Sign In with React and Firebase

Based on what we learned in the Sign-Up section, we'll implement Sign-In functionality in this lesson.

WE'LL COVER THE FOLLOWING

Sign-Up results in the user getting logged/signed into the app automatically. However, we cannot rely on this mechanism since in some cases, a user could be signed up but not signed in.

To rectify this situation, we will implement the **login** functionality using Firebase. The implementation is pretty similar to the sign-up mechanism and components, so this time we won't have to split it into so many code blocks.

Implementation

We use the code given below in the `src/components/` **`SignIn/index.js`** file:

```
import React, { Component } from 'react';
import { withRouter } from 'react-router-dom';
import { compose } from 'recompose';

import { SignUpLink } from '../SignUp';
import { withFirebase } from '../Firebase';
import * as ROUTES from '../../constants/routes';

const SignInPage = () => (
  <div>
    <h1>SignIn</h1>
    <SignInForm />
    <SignUpLink />
  </div>
);

const INITIAL_STATE = {
  email: '',
  password: '',
  error: null,
};

class SignInFormBase extends Component {

  constructor(props) {
    super(props);

    this.state = { ...INITIAL_STATE };
  }

  onSubmit = event => {
    const { email, password } = this.state;

    this.props.firebase
      .doSignInWithEmailAndPassword(email, password)
      .then(() => {
        this.setState({ ...INITIAL_STATE });
        this.props.history.push(ROUTES.HOME);
      })
      .catch(error => {
        this.setState({ error });
      });

    event.preventDefault();
  };

  onChange = event => {
    this.setState({ [event.target.name]: event.target.value });
  };

  render() {
    const { email, password, error } = this.state;

    const isInvalid = password === '' || email === '';

    return (
      <form onSubmit={this.onSubmit}>
        <input
          name="email"
          value={email}
          onChange={this.onChange}
          type="text"
          placeholder="Email Address"
        />
        <input
          name="password"
          value={password}
          onChange={this.onChange}
          type="password"
          placeholder="Password"
        />
        <button disabled={isInvalid} type="submit">
          Sign In
        </button>

        {error && <p>{error.message}</p>}
      </form>
    );
  }
}

const SignInForm = compose(
  withRouter,
  withFirebase,
)(SignInFormBase);

export default SignInPage;

export { SignInForm };
```

SignIn/index.js

Explanation

The process is similar to that of the sign-up form. Note that the input fields take all the necessary information such as username and password.

The *validation step*, i.e. enabling or disabling the submit button, ensures that the email and password are present in their respective input fields before the request is processed.

The **authentication API** is used once again, but this time, with a function to sign the users *in* rather than **signing** them up. If the sign-in is successful, the local state is updated with the initial state and the user is redirected again. If the sign-in fails, an error object is stored in the local state and an error message appears.

The `SignUpLink` which was defined earlier in the `SignUp` module is used on the sign-in page. Found on the sign-in page, it lets the users sign up if they don't have an account.

Next, we'll run and verify our app to test if our Sign-In component is working.