        <Form.Control
          type="password"
          placeholder="Password"
          isInvalid={props.loginForm.errors.password.length > 0}
          isValid={
            props.loginForm.values.password &&
            props.loginForm.errors.password.length === 0
          }
          onChange={e => setPassword(e.target.value)}
        />
        <Form.Control.Feedback type="invalid">
          {props.loginForm.errors.password}
        </Form.Control.Feedback>
      </Form.Group>
      <Form.Group controlId="formBasicCheckbox">
        <Form.Check type="checkbox" label="Check me out" />
      </Form.Group>
      <Button
        variant="primary"
        type="button"
        onClick={() =>
          props.dispatch({ type: "FORM_SUBMIT", payload: { email, passwor
        }
      >
        Submit
      </Button>
    </Form>
  );
};
```

```
const mapStateToProps = state => ({
  loginForm: state.loginForm
});

export default connect(mapStateToProps)(LoginForm);
```

On form submission, we will dispatch an `FORM_SUBMIT` action with the form values in the `payload`.

reducer.js

js

```
function validateEmail(email) {
  var re = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))
  return re.test(String(email).toLowerCase());
}

const initialState = {
  loginForm: {
    values: {
      email: "",
      password: ""
    },
    errors: {
      email: "",
      password: ""
    }
  }
};

const setErrors = (email, password) => {
  let errors = { email: "", password: "" };
  if (!email && email.length === 0) {
    errors.email = "Email is required";
  } else if (!validateEmail(email)) {
    errors.email = "Email is invalid";
  }
  if (!password && password.length === 0) {
    errors.password = "Password is required";
  } else if (password.length < 8) {
    errors.password = "Password must have 8 characters";
  }
  return errors;
};

export default (state = initialState, action) => {
  if (action.type === "FORM_SUBMIT") {
    const { email, password } = action.payload;
    const values = {
      email,
      password
    };
    const errors = setErrors(email, password);
    return {
      loginForm: {
        values,
        errors
      }
    };
  }
  return state;
};
```

index.js

jsx

```jsx
import React from "react";
import ReactDOM from "react-dom";
import "bootstrap/dist/css/bootstrap.min.css";
import Card from "react-bootstrap/Card";
import { Provider } from "react-redux";
import { createStore } from "redux";

import reducer from "./reducer";

import LoginForm from "./LoginForm";

const store = createStore(reducer);

function App() {
  return (
    <Provider store={store}>
      <div className="App">
        <Card body>
          <LoginForm />
        </Card>
      </div>
    </Provider>
  );
}

const rootElement = document.getElementById("root");
ReactDOM.render(<App />, rootElement);
```

Conclusion

React Bootstrap allows us to quickly set up our application with a decent design and focus more on the business logic of the app. It also can be used as a starting point for building applications with complex UI as it's very flexible and can be highly customized.

I hope you like this guide. If you have any queries regarding this topic, feel free to contact me at CodeAlphabet.

To learn more, check out React Bootstrap Forms.

28