Provide Firebase in React

Now, we'll learn how to use Firebase in our React application.

WE'LL COVER THE FOLLOWING

In the previous lessons, we created a Firebase class but we have not used it in our React application yet. In this lesson, we'll connect Firebase with the React world.

Different Approaches

The simpler approach would be to create a Firebase instance with the Firebase class and then import the instance (or class) in every React component where it's needed, but that is not a good approach for two reasons:

- It will be more difficult to test our React components.
- Firebase should only be initialized once in our application (singleton) and by exposing the Firebase class to every React component, we could end up with multiple Firebase instances by mistake.

The alternative way is to use React's Context API to provide a **Firebase instance** once at the top-level of our application's component hierarchy.

Create a new `src/components/Firebase/context.js` file in your *Firebase module* and implement the code given below:

```
import React from 'react';

const FirebaseContext = React.createContext(null);

export default FirebaseContext;
```

Firebase/context.js

The `createContext()` function essentially creates two components:

1. The `FirebaseContext.Provider` component: It is used to provide a Firebase instance once at the top-level of our React component tree, which we will do in this lesson
2. The `FirebaseContext.Consumer` component: It is used to retrieve the Firebase instance if needed in the React component.

For a well-encapsulated Firebase module, we create an `index.js` file in our Firebase folder that exports all necessary functionalities e.g. Firebase class, Firebase context (for Consumer), as well as Provider components.

```
import FirebaseContext from './context';
import Firebase from './firebase';

export default Firebase;

export { FirebaseContext };
```

Firebase/index.js

Purpose of the Firebase Context

The **Firebase Context**, located in the Firebase module (folder), is used to provide a *Firebase instance* to the entire application in the `src/index.js` file. We only create the Firebase instance using the Firebase class and pass it as a *value prop* to React's Context as shown in the code snippet below:

```
import React from 'react';
import ReactDOM from 'react-dom';

import './index.css';
import * as serviceWorker from './serviceWorker';

import App from './components/App';
import Firebase, { FirebaseContext } from './components/Firebase';

ReactDOM.render(
  <FirebaseContext.Provider value={new Firebase()}>
    <App />
  </FirebaseContext.Provider>,
  document.getElementById('root'),
);

serviceWorker.unregister();
```

src/index.js

By following this procedure, we can be sure that Firebase is only instantiated once and is injected via React's Context API to React's component tree. So if any component is interested in using Firebase, it has access to the Firebase instance through the `FirebaseContext.Consumer` component. Having said that, we will see it being used first-hand later in this application, but the following code snippet gives the basic idea of how it would work:

```
import React from 'react';

import  { FirebaseContext } from '../Firebase';

const SomeComponent = () => (
  <FirebaseContext.Consumer>
    {firebase => {
      return <div>I've access to Firebase and render something.</div>;
    }}
  </FirebaseContext.Consumer>
);

export default SomeComponent;
```

Example

We have now connected Firebase and React, which is the fundamental step to make the layers communicate with each other.

In the next lesson, you can run and verify the application after the addition of Firebase in it.