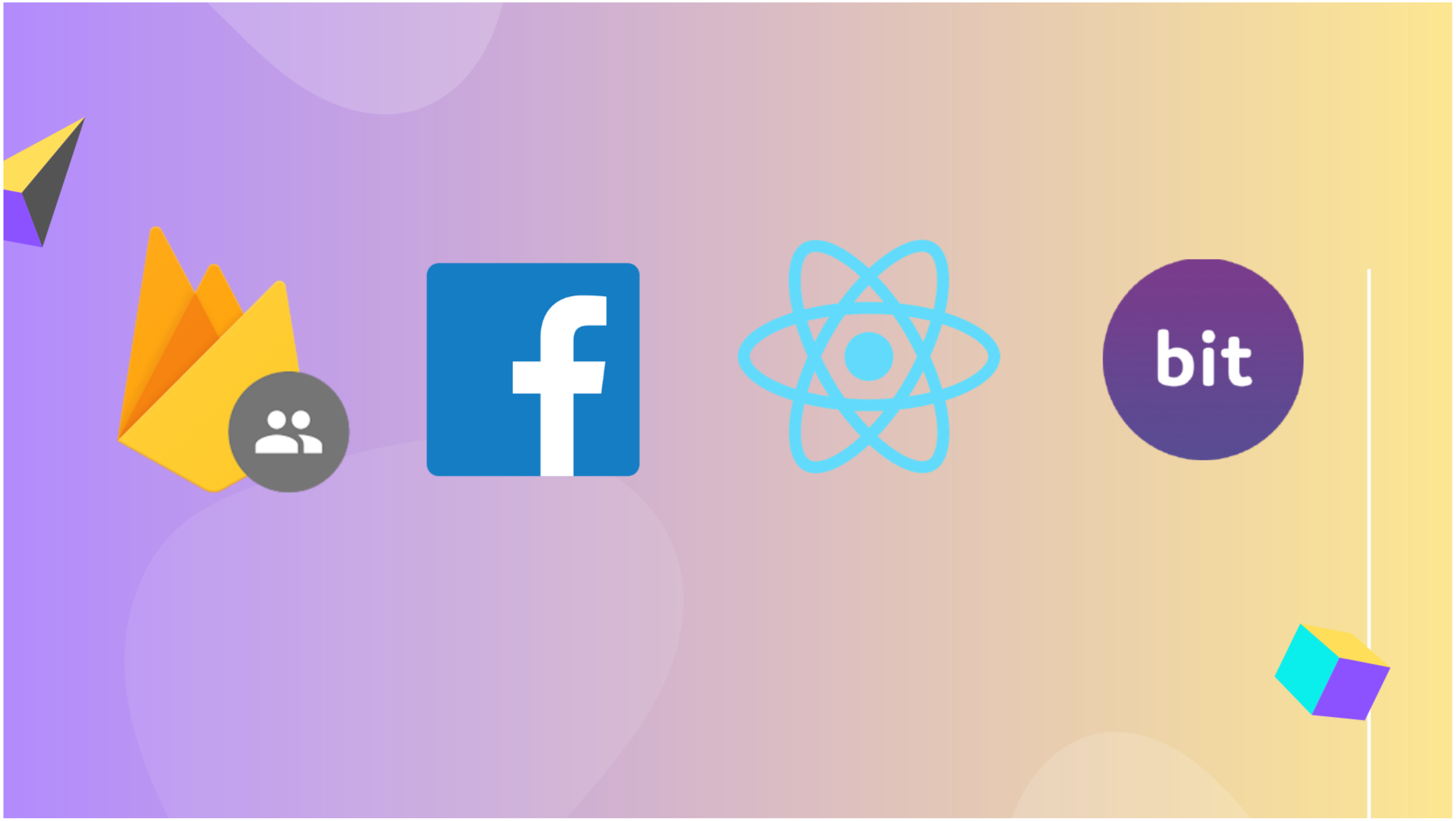
Building a Reusable Firebase Facebook Login Component

Learn how to build a reusable and shareable Firebase Facebook login component

Krissanawat Kaewsanmuang Follow
Jul 2, 2019 · 8 min read



In this tutorial, we are following the first chapter of clone Firebaseweb-UI demo and use codes from it as well. Here, we are going to learn how to create a reusable and shareable Facebook login component with React and Bit. First, we will prepare a facebook login component using firebase integration and authentication in a React project. Then, we are will push the code to Bit. We will also learn to import modules from packages installed using the Node Package Manager(NPM). We have broken down this tutorial into various sub-headings for you to grasp the concept and work accordingly.

What you'll learn …?

- How to create new React app.
- How to install and using firebase in React.
- How to use Firebase Authentication.
- How to create and set up a Facebook app.
- How to install and use Bit.
- How to pass props in React.

Problem

Firebaseweb-UI demo built using vanillaJS is not split up into component which makes it hard for maintenance purpose. Since the components are not used, the codes may be re-written in most parts creating a heavy form of project.

Requirements

Here's a complete list of plugins, packages, and services you're going to need in order to gain something from this tutorial:

- Nodejs `v8.x.x` or higher installed along with NPM/yarn.
- Firebase, Bit and Facebook developer account.
- Already installed React project.
- Firebase and Bit package.

First Step

Before working on the Facebook login component, we are going to complete every configuration required while using *App.js*.

Firstly, try to grab CSS from Firebaseweb-UI demo and paste from

firebase-ui.css -> **App.css**

Here, we need to include firebase CSS to *App.js*.

Then include some styles to *Index.css,*

style.css -> **Index.css**

which we need to include in the *index.js* file.

Then, we need to grab the button code from firebaseweb-ui and then add it to our *App.js* code as follows:

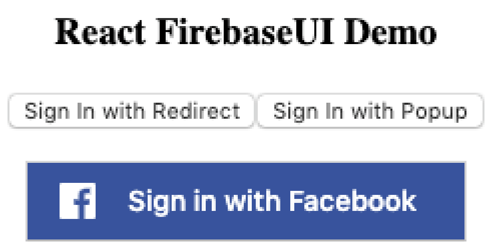
```
render() {
    return (
      <div id="container">
        <h3>React FirebaseUI Demo</h3>
        <div id="firebaseui-spa">
          <p>
            <button id="sign-in-with-redirect">Sign In with Redirect</button>
            <button id="sign-in-with-popup">Sign In with Popup</button>
          </p>
          <div id="firebaseui-container">
            <div id="firebaseui-container firebaseui-page-provider-sign-in firebaseui-id-page-pr
            <div id="firebaseui-card-content">
              <ul class="firebaseui-idp-list">
```

```
                <li class="firebaseui-list-item">
                  <button class="firebaseui-idp-button mdl-button mdl-js-button mdl-button--rais
                    <span class="firebaseui-idp-icon-wrapper">
                    <img class="firebaseui-idp-icon" alt="" src="https://www.gstatic.com/firebas
                    <span class="firebaseui-idp-text firebaseui-idp-text-long">Sign in with Face
                    <span class="firebaseui-idp-text firebaseui-idp-text-short">Facebook</span>
                  </button>
                </li>
              </ul>
            </div>
          </div>
        </div>
      </div>

    );
  }
```

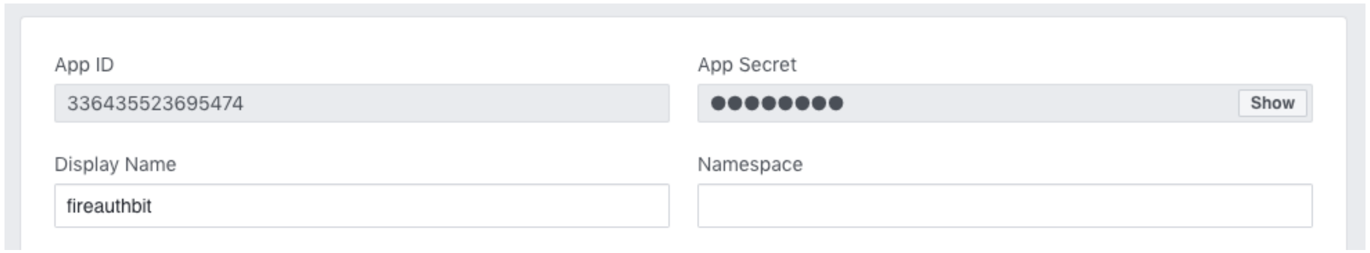
facebooklogin.js hosted with ❤ by GitHub view raw

Now, in the browser, we can see the facebook login component button. The screenshot of the result from the above code snippet is shown below:



Second Step: create and setup Facebook App and Firebase

In this step, we are integrating facebook login with firebase. You will need to create a facebook app from your facebook developer account which will include APP_ID and APP_SECRET key. Then, open your facebook developer account and copy APP_ID and APP_SECRET. After that, we need to paste it on firebase console as shown in the screenshot below:



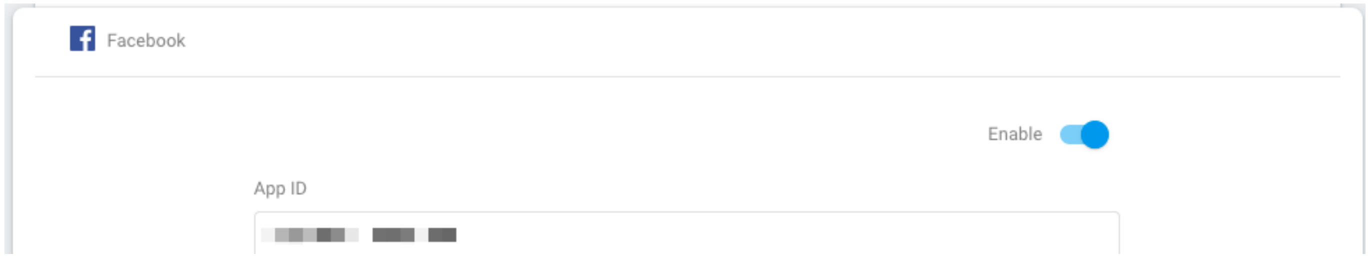
App ID: 336435523695474

App Secret: ●●●●●●●● Show

Display Name: fireauthbit

Namespace:

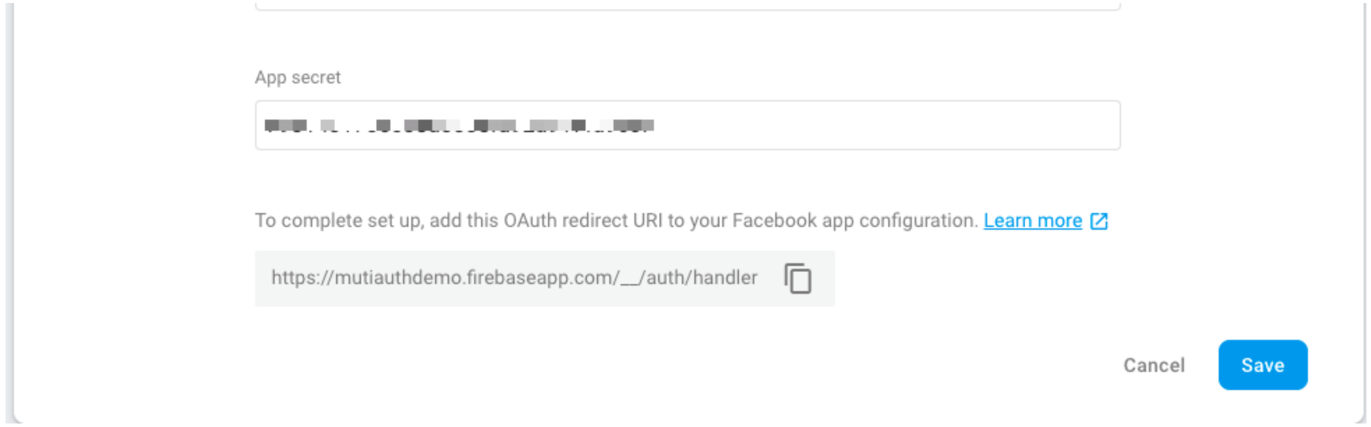
Then, we need to open the firebase console and paste APP_ID and APP_SECRET key on authentication sign-in method tab as shown in the screenshot below:



Facebook

Enable

App ID: [redacted]



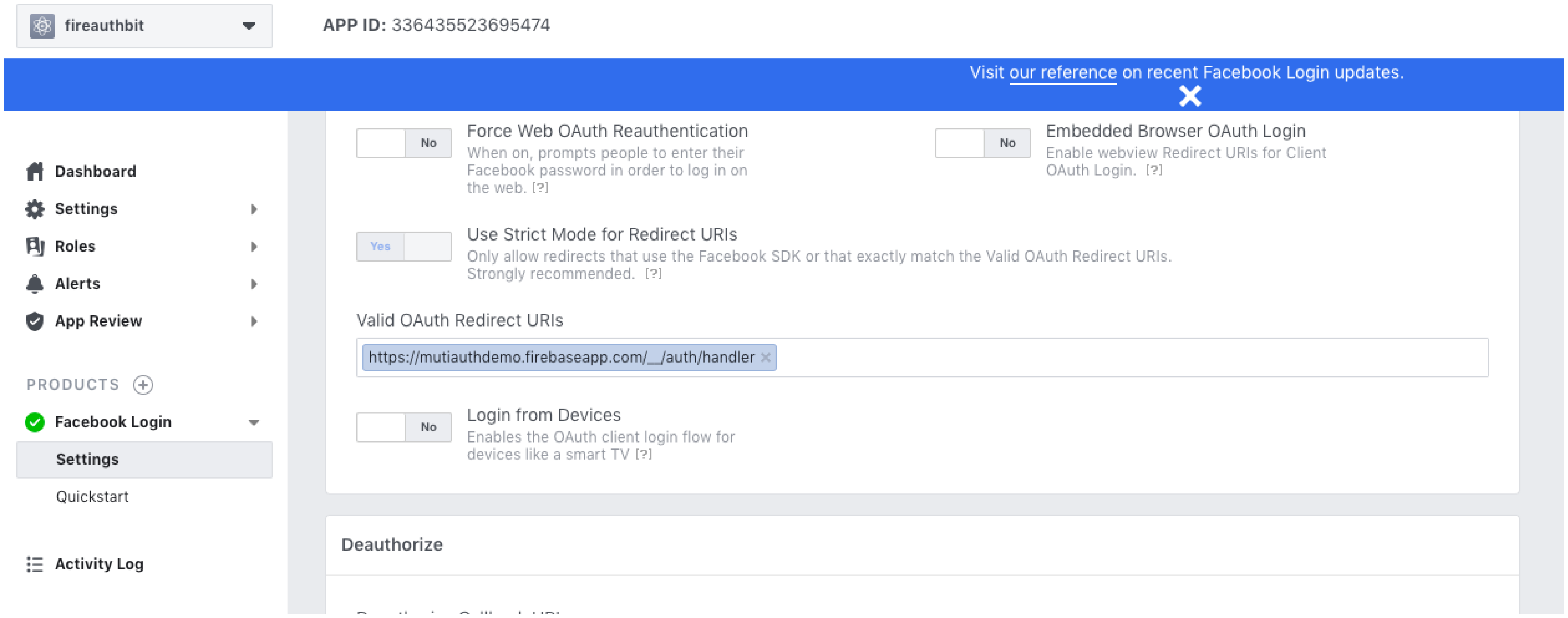
App secret

[redacted]

To complete set up, add this OAuth redirect URI to your Facebook app configuration. Learn more

https://mutiauthdemo.firebaseapp.com/__/auth/handler

After that, you will see a redirect URI at the bottom as shown in the screenshot above. We need to copy that redirect URI. Then, return to the Facebook developer console. In the developer console, we need to go to Facebook login setting and scroll to Valid OAuth Redirect and paste the URI there as shown in the screenshot below:


fireauthbit

APP ID: 336435523695474

Visit our reference on recent Facebook Login updates.

No — Force Web OAuth Reauthentication
When on, prompts people to enter their Facebook password in order to log in on the web. [?]

No — Embedded Browser OAuth Login
Enable webview Redirect URIs for Client OAuth Login. [?]

Yes — Use Strict Mode for Redirect URIs
Only allow redirects that use the Facebook SDK or that exactly match the Valid OAuth Redirect URIs. Strongly recommended. [?]

Valid OAuth Redirect URIs
https://mutiauthdemo.firebaseapp.com/__/auth/handler

No — Login from Devices
Enables the OAuth client login flow for devices like a smart TV [?]

Deauthorize

Then, just save the configuration.

In the next step, we need to create a Firebase app. Here, we need to get firebase configuration and paste it on our App.js file. The configuration for the firebase is given below:

```
const firebaseConfig = {
  apiKey: "xxxxxxxxxxx",
  authDomain: "xxxxxxxxxxx.firebaseapp.com",
  databaseURL: "https://xxxxxxxxxxx.firebaseio.com",
  projectId: "xxxxxxxxxxx",
  storageBucket: "xxxxxxxxxxx.appspot.com",
  messagingSenderId: "xxxxxxxxxxx",
  appId: "1:14017xxxxxxxxxxx382be09e",
};
```

Now, we need to import *firebase* from firebase/app package and firebase/auth package which is necessary to configure other modules and firebase and its authentication configuration.

```
import firebase from "firebase/app";
import "firebase/auth";
```

Now, we need to create a constructor that handles initialize firebase as shown in the code snippet below:

```
constructor(props) {
    super(props);

    if (!firebase.apps.length) {
      firebase.initializeApp(firebaseConfig);
```

```
  }
 }
```

Then, we need to create a new function to trigger Firebase Authentication with the Facebook code shown below which you can find on Firebase doc.

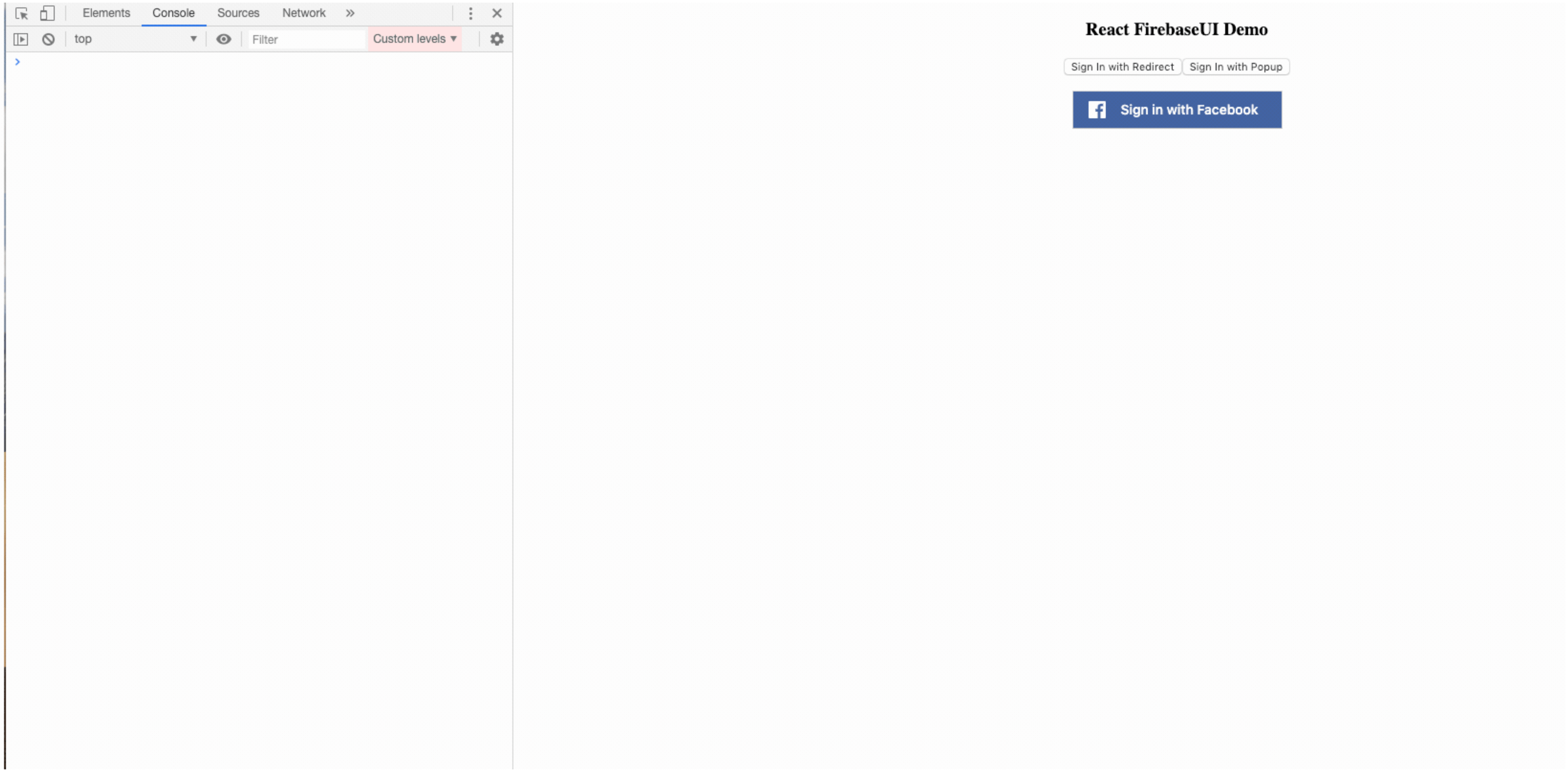
```
handleLogin() {
  var provider = new firebase.auth.FacebookAuthProvider();
  firebase
   .auth()
   .signInWithPopup(provider)
   .then(function(result) {
    // This gives you a Facebook Access Token. You can use it to
access the Facebook API.
    var token = result.credential.accessToken;
    // The signed-in user info.
    var user = result.user;
    console.log(user);
    // ...
   })
   .catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // The email of the user's account used.
    var email = error.email;
    // The firebase.auth.AuthCredential type that was used.
    var credential = error.credential;
    // ...
   });
 }
```

Third Step: let's make it work

In this step, we are going to make everything work and configured properly. First, add a `handleLogin` button to handle to trigger a *handleLogin* function as shown in the code snippet below:

```
<button onClick={this.handleLogin} ... />
```

Now, test the login operation which can be seen in the simulation below:



Hence, you can see that it works like a charm.

Now, we need to make it better using Bit.

Make it Reusable and Sharable with Bit

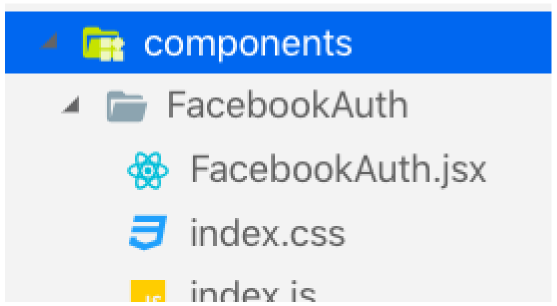
Share reusable code components as a team · Bit

Easily share reusable components between projects and applications to build faster as a team. Collaborate to develop…

bit.de

To make our code reusable and shareable, we'll do some restructuring and move all our code to 'components' (this is not mandatory but a better practice, when using Bit). Then, we'll export it (with all its dependencies), to a component collection on Bit's cloud, to be shared with others and easily reused.

Now, we structure our component as shown in the project folder structure below:



```
components
  FacebookAuth
    FacebookAuth.jsx
    index.css
    index.js
```

Now, we need to move all code from *App.js* to *FacebookAuth.jsx* file. The code is shown in the code snippet below:

Then we need to change three things which are mentioned below:

1. First, we need to change the class name to FacebookAuth and export default.
2. Second, we need to move CSS from *App.css* to components/FacebookAuth/index.css file.
3. Finally, we need to use props from a parent to pass configuration values to components.

Then, use the props value of firebaseConfig to initialize the firebase project as shown in the code snippet below:

```
firebase.initializeApp(this.props.firebaseConfig);
```

And in the index.js file, we will wrap it to the component as shown in the code snippet below:

Finally, we need to change the *App.js* file to include the component and pass firebase configuration as shown in the code snippet below:

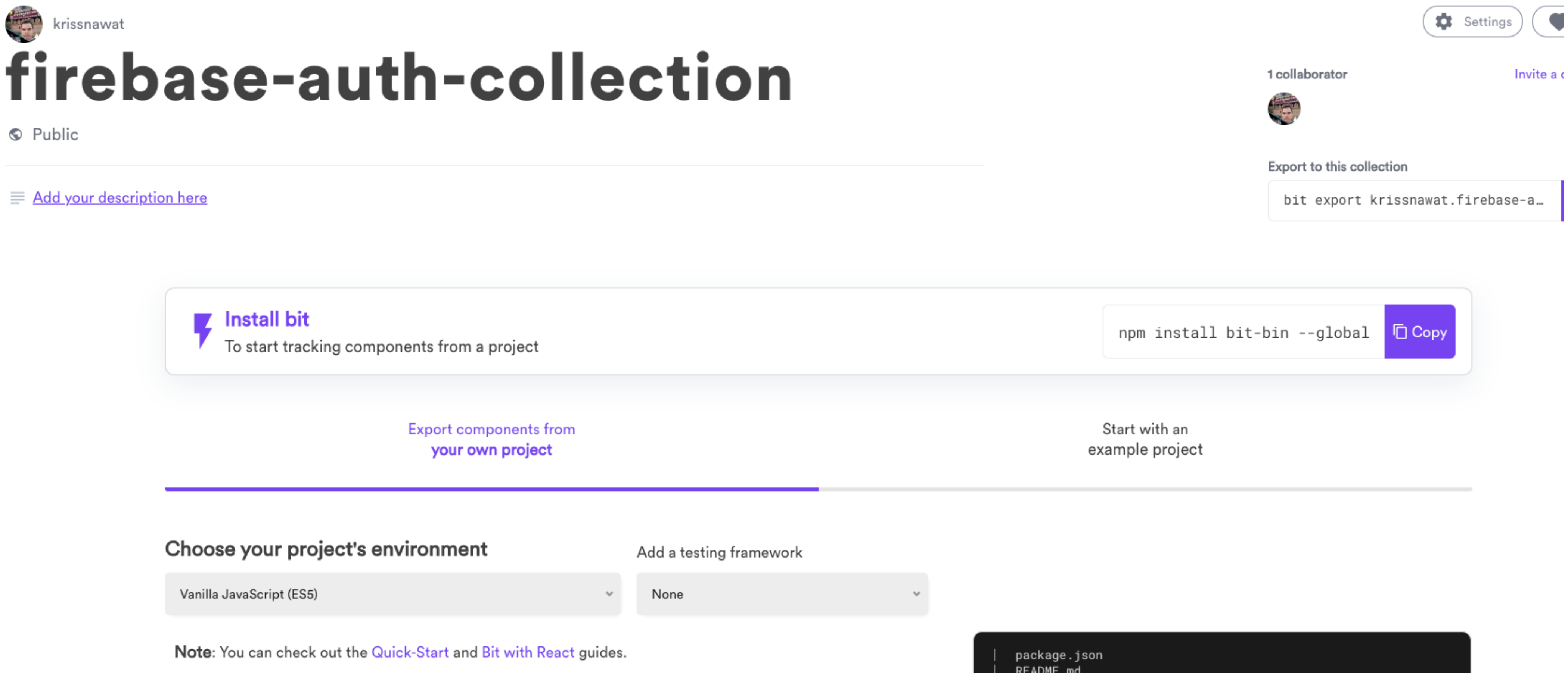
The work to create a component for facebook login with firebase configuration props is done.

Now, let's move on to a Bit part and get started on Bit console to create a new collection.

1. Initialize Bit on a project

Here, we create a new collection on Bit to which we are going to push all our component code.

To successfully push the whole project to Bit, you need to follow the instructions given on the page shown below:


krissnawat

firebase-auth-collection

Public

Add your description here

Settings
1 collaborator
Invite a
Export to this collection
Install bit
To start tracking components from a project
Copy
Export components from your own project
Start with an example project
Choose your project's environment
Add a testing framework

```
bit export krissnawat.firebase-a…
```

```
npm install bit-bin --global
```

Choose your project's environment: Vanilla JavaScript (ES5)

Add a testing framework: None

Note: You can check out the Quick-Start and Bit with React guides.

```
| package.json
| README.md
```

First, we need to install Bit CLI

> *Using NPM,*
>
> *npm install bit-bin --global*

Then, we need to go to the project directory in our Bit CLI and perform the commands `bit init` and run `bit login` on Bit CLI as shown in the screenshot below:

2. Track components

Here, we are going to stage our component to Bit repository we created. We do this by using the command `bit add src/components/FacebookAuth`

The result after the command is performed is shown below in the screenshot:

4

3. Configure a React compiler for our component

When we configure a compiler we tell Bit to capsule the component with it. Capsuling components together with their compilers gives us the freedom to use, build and test them anywhere. This includes being able to run the code in any of the applications we'd like to use the component in, as well as running it in the cloud to enable features such as the live component playground.

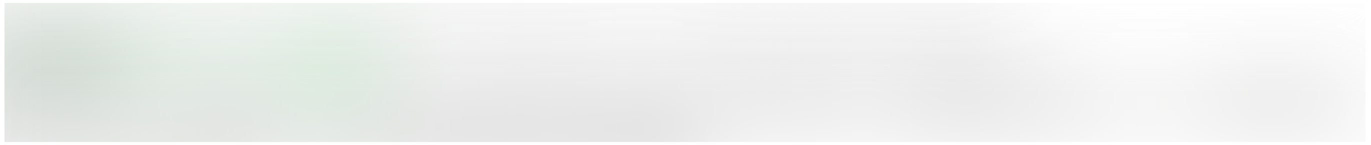
```
bit import bit.envs/compilers/react --compiler
```

4. Tag and export

Here, we are going to set a version to all tracked components, and export to this collection using following bit command

```
bit tag --all 1.0.0
```

The result of the command is shown in the following screenshot:



Finally, we need to perform following bit command to push our component code to bit collection that we created before:

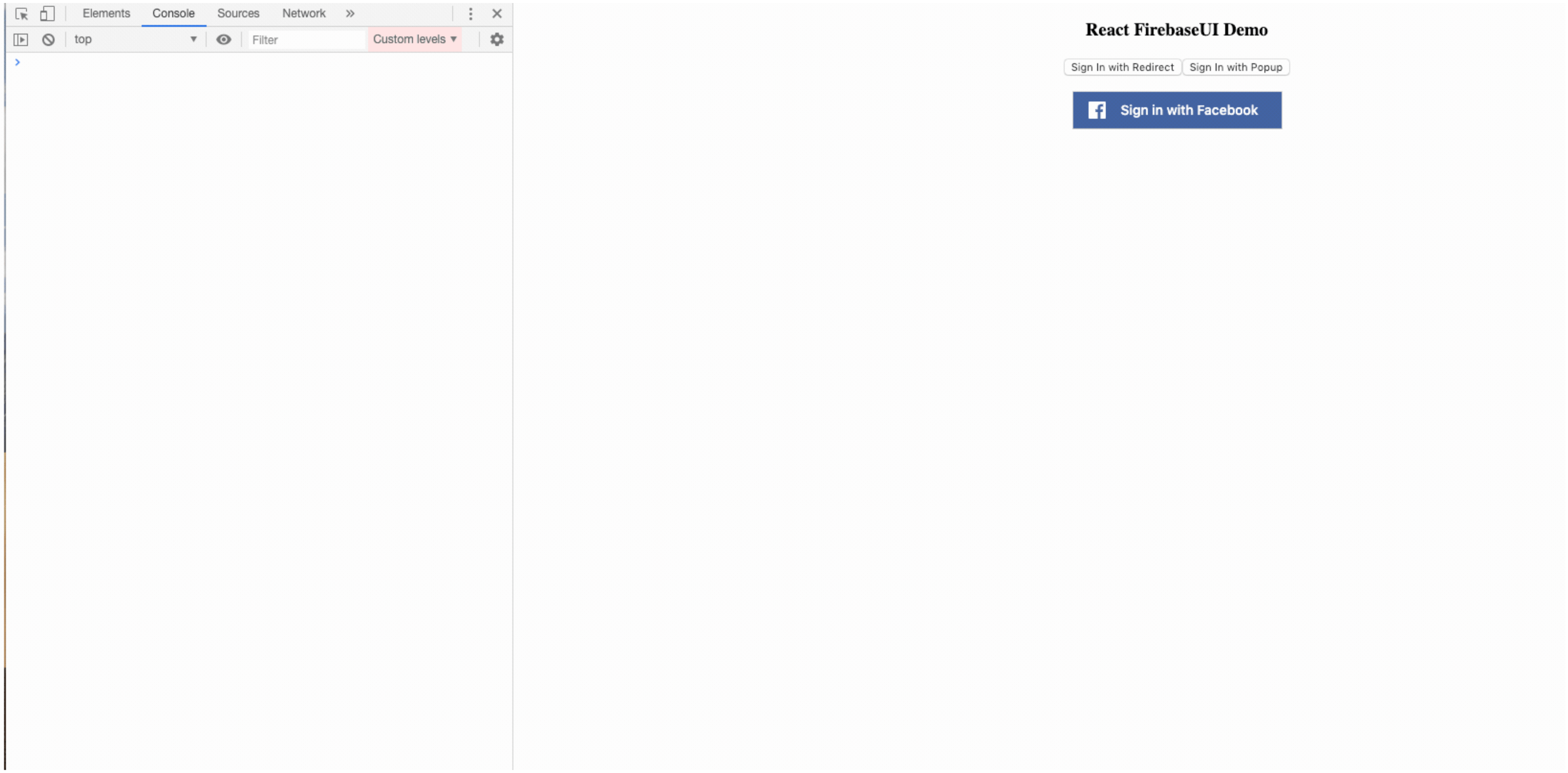
```
bit export krissnawat.firebase-auth-collection
```

Now, our package is live on Bit. You can either install it using NPM or Yarn or, you can further develop it (get its source code) using `$ bit import` (on an initialized Bit workspace). The code to do that is shown below:

```
  npm i @bit/krissnawat.firebase-auth-collection.facebook-auth
```

Lastly, we need to replace the component in our *App.js* file as shown in the code snippet below:

Finally, all our work is complete and you can run the project locally to test everything. When you run the project locally you will see the result shown in the simulation below:



Conclusion

In this tutorial, we learned how to create Facebook and Firebase app and register facebook app on Firebase. This is useful for integrating facebook login in many other projects as well. Then, we learned how to make them work together and make the code better by organizing them to component base. Components make our code simpler, easy, reusable and clean. Lastly, we made our facebook login component shareable using Bit. For the next tutorial, we will learn about building Google login component.

Learn More

How to Share React UI Components between Projects and Apps

A simple guide to help you organize, share and sync React components between your team's apps.

blog.bitsrc.io

11 React UI Component Libraries you Should Know in 2019

11 React component libraries with great components for building your next app's UI interface in 2019.

blog.bitsrc.io

5 Ways to Style React Components in 2019

Making sense of styling React components in 2019 in a short yet detailed review

blog.bitsrc.io

JavaScript React Firebase Frontend Programming