



Open in app

Get started



Published in JavaScript in Plain English

You have **2** free member-only stories left this month. [Sign up for Medium and get an extra one](#)



Chad Murobayashi

Follow

May 10, 2021 · 5 min read ★ · [Listen](#)

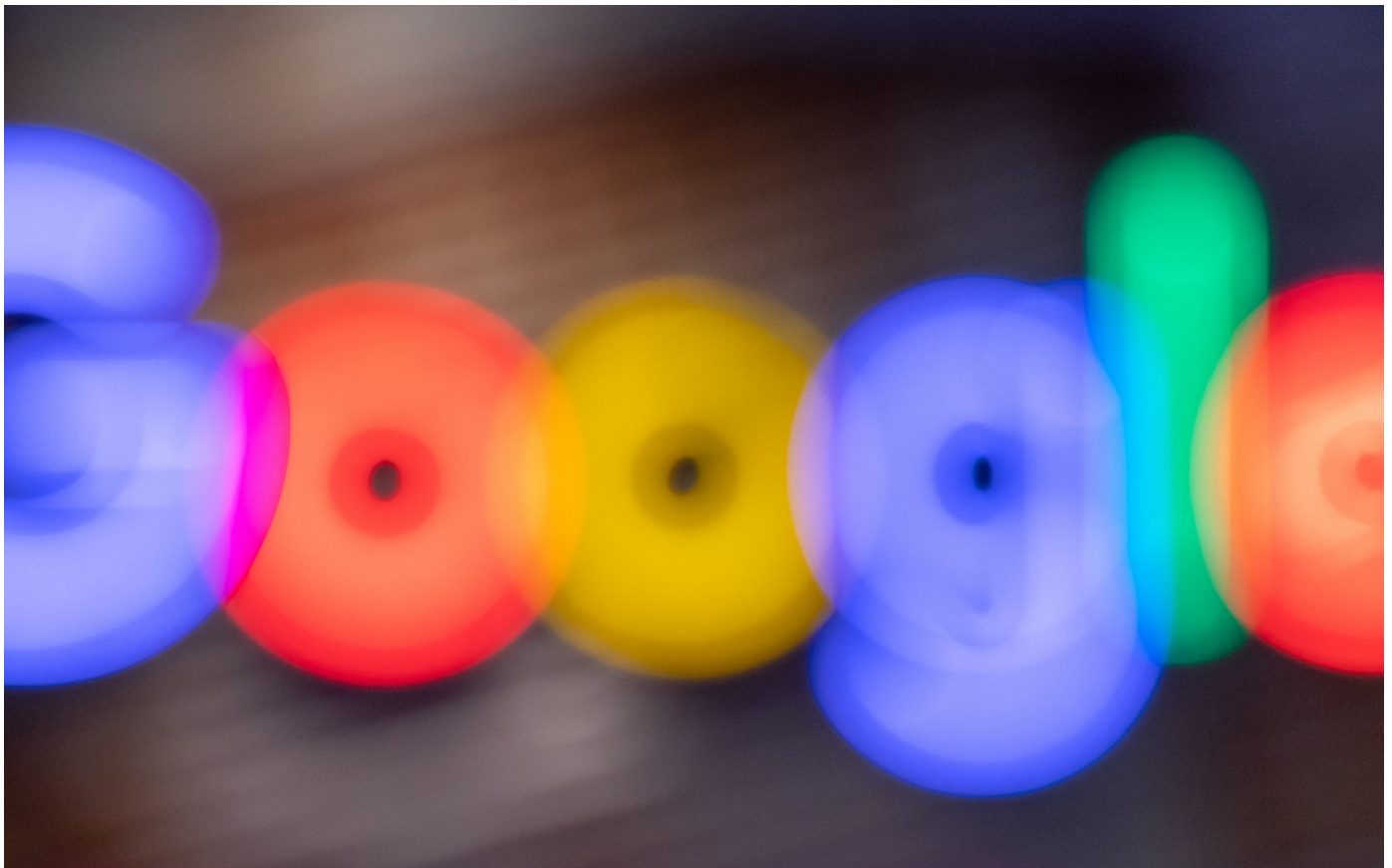


Save



How to Set Up Google OAuth in React with React-Google-Login

Create a Google OAuth login component in React.



[Open in app](#)[Get started](#)

OAuth is a common way for users to give websites and applications access to their information without creating a new password. Most large companies use OAuth to share information with third-party websites and applications. Google is one of these large companies that use the OAuth protocol for authentication and authorization.

Understanding the flow to implement OAuth in a React application can be tricky. Luckily, there is a third-party package called `react-google-login` that we can use to make things a little easier.

In this article, we will take a look at how to access Google APIs OAuth in a React application. By the end of the article you should have an understanding of how to do the following:

- Create a login button in React using `react-google-login`
- Obtain credentials from Google API console
- Obtain an access token from the Google authorization server

Create a login button in React using `react-google-login`

For this example, I will create a basic React application using `create-react-app`. Then we will clean up the `App.js` file to render a blank screen.

```
npx create-react-app YOUR-PROJECT-NAME
```

We will use the `react-google-login` package to create a button component and connect to the Google authorization server.

The first step will be to install the package in your project.



[Open in app](#)[Get started](#)

Then, import the `GoogleLogin` component, and render it in your app. The `GoogleLogin` component will take the following props:

- `clientId` — Client ID from Google API console (we will get this in the next section)
- `buttonText` — Text to be displayed on the button
- `onSuccess` — Callback function that is run on success
- `onFailure` — Callback function that is run on failure
- `cookiePolicy` — The domains for which to create sign-in cookies

For the `onSuccess` and `onFailure` props, we will create a callback function that simply prints the response to the console.

```
const responseGoogle = response => {  
  console.log(response);  
};
```

The `GoogleLogin` component will look like the following:

```
<GoogleLogin  
  clientId=""  
  buttonText="Login with Google"  
  onSuccess={responseGoogle}  
  onFailure={responseGoogle}  
  cookiePolicy="single_host_origin"  
>
```

The button will be disabled for now since we haven't added a client ID yet. It will look something like this.





Open in app

Get started



Login with Google

Obtain credentials from Google API console

The next step will be to obtain credentials from the [Google API console](#).

Visit the website, and create a new project.

New Project



You have 13 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name *

React Google Login



Project ID: **credible-flag-313108**. It cannot be changed later. [EDIT](#)

Location *



No organization

[BROWSE](#)

Parent organization or folder





[Open in app](#)

[Get started](#)

Once your project is created, select your project and visit the **OAuth consent screen**.

API APIs & Services



Dashboard



Library



Credentials



OAuth consent screen



Domain verification



Page usage agreements

Here, you will need to configure and register your app. Take the following steps:

1. Select User Type = External





Open in app

Get started

User Type

☐ Internal 

Only available to users within your organization. You will not need to submit your app for verification. [Learn more](#)

☒ External 

Available to any test user with a Google Account. Your app will start in testing mode and will only be available to users you add to the list of test users. Once your app is ready to push to production, you may need to verify your app. [Learn more](#)

CREATE

2. Fill out the required fields in App Information and Developer Contact Information



[Open in app](#)[Get started](#)

App information

This shows in the consent screen, and helps end users know who you are and contact you

App name *

React Google Login

The name of the app asking for consent

User support email *

For users to contact you with questions about their consent

App logo

[BROWSE](#)

Upload an image, not larger than 1MB on the consent screen that will help users recognize your app. Allowed image formats are JPG, PNG, and BMP. Logos should be square and 120px by 120px for the best results.

3. Add scopes or test users (optional)

Once this is complete, visit the **Credentials** tab.



[Open in app](#)[Get started](#)

API APIs & Services



Dashboard



Library



Credentials



OAuth consent screen



Domain verification



Page usage agreements

From here, click the **Create Credentials** button and select **OAuth Client ID**. Select an Application type of Web application and give the OAuth client a name. Finally, add an Authorized JavaScript origin.

The authorized JavaScript origin is used to identify the domains from which your application can send requests to the OAuth 2.0 server. In our example, we will set it as <http://localhost:3000>.



[Open in app](#)[Get started](#)

Create OAuth client ID

Application type *

Web application ▼

[Learn more](#) about OAuth client types

Name *

React Google Login

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.



The domains of the URIs you add below will be automatically added to your [OAuth consent screen](#) as [authorized domains](#).

Authorized JavaScript origins ?

For use with requests from a browser

URIs *

http://localhost:3000



[+ ADD URI](#)

Finally, click **Create**, and wait for the OAuth client to be created. The following modal will appear.



[Open in app](#)[Get started](#)


OAuth client created

The client ID and secret can always be accessed from Credentials in APIs & Services



OAuth is limited to 100 [sensitive scope logins](#) until the [OAuth consent screen](#) is verified. This may require a verification process that can take several days.

Your Client ID

51829459292-vk2eiebsnl87t9fcfq18rg3n4eprg944.apps.goc 

Copy this client ID. We will need it in the next step.

Obtain an access token from the Google authorization server

Now that we have the client ID, go back to your text editor and add it as a prop for the `GoogleLogin` component.

In your app, the login button should now be shown as clickable.



Login with Google



[Open in app](#)[Get started](#)

Once you are signed in, the `onSuccess` callback function will be called and you will see an object with the response information in your console. The response will include an access token, a profile object with your personal information, and other details.

Congratulations! You are now authenticated through Google.

Conclusion

Thanks for reading! Using OAuth is a good way to authenticate users in your website or application. It saves users the hassle of having to create and remember a new password.

I hope this article was helpful for you to set up Google OAuth in your React application.

If you want to learn how to build a signup page in React, check out the article below.

Create a Signup Page with React and Material-UI

Learn how to create a dialog and form to use for your next website

levelup.gitconnected.com

More content at plainenglish.io





Open in app

Get started

