OAuth Documentation

Most applications need authentication. Authentication is the process during which a person or computer is checked on whether they are who they are constructing to be. Getting authentication right is extremely important (during some cases more than others), because valuable user data, or other types of data, are at stake.

That’s why it’s usually better to use an external authentication service, rather than building your own, because it most likely won’t be as secure. A lot of frameworks have their own authentication scaffold and big platforms have their own authentication service. I decided to work with OAuth 2 to access Google accounts.

# Registering your app

Your app needs to be registered at Google. Or else you won’t be able to login using your Google-account. Go to <https://console.cloud.google.com/> and login with an existing account or create a new one.

Next is creating a new project, give it whatever name you like, preferable the same name your app has (if it has one).

* Open the project you just created.
* Open the side panel on the left side.
* Open the “APIs & Services” tab.
* Click on “Credentials”.

Here you can create and view your credentials. Some services need credentials or API keys to be accessed. Go ahead and click on “Create Credentials” and click on “OAuth Client ID”. I’m created a React app, which is a webapp, so for “Application type” I will be selecting “Web application”. Each app type has it’s own keys or data you need to provide.

In the case of web applications, you need to provide “Authorized JavaScript origins”. These are URIs on which your app will be running, you need to specify these for your app to be able to use the authentication. Usually React apps run at <http://localhost:3000>, so go ahead and put that in. Click create and your credentials should be created and Google will show you your Client ID and Client Secret, you may need these for specific usecases.

You also need to add some testusers for your app to login with, if you don’t add a Google Account as a testuser, you won’t be able to login with that account. While you’re still in the same project:

* Click on “OAuth Consent Screen”, again under “APIs & Services”
* Under “Test users”, click on “Add Users”
* Add the accounts you would like to test.

# React component

We’re going to be using the “useGoogleLogin” method to authorize the user, to install Google OAuth use the following command:



To actually start authorizing, you need a button to click on, for which we will be using the “GoogleButton”, go ahead and install that:



Create a new component or implement this in a page, up to you. We’re going to call the “useGoogleLogin” method and as a parameter, pass an object. That object is going to contain two methods, an “onSuccess” method and an “onError” method. The “OnError” method should display some error, but that’s not the star of the show.

The real star is the “onSuccess” method, which is asynchronous. Which receives a response called “tokenResponse” which should contain an access token, that is a token you can use to access different Google apis.

Next we’re going to make a call to the following url <https://www.googleapis.com/oauth2/v3/userinfo>, along with a header called “Authorization: Bearer”, which is going to contain the access token we received from that response. If you would like to read more about this part of logging in, feel free to read the following article:

<https://www.oauth.com/oauth2-servers/signing-in-with-google/verifying-the-user-info/>.

If everything went fine, you should receive an object with user data, do whatever you’d like with it.



But don’t forget the button! We’re going to use the button and give it an “onClick” property, to which we’ll pas the method above:



# Google OAuth Provider

One last thing to do, I’m serious! To use the Google authentication that we just built, it has to be wrapped in a “GoogleOAuthProvider” component. No need to install it, because it’s part of the same package that the “useGoogleLogin” method is in.

So if you’re separating the provider and the login in different components, go ahead and import the component on top of the component your provider will be in:



If you’re putting those in the same component, just edit your existing import-statement, from:



To:



Next is to wrap your authentication code/component with the “GoogleOAuthProvider”. I put my code in a separate component called “LoginButton”, so I’ll be wrapping that:



Remember the Client ID Google gave us back? Pass that as a prop to the “GoogleOAuthProvider” component as a final step.

If you followed along everything should work now!