Gaurav Singhal

How to Use React with React Bootstrap

Gaurav Singhal

Jun 22, 2020 • 10 Min read • 1,736 Views

Jun 22, 2020 • 10 Min read • 1,736 Views

Web Development Front End Web Dev... Client-side Framew... React

Introduction

9

Introduction

React Bootstrap is an open-source UI library built specifically for React to help you use native Bootstrap components as pure React components, such as modals, popovers, buttons, and so on. It's simple to use and can be easily integrated with an existing React app to customizie UI without compromising functionality.

This guide explores how to use React with React Bootstrap in your app.

Setting up a React Bootstrap App

Create a new React project.

shell

```
npx create-react-app react-and-react-bootstrap-app
```

Install react-bootstrap and Bootstrap.

shell

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

Creating a react-bootstrap Modal

Use a react-bootstrap modal that acts as a sign in form popup inside `App.js`. First, import the `useState` hook from React to use state variables inside a functional component. Then import both `Modal` and `Button` from react-bootstrap along with Bootstrap's minified CSS to use Bootstrap's style classes.

jsx

```jsx
import React,{useState} from 'react';
import {Modal} from 'react-bootstrap';
import {Button} from 'react-bootstrap';
import 'bootstrap/dist/css/bootstrap.min.css';
```

The `Modal` uses a `show` prop to control its *open state* on the page. `show` stores a boolean value indicating if the popup is supposed to be open or closed. You can toggle `show` using an event handler hooked to its buttons. Have a look at the following code to render a *react-bootstrap* modal that opens when you click on the sign-in button.

jsx

```jsx
...
function App() {
  const [show, setShow] = useState(false);

  const handleClose = () => setShow(false);
  const handleShow = () => setShow(true);

  return (
    <>
      <Button variant="primary" onClick={handleShow}>
        Sign In
      </Button>

      <Modal show={show} onHide={handleClose}>
        <Modal.Header closeButton>
          <Modal.Title>Sign In Form</Modal.Title>
        </Modal.Header>
        <Modal.Body>
         {/*add a react form here*/}
         </Modal.Body>
        <Modal.Footer>
          <Button variant="secondary" onClick={handleClose}>
            Sign Up
          </Button>
          <Button variant="primary" onClick={handleClose}>
            Sign In
          </Button>
        </Modal.Footer>
      </Modal>
    </>
  );
}

export default App;
```

Creating a Simple Form Inside the Modal

Populate the modal's body with a `Form` with two input fields: one for entering an email address and the other for a password. Create a state variable to store the email inside the state of the component.

jsx

```jsx
...
    const [email,setEmail]=useState('');
...
```

You can use Bootstrap's *rows* and *columns* to manage content's layout and enclose the form fields inside the `form-group` class for a neat view. Attach an `onChange` handler to the email field that captures the value entered in the field and sets it to the `email` state variable.

jsx

```jsx
...
<Modal.body>
 <Form>

    <div className="form-group">

        <div className="row">
            <div className="col-3">
                <label for="email" style={{fontWeight:500}} >Email:</labe
           <div className="col-9">
               <input name="email" type="text" placeholder="Enter Email..
                        onChange={(e)=>{setEmail(e.target.value)}}
                />
           </div>
        </div>

  </div>

  <div className="form-group">
      <div className="row">
           <div className="col-3">
              <label for="password" style={{fontWeight:500}}>Password:</l
           </div>
           <div className="col-9">
              <input name="password" type="password" placeholder="Enter P
           </div>
      </div>
  </div>

 </Form>
</Modal.body>
...
```

To verify whether the state has been set appropriately, log it on the console when the user clicks the sign-in button. You can do this inside modal's `handleClose()` method, which is fired when the modal closes as shown below.

Adding a react-bootstrap Alert Component

Alerts are a great way to give a sense of feedback to the user upon successful submission of forms or in case of errors. Import the `Alert` component from react-bootstrap on the top inside `App.js`.

jsx

```
import {Alert} from 'react-bootstrap';
```

Create a separate functional component for rendering the alert. It returns a react-bootstrap alert containing a heading, content, and a close button. Just like the modal, it uses the `show` prop to indicate whether the alert will remain opened or closed in the current view.

jsx

```jsx
function MyAlert() {
  const [show, setShow] = useState(true);

  return (
    <>
      <Alert show={show} variant="success">
        <Alert.Heading>Welcome user :)</Alert.Heading>
        <p>
         you have successfully signed in!
        </p>
        <hr />
        <div className="d-flex justify-content-end">
          <Button onClick={() => setShow(false)} variant="outline-succes
            Close me ya'll!
          </Button>
        </div>
      </Alert>

      {!show && <Button onClick={() => setShow(true)}>Show Alert</Button
    </>
  );
}
```

Render `MyAlert` inside `App.js` .

jsx

```jsx
...
      <MyAlert/>
```

```
...
```

By default, the alert will remain open on the page due to the initial value of `show`.

Combining All the Features

For the `MyAlert` component to interact with the `App` component, pass in `email` and another state variable as props to the former. This state variable controls the alert's `show` state on the page concerning the modal. Since the alert needs to be hidden or closed before the user has clicked to sign in, initialize it as false.

```jsx
...
    const [showAlert,setShowAlert]=useState(false);
...
```

Set it to true when the user clicks to sign in inside `handleClose()` of the modal.

```jsx
...
    const handleClose = () =>{
    console.log(email);
```

```
     setShow(false);
     setShowAlert(true);
  }
...
```

Conditionally render the `MyAlert` component and pass in `email` and `showAlert` as props.

```jsx
...
    {showAlert && < MyAlert email={email} showAlert={showAlert} />}
...
```

Initialize `show` as `showAlert` inside the `MyAlert` component so the alert is initially closed and is fired only when the user clicks the sign-in button.

```jsx
...
    const [show, setShow] = useState(showAlert);
...
```

Output `email` inside the alert's body.

```jsx
...
    <p>
          {email} has successfully signed in!
```

```
        </p>
    ...
```

Finally, your `App.js` will look like this:

jsx

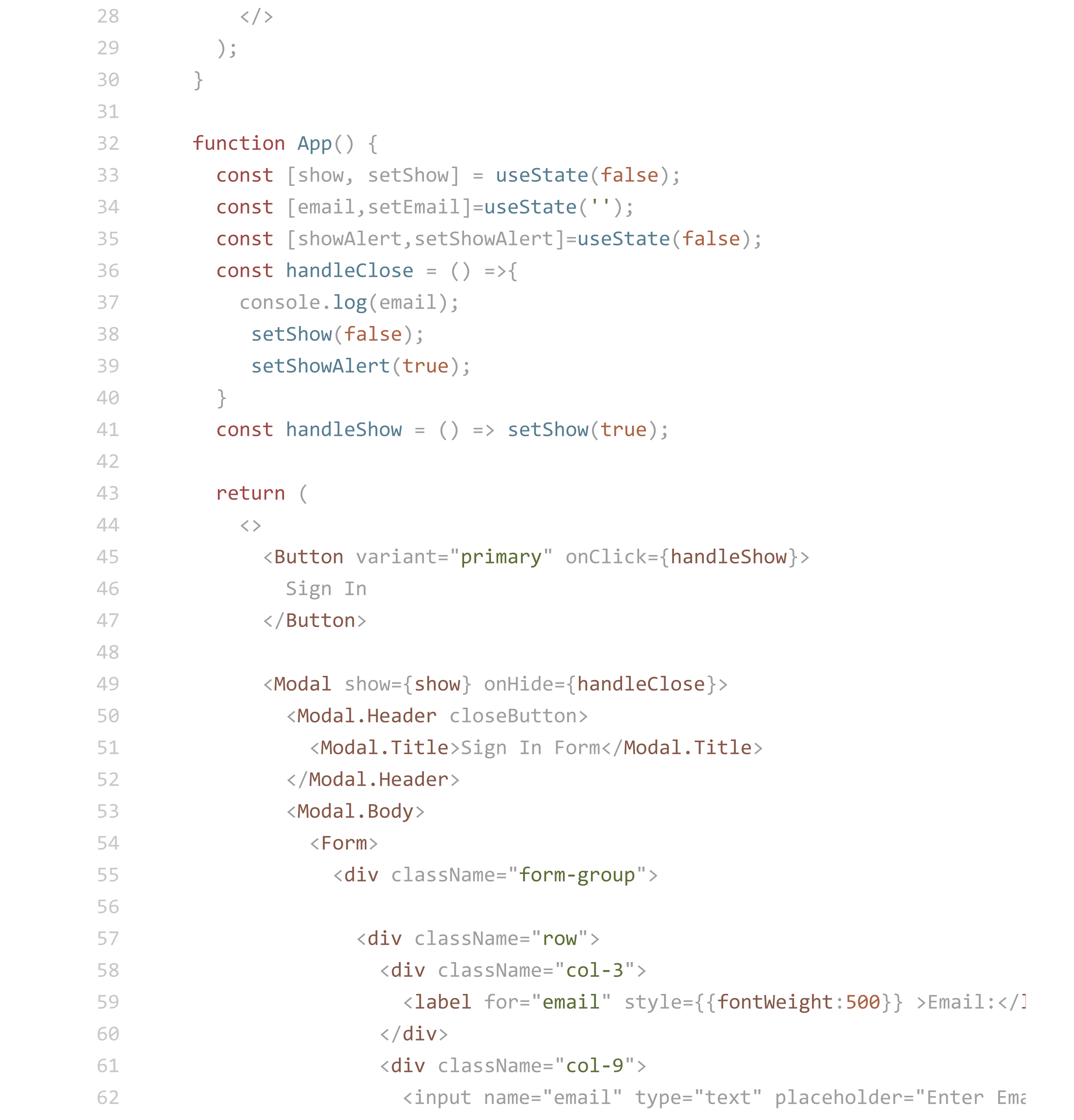
```
import React,{useState} from 'react';
import {Modal, Form} from 'react-bootstrap';
import {Button} from 'react-bootstrap';
import {Alert} from 'react-bootstrap';
import 'bootstrap/dist/css/bootstrap.min.css';



function MyAlert({email,showAlert}) {
  const [show, setShow] = useState(showAlert);

  return (
    <>
      <Alert show={show} variant="success">
        <Alert.Heading>Welcome user :)</Alert.Heading>
        <p>
          {email} has successfully signed in!
        </p>
        <hr />
        <div className="d-flex justify-content-end">
          <Button onClick={() => setShow(false)} variant="outline-succes
            Exit
          </Button>
        </div>
      </Alert>


```



```
    </>
  );
}

function App() {
  const [show, setShow] = useState(false);
  const [email,setEmail]=useState('');
  const [showAlert,setShowAlert]=useState(false);
  const handleClose = () =>{
    console.log(email);
     setShow(false);
     setShowAlert(true);
  }
  const handleShow = () => setShow(true);

  return (
    <>
      <Button variant="primary" onClick={handleShow}>
        Sign In
      </Button>

      <Modal show={show} onHide={handleClose}>
        <Modal.Header closeButton>
          <Modal.Title>Sign In Form</Modal.Title>
        </Modal.Header>
        <Modal.Body>
          <Form>
            <div className="form-group">

              <div className="row">
                <div className="col-3">
                  <label for="email" style={{fontWeight:500}} >Email:</l
                </div>
                <div className="col-9">
                  <input name="email" type="text" placeholder="Enter Ema
```

```
                      }/>
                  </div>
                </div>

              </div>
              <div className="form-group">
                <div className="row">
                <div className="col-3">
                  <label for="password" style={{fontWeight:500}}>Password:
                </div>
                <div className="col-9">
                  <input name="password" type="password" placeholder="Ente

                </div>
                </div>

              </div>
            </Form>
          </Modal.Body>
          <Modal.Footer>
            <Button variant="secondary" onClick={handleClose}>
              Sign Up
            </Button>
            <Button variant="primary" onClick={
              handleClose

            }>
             Sign In Now
            </Button>
          </Modal.Footer>
        </Modal>

       {showAlert && < MyAlert email={email} showAlert={showAlert} />}
      </>
    );
```

```
}

export default App;
```

Now you have a modal that opens a sign-in form connected to your state, which triggers an alert when a user clicks the sign-in button.

Conclusion

In this guide you learned how to use regular React features, such as managing the state, conditionally rendering a component, and passing props to a child component, with React Bootstrap components. React Bootstrap is a great library that provides several other functionalities. You can conveniently mold the utility of any React Bootstrap component to align with your existing app's features.

I hope this guide helped you get started with React Bootstrap. I would also recommend going through the documentation to learn more about available components.