Firebase Auth w/ Google Sign-In in Chrome Extensions

Contents

This sample demonstrates how to authorize a user with Firebase in a Chrome extension using Google Sign-In and setup the Chrome extension to allow the use of the Realtime Database and Firebase Storage.

Introduction

- Read more about Firebase Auth

Setting up this sample

Creating a dummy Chrome extension (how to obtain a Chrome App ID and Public Key)

Setting up authentication in Chrome extensions is a bit of a chicken vs. egg problem. You need the **Public Key** and **Chrome App ID** to configure your manifest.json file, but you have to publish that to the Chrome Store to get the values. So we will publish a dummy app in order to obtain these.

```
  "permissions": [
    "identity"
  ]
}
```

Zip the directory:

```
cd ..
zip -r <chrome ext name>.zip <directory name>
```

Upload the dummy extension to the app store by going to the Chrome App Developer Dashboard and clicking **Add a New Item**. For more reading on publishing to the Chrome Web Store, go here.

> Of course if you already own an extension and would like to add Firebase to it, skip the steps above.

Once the extension is uploaded and visible under **Your Listings**, click the **more info** link to the right. Copy down both the **Item ID** (e.g. `kjdohecpbfnjbinmakjcmpgpbbhhijgf`) and **Public Key** (e.g. `MIIBIjANBgkqhkiG9w0B...long string of text...unbQIDAQAB`). You will need both of these below.

Creating an OAuth Client

- Create a new **OAuth Client ID** in your project's Developers Console (Click this link and select your Firebase project).
- Select **Chrome App** and enter your Chrome Extension/App ID (the `Item ID` obtained above).
- Note the `Client ID` (e.g. `7159....j00.apps.googleusercontent.com`) as you will need this below.

Configuring your Firebase Project

- Create or select a Firebase project at Firebase Console.

- Edit the `manifest.json`
  - Enter your **OAuth Client ID** and your extension's **Public Key**.
  - Remove all comment lines (starting with `//`) in the `manifest.json` file before deploying your extension online.
- Install the Extension in your browser and click on the extension's icon once installed. The first time your users will install the extension they will have to authorize Firebase using the login button.

Using Firebase in your own extension

The keys to using Firebase in a Chrome extension are:

- Because of Chrome Extensions' Content Security Policy you need to avoid inline JavaScript in your HTML pages so you need to add the Firebase initialisation snippet in your JS file instead of inside the HTML file as we typically instruct. The Firebase initialisation snippet looks like this:

```
// Initialize Firebase
var config = {
 apiKey: "<qwertyuiopasdfghjklzxcvbnm>",
 databaseURL: "https://<my-app-id>.firebaseio.com",
 storageBucket: "<my-app-id>.appspot.com"
};
firebase.initializeApp(config);
```

- Create a Google Client ID that's authorized for your Chrome extension and whitelist it in your Firebase project:
  - Create a new OAuth Client ID in your project's Developers Console, Select **Chrome App** and enter your Chrome Extension/App ID.
  - In your project's Firebase Console, enable the **Google** authentication method in the **Auth** section > **SIGN IN METHOD** tab.

Documentation

GUIDES GITHUB

```
var credential = firebase.auth.GoogleAuthProvider.credential(null, token);
firebase.auth().signInWithCredential(credential);
```

- Add the following content security policy to your `manifest.json` to allow importing the Firebase SDK and accessing the Realtime Database as well as Firebase Storage:

```
"content_security_policy":"script-src 'self' https://www.gstatic.com/ https
```

Support

https://firebase.google.com/support/

License