



Cloud Storage for Firebase

Cloud Storage for Firebase is a powerful, simple, and cost-effective object storage service. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality.

Check the [official page](#) for more information.

Setup

Before starting to use any Firebase extensions, you are required to follow some initial configuration steps.

- [Create Project](#)
- [Firebase Console](#)
- [Platform Setup](#)

Functions

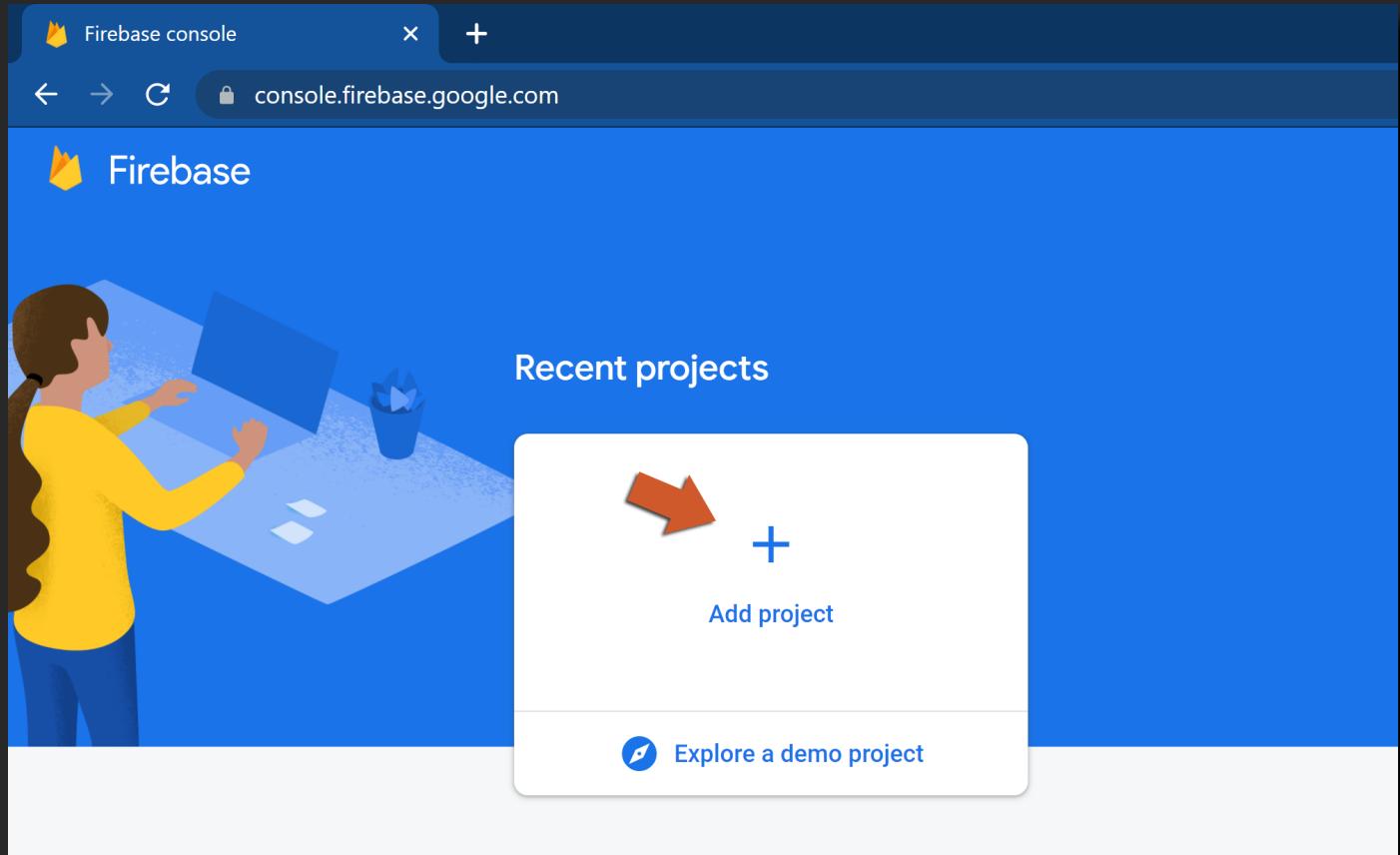
The following functions are provided for working with the Firebase Cloud Storage extension:

- [FirebaseStorage_Cancel](#)
- [FirebaseStorage_Delete](#)
- [FirebaseStorage_Download](#)

Create Project

Before working with any Firebase functions, you must set up your Firebase project:

1. Go to the [Firebase Console](#) web site.
2. Click on **Add Project** to create your new project.



3. Enter a name for your project and click on the **Continue** button.

- ✗ Create a project (Step 1 of 3)

Let's start with a name for
your project?

Enter your project name

my-awesome-project-id

Select parent resource

Continue

4. On the next page, make sure that **Enable Google Analytics for this project** is enabled and then click the **Continue** button:

- ✗ Create a project (Step 2 of 3)

Google Analytics

for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, Predictions, and Cloud Functions.

Google Analytics enables:

💡 A/B testing ?

🌀 Crash-free users ?

🌐 User segmentation & targeting across
Firebase products

👉 Event-based Cloud Functions triggers ?

📈 Predicting user behavior ?

📊 Free unlimited reporting ?



Enable Google Analytics for this project
Recommended

Previous



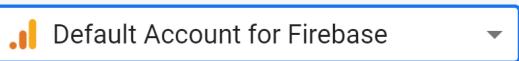
Continue

5. Select your account and click the **Create project** button:

X Create a project (Step 3 of 3)

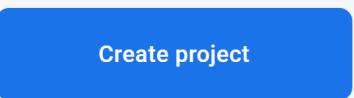
Configure Google Analytics

Choose or create a Google Analytics account [?](#)

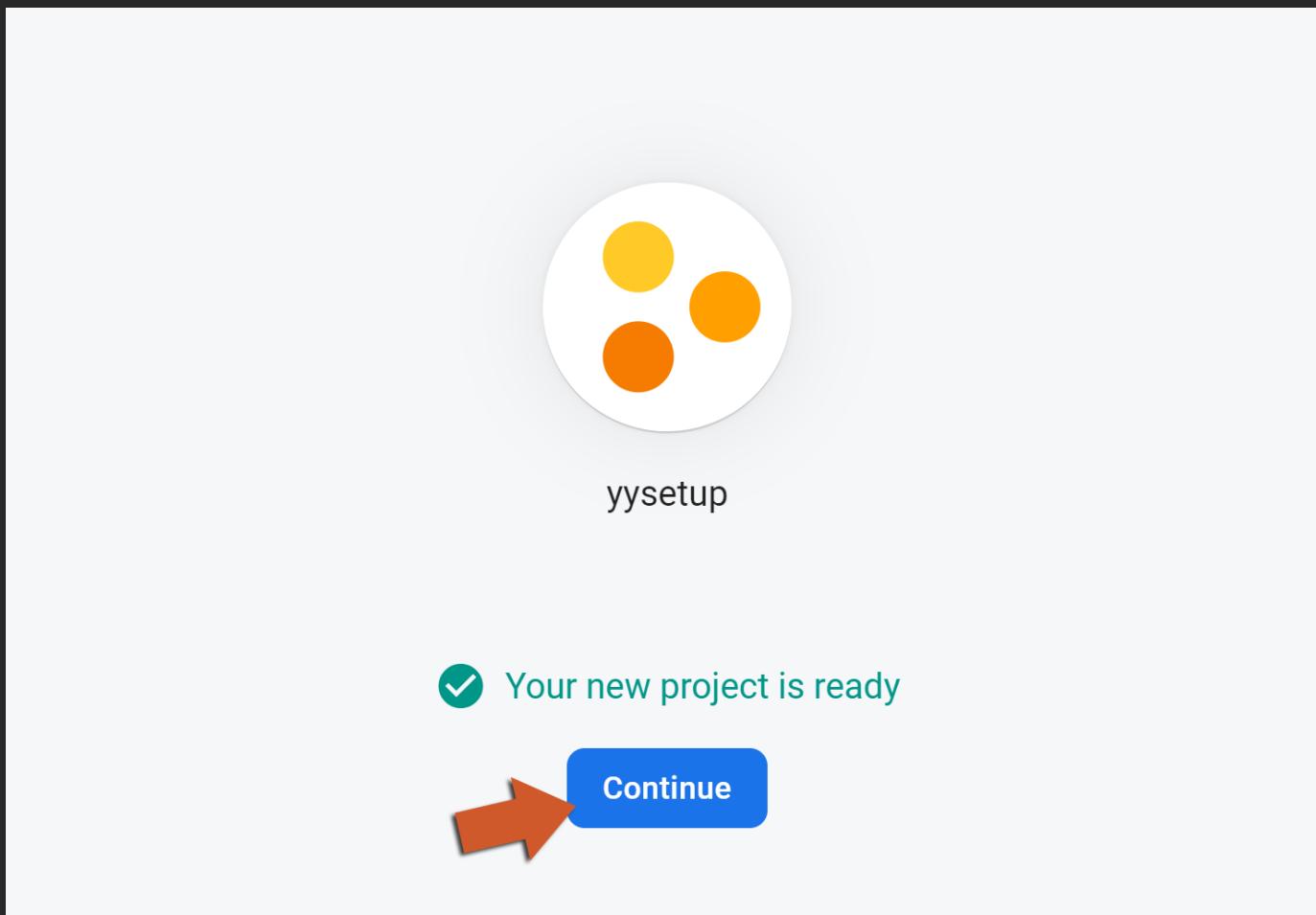
 

Automatically create a new property in this account 

Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. [Learn more.](#)

[Previous](#)  

6. Wait a moment until your project is created; after a few moments you should see the page shown below:



7. You will now be taken to your new project's home page:

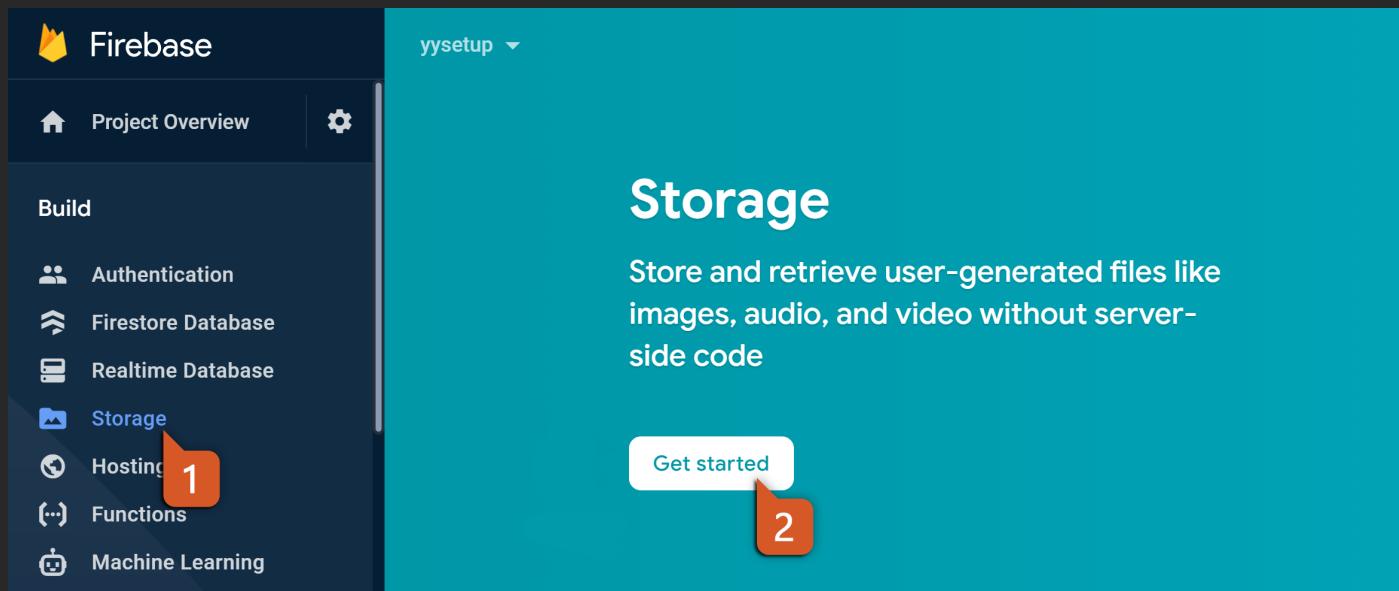
The screenshot shows the Firebase console interface. On the left, a dark sidebar lists various services: Project Overview, Authentication, Firestore Database, Realtime Database, Storage, Hosting, Functions, Machine Learning, Crashlytics, Performance, Test Lab, App Distribution, and Extensions. Below the sidebar, it says 'Spark Free \$0/month' and has an 'Upgrade' button. At the top right are links for 'Go to docs', a bell icon, and a user profile with the letter 'J'. The main content area is titled 'ysetup' with a 'Spark plan' button. It features a blue background with two people: a woman in a yellow shirt and a man in a grey hoodie, both interacting with a large smartphone. A banner below them says 'Get started by adding Firebase to your app' and includes icons for iOS, Android, web development, and a speaker. Below this is another banner with the text 'Store and sync app data in milliseconds' and images of a smartphone and a laptop.

8. Continue your adventure with the Firebase extensions provided for GameMaker!

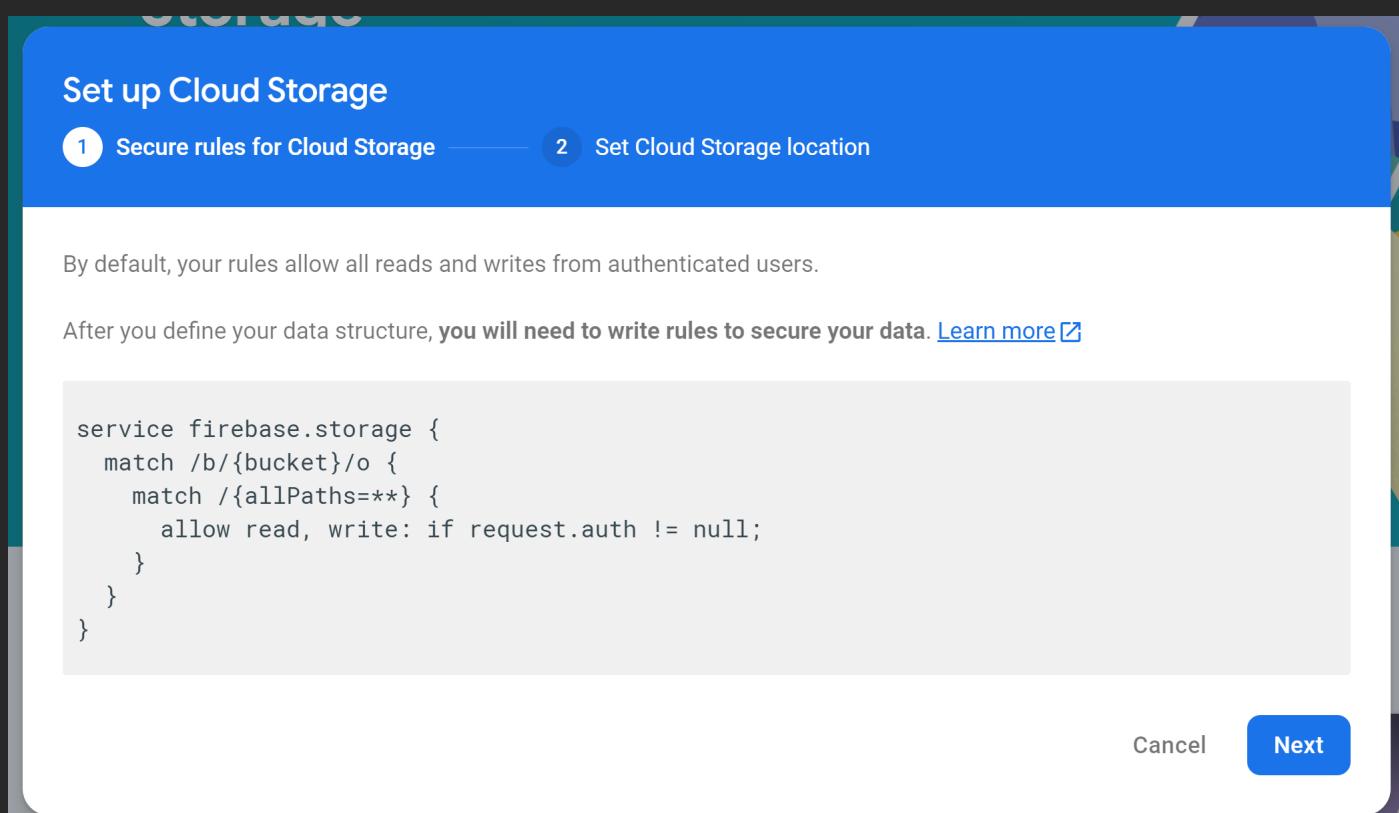
Firebase Console

Before using the Firebase Could Storage extension, you need to configure and enable your storage services in the [Firebase Console](#). Follow the steps below to get started:

1. Go to **Storage** section and click on the **Get Started** button:



2. You will get a pop-up (we will change the rules later), so click on the **Next** button:



3. Select your **Cloud Storage location** and click on **Done**:

Set up Cloud Storage



Secure rules for Cloud Storage

2

Set Cloud Storage location

Your location setting is where your default Cloud Storage bucket and its data will be stored.

A After you set this location, you cannot change it later. This location setting will also be the default location for Cloud Firestore.

[Learn more](#)

Cloud Storage location

nam5 (us-central)



Blaze Plan customers can choose other locations for additional buckets

[Cancel](#)

Done

4. Go to **Rules** section and change the rules condition to `true` (this just for testing purposes, and you will need to create your own rules before sending the app to production). Then click on **Publish**.

Storage

Files

Rules

Usage

[Edit rules](#)

[Monitor rules](#)

1



Guard your data with rules that define who has access to it and how it is structured.

unpublished changes

Publish

Discard

3

```
1  rules_version = '2';
2  service firebase.storage {
3      match /{bucket}/o {
4          match /{allPaths=**} {
5              allow read, write: if true; // request.auth != null;
6          }
7      }
8  }
9 }
```

2

5. You can now use Firebase Cloud Storage on iOS & Android, and some features on Web.

Platform Setup

Firebase Cloud Storage implementation uses SDK dependencies and therefore is only available on **Android**, **iOS** targets and **Web** with limited functionality. In this section we will cover the required setup necessary to start using the Firebase Cloud Storage extension on your game.

Select your target platform below and follow the simple steps to get your project up and running:

IMPORTANT Web target exports will have limited functionality.

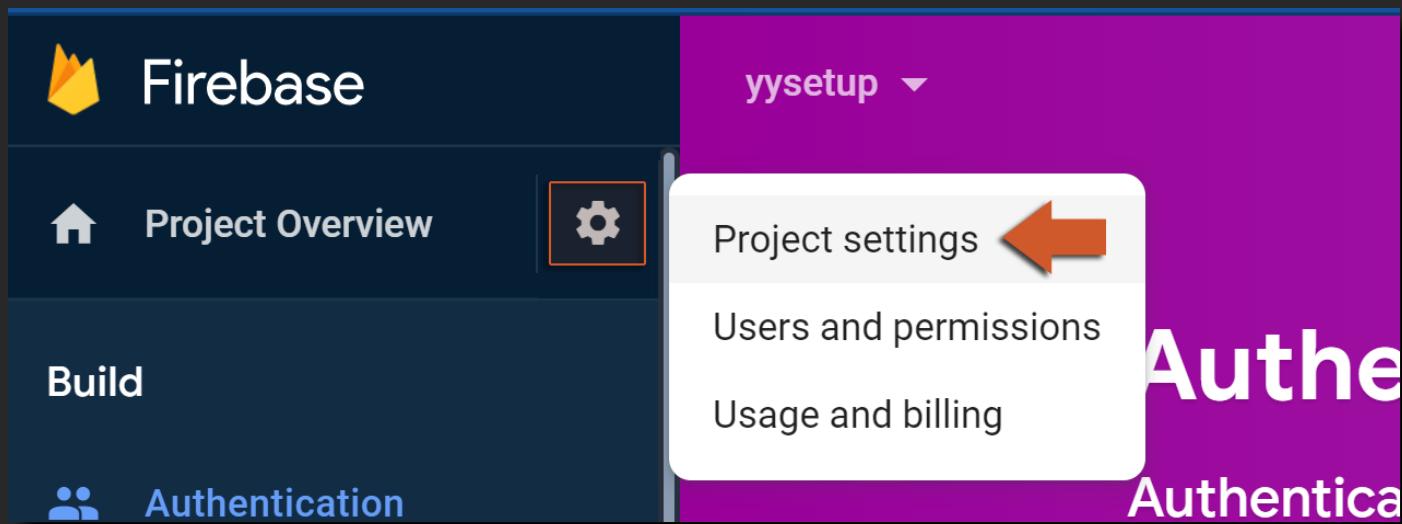
- [Android Setup](#) (once per project)
- [iOS Setup](#) (once per project)
- [Web Setup](#) (once per project)

Android Setup

This setup is necessary for all the Firebase modules using Android and needs to be done once per project, and basically involves importing the `google-services.json` file into your project.

IMPORTANT Please refer to [this Helpdesk article](#) for instructions on setting up an Android project.

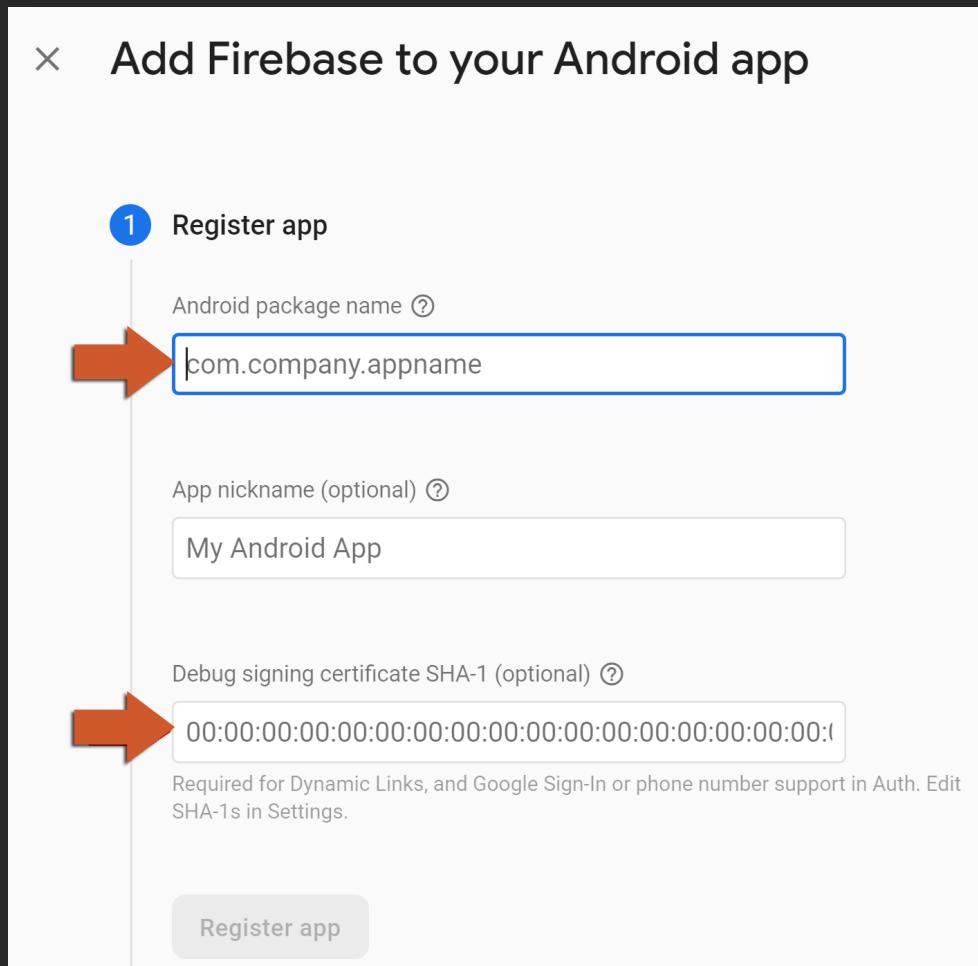
1. Click the **Settings** icon (next to Project Overview) and select **Project settings**:



2. Now go to the **Your apps** section and click on the **Android** button:

The screenshot shows the 'Project settings' page under the 'ysetup' dropdown. The 'Your apps' section is visible, showing a message: 'There are no apps in your project. Select a platform to get started.' Below this message are four circular icons representing different platforms: iOS, TV, Android (highlighted with a red box), and Web. The 'Your apps' button is also highlighted with a red box.

3. On this screen you need enter your **Package name** (required), **App nickname** (optional) and **Debug signing certificate SHA-1** (required if you are using Firebase Authentication).



You can get your package name from the [Android Game Options](#), and your [Debug signing certificate SHA-1](#) from the [Android Preferences](#) (under Keystore):



4. Click on **Download google-services.json** (make sure to save this file as we will need it in subsequent steps).

× Add Firebase to your Android app

✓ Register app

Android package name: com.yoyogames.YoyoPlayServices2, App nickname: yysetup_android

2 Download config file

Instructions for Android Studio below | [Unity](#) [C++](#)

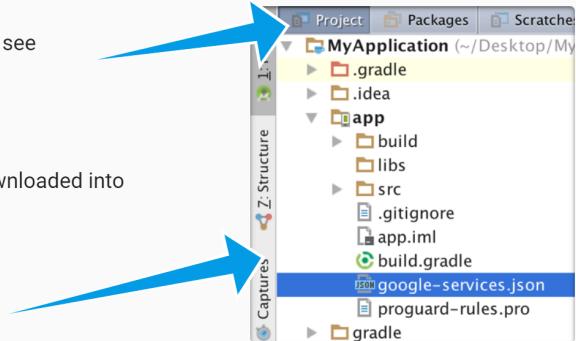
 [Download google-services.json](#)

Switch to the Project view in Android Studio to see your project root directory.

Move the google-services.json file you just downloaded into your Android app module root directory.



google-services.json



[Next](#)

5. Ignore this screen, as this is already done in the extension.

× Add Firebase to your Android app

✓ Register app

Android package name: com.yoyogames.YoyoPlayServices2, App nickname: yysetup_android

Download config file

3 Add Firebase SDK

Instructions for Gradle | [Unity](#) [C++](#)

The Google services plugin for [Gradle](#) loads the google-services.json file you just downloaded. Modify your build.gradle files to use the plugin.

Project-level build.gradle (<project>/build.gradle):

```
buildscript {  
    repositories {  
        // Check that you have the following line (if not, add it):  
        google() // Google's Maven repository  
    }  
    dependencies {  
        ...  
    }  
}
```

6. Click on the Continue to console button.

× Add Firebase to your Android app

Register app

Android package name: com.yoyogames.YoyoPlayServices2, App nickname: yysetup_android

Download config file

Add Firebase SDK

4 Next steps

You're all set!

Make sure to check out the [documentation](#) to learn how to get started with each Firebase product that you want to use in your app.

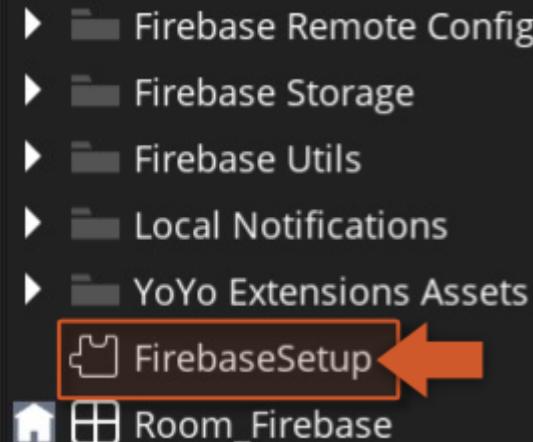
You can also explore [sample Firebase apps](#).

Or, continue to the console to explore Firebase.

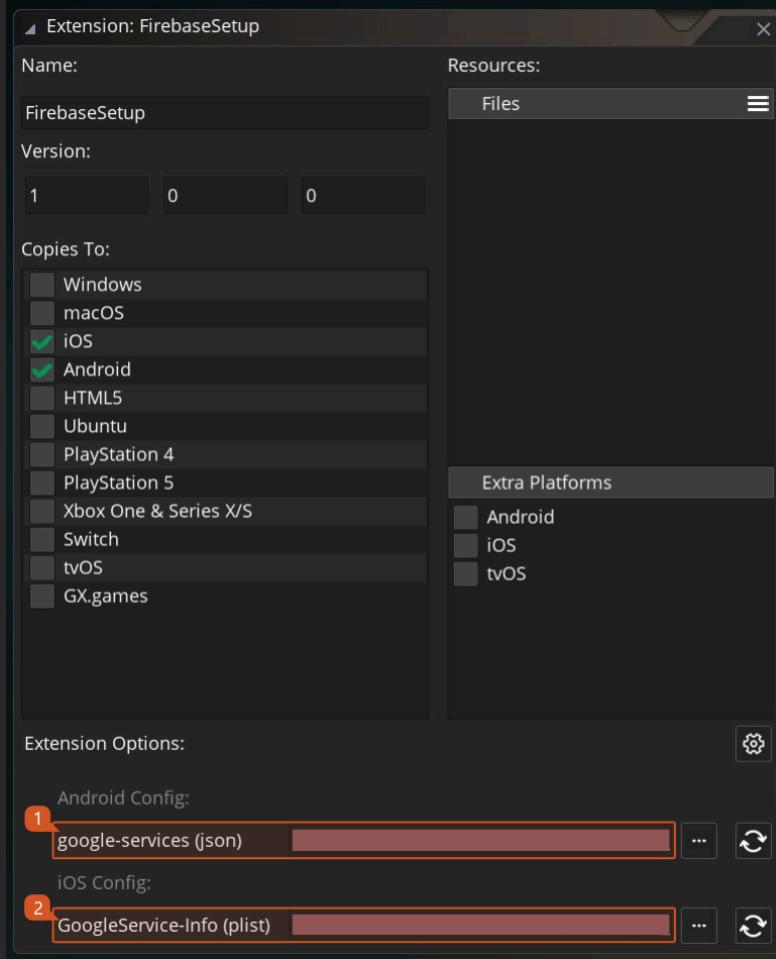
Previous

Continue to console

7. Now go into GameMaker, double click the extension **FirebaseSetup** asset.



8. In the extension panel just fill in the paths for the correct files (Android and/or iOS).



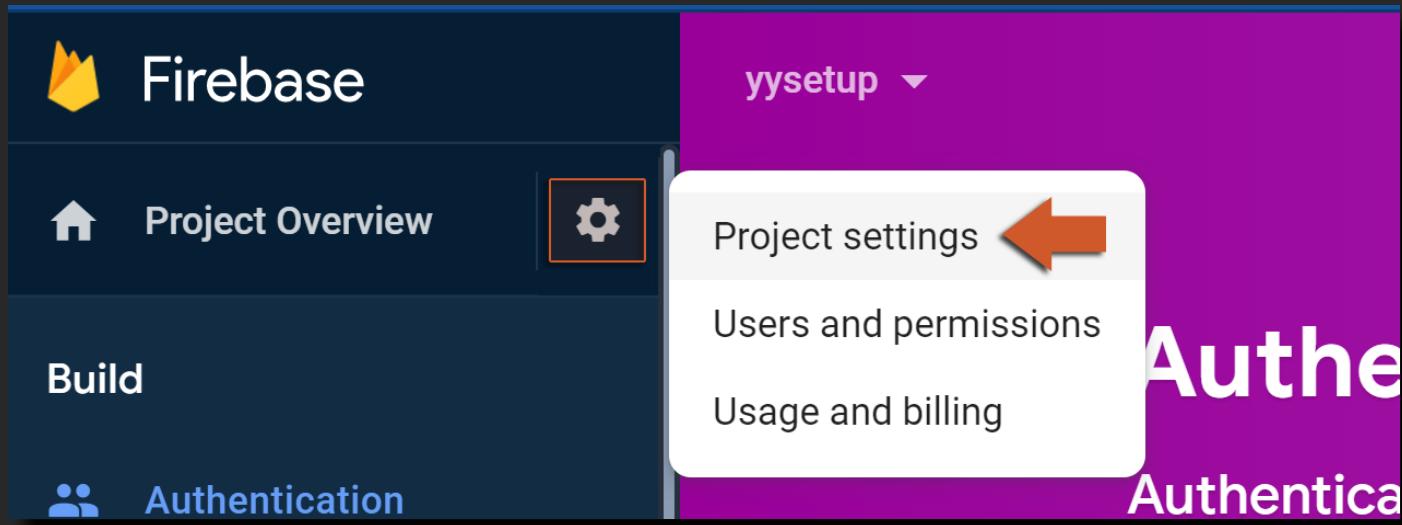
9. You have now finished the main setup for all Firebase Android modules!

iOS Setup

This setup is necessary for all the Firebase modules using iOS and needs to be done once per project, and basically involves importing the `GoogleServices-Info.plist` file into your project.

IMPORTANT Please refer to [this Helpdesk article](#) for instructions on setting up an iOS project.

1. Click the **Settings** icon (next to Project Overview) and select **Project settings**:



2. Now go to the **Your apps** section and click on the **iOS** button:

The screenshot shows the 'Project settings' page under the 'Your apps' section. It displays a message: 'There are no apps in your project. Select a platform to get started.' Below this message are three circular icons representing different platforms: iOS, TV, and Web. The iOS icon is highlighted with a red box. To the right of the icons is a circular icon with a play symbol.

3. Fill the form with your iOS Bundle ID, App nickname and AppStore ID (last two are optional).

X Add Firebase to your iOS app

1 Register app

iOS bundle ID ⓘ

An orange arrow points to this input field.

App nickname (optional) ⓘ

App Store ID (optional) ⓘ

Register app

4. Click on **Download GoogleService-info.plist** (make sure to save this file as we will need it in subsequent steps).

× Add Firebase to your iOS app

✓ Register app

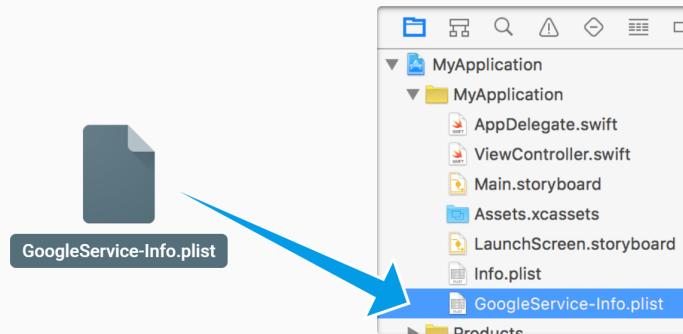
iOS bundle ID: com.yoyogames.yyfirebase

2 Download config file

Instructions for Xcode below | [Unity](#) [C++](#)

→ [Download GoogleService-Info.plist](#)

Move the GoogleService-Info.plist file you just downloaded into the root of your Xcode project and add it to all targets.



[Next](#)

5. Ignore this screen, as this is already done in the extension.

× Add Firebase to your iOS app

✓ Register app

iOS bundle ID: com.yoyogames.yyfirebase

Download config file

3 Add Firebase SDK

Instructions for CocoaPods | [SwiftPM](#) [Download ZIP](#) [Unity](#) [C++](#)

Google services use [CocoaPods](#) to install and manage dependencies. Open a terminal window and navigate to the location of the Xcode project for your app.

Create a Podfile if you don't have one:

\$ pod init



Open your Podfile and add:

```
# add the Firebase pod for Google Analytics
```

6. Ignore this screen as well, as this is also done in the extension.

× Add Firebase to your iOS app

- ✓ Register app
iOS bundle ID: com.yoyogames.yyfirebase

- Download config file

- Add Firebase SDK

4 Add initialization code

To connect Firebase when your app starts up, add the initialization code below to your main `AppDelegate` class.

Swift Objective-C

```
import UIKit
import Firebase

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {
```



7. Click on the Continue to console button:

× Add Firebase to your iOS app

- ✓ Register app
iOS bundle ID: com.yoyogames.yyfirebase

- Download config file

- Add Firebase SDK

- Add initialization code

5 Next steps

You're all set!

Make sure to check out the [documentation](#) to learn how to get started with each Firebase product that you want to use in your app.

You can also explore [sample Firebase apps](#).

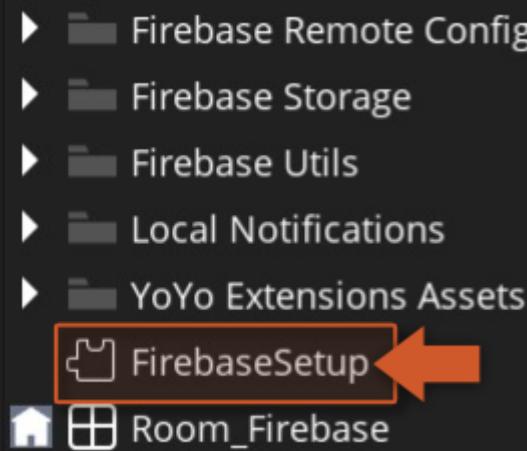
Or, continue to the console to explore Firebase.

Previous

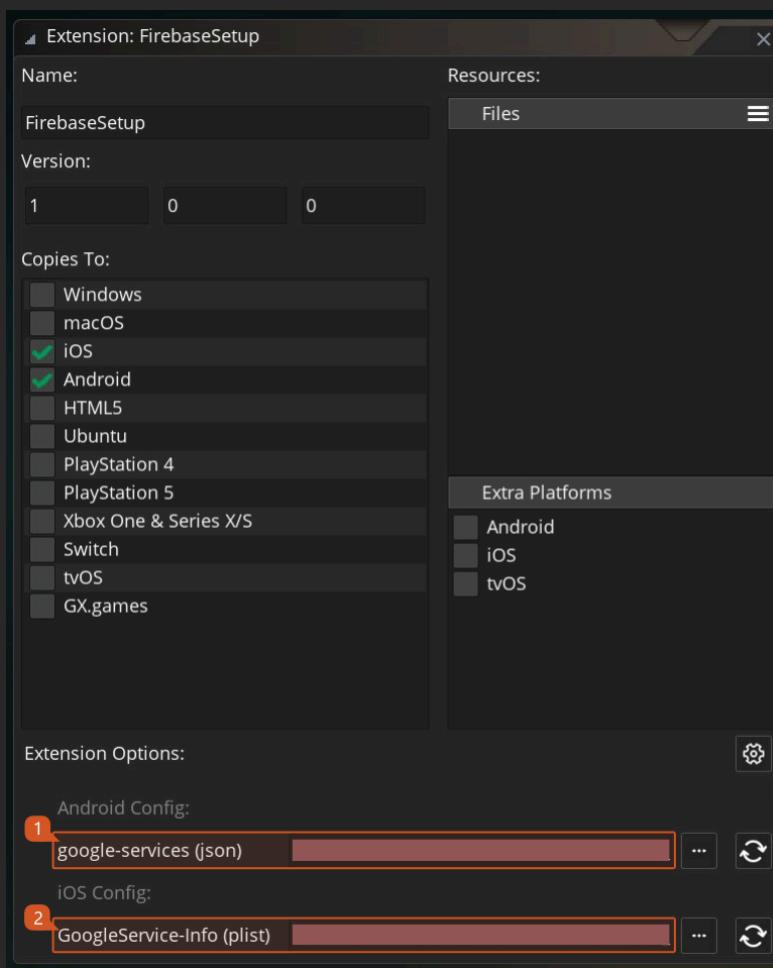
Continue to console



8. Now go into GameMaker, double click the extension **FirebaseSetup** asset.



9. In the extension panel just fill in the paths for the correct files (Android and/or iOS).

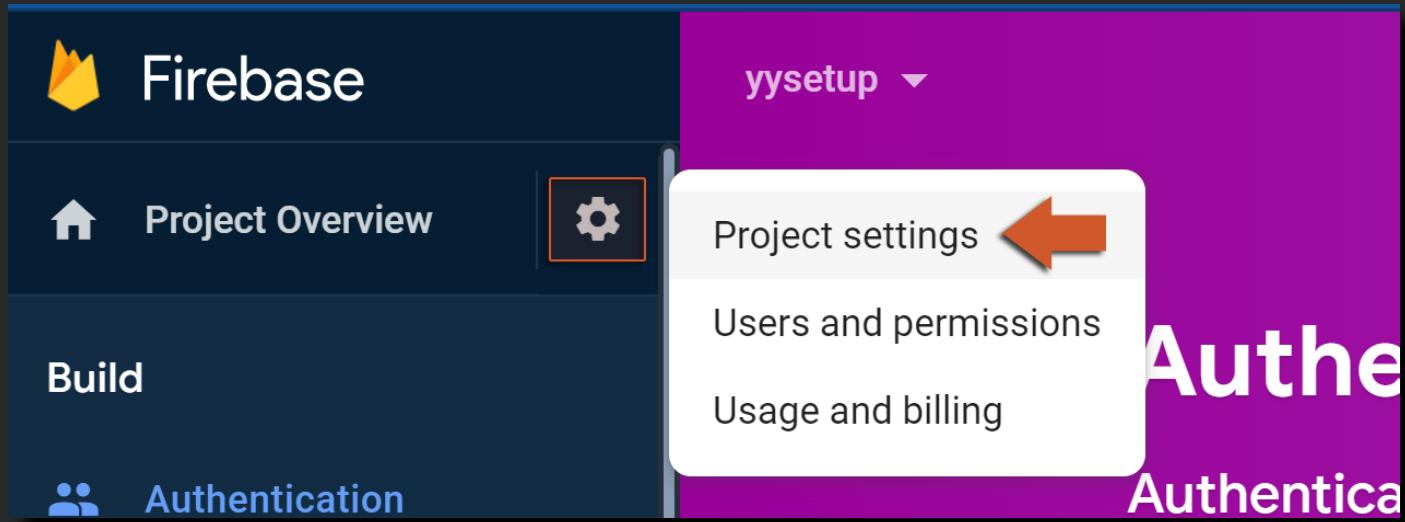


10. Make sure to set up **CocoaPods** for your project *unless* you are only using the REST API in an extension (if one is provided -- not all extensions provide a REST API) or the Firebase Cloud Functions extension (which only uses a REST API).
11. You have now finished the main setup for all Firebase iOS modules!

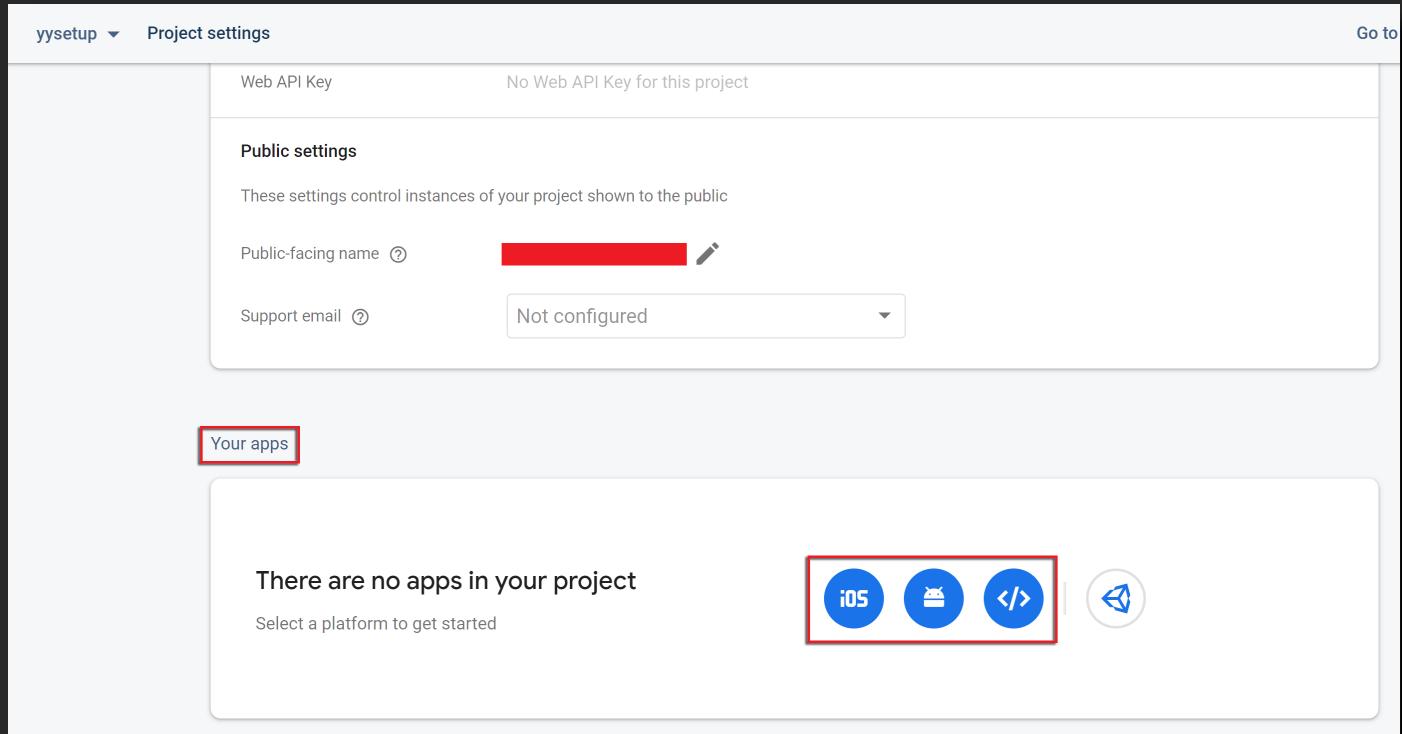
Web Setup

This setup is necessary for all the Firebase modules using Web export and needs to be done once per project, and basically involves adding Firebase libraries and your Firebase values to the `index.html` file in your project.

1. Click the **Settings** icon (next to Project Overview) and then **Project settings**:



2. Now go to the **Your apps** section and click on the **Web (</>)** button:



3. Enter your App nickname (required):

X Add Firebase to your web app

1 Register app

App nickname ⓘ

My web app



! An app nickname is required.

Also set up **Firebase Hosting** for this app. [Learn more ↗](#)

Hosting can also be set up later. It's free to get started anytime.

Register app

4. On this screen, just copy the `firebaseConfig` struct:

2 Add Firebase SDK

Use npm ⓘ Use a <script> tag ⓘ

If you're already using [npm ↗](#) and a module bundler such as [webpack ↗](#) or [Rollup ↗](#), you can run the following command to install the latest SDK:

```
$ npm install firebase
```



Then, initialize Firebase and begin using the SDKs for the products you'd like to use.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries
```

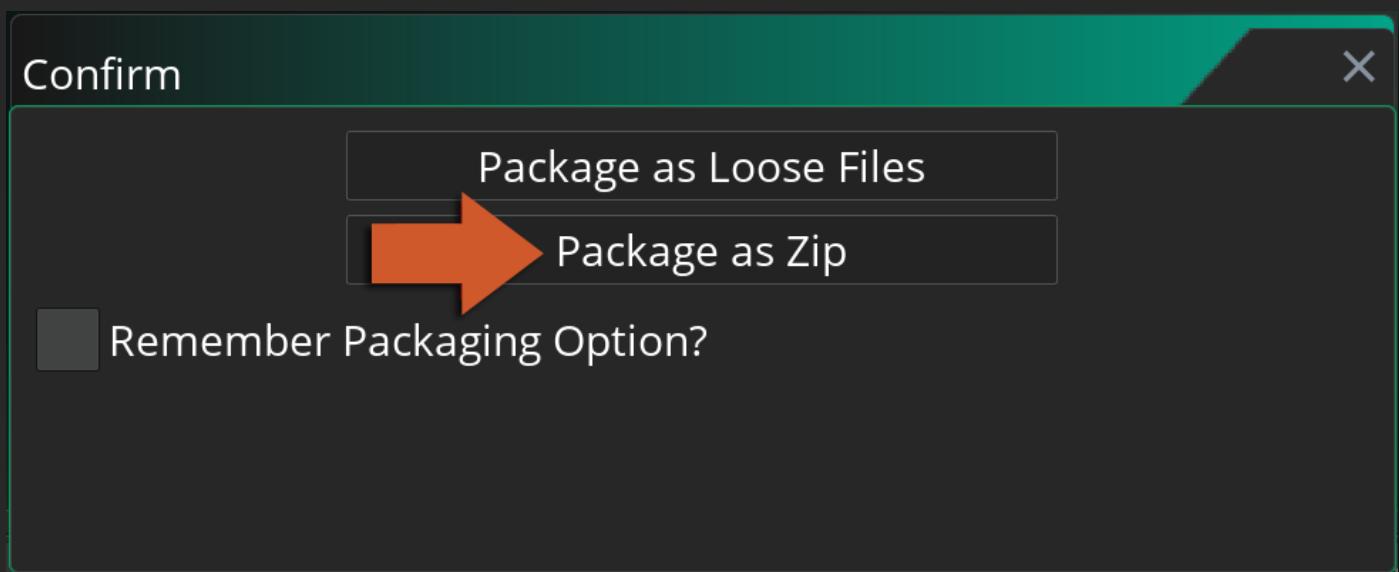
```
// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: [REDACTED],
  authDomain: [REDACTED],
  projectId: [REDACTED],
  storageBucket: [REDACTED],
  messagingSenderId: [REDACTED],
  appId: [REDACTED],
  measurementId: [REDACTED]
};
```

```
// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

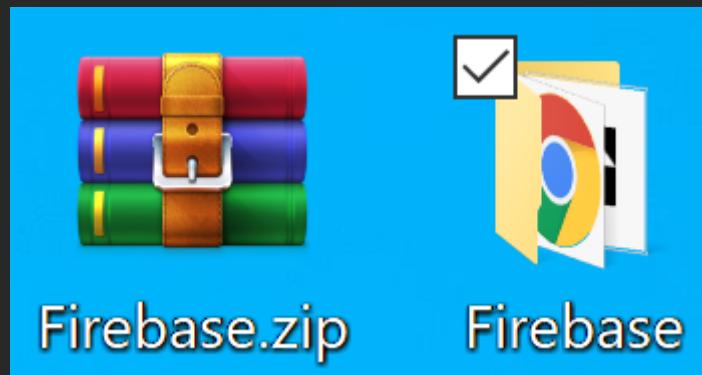


5. Now go back into GameMaker Studio 2 and build your project.

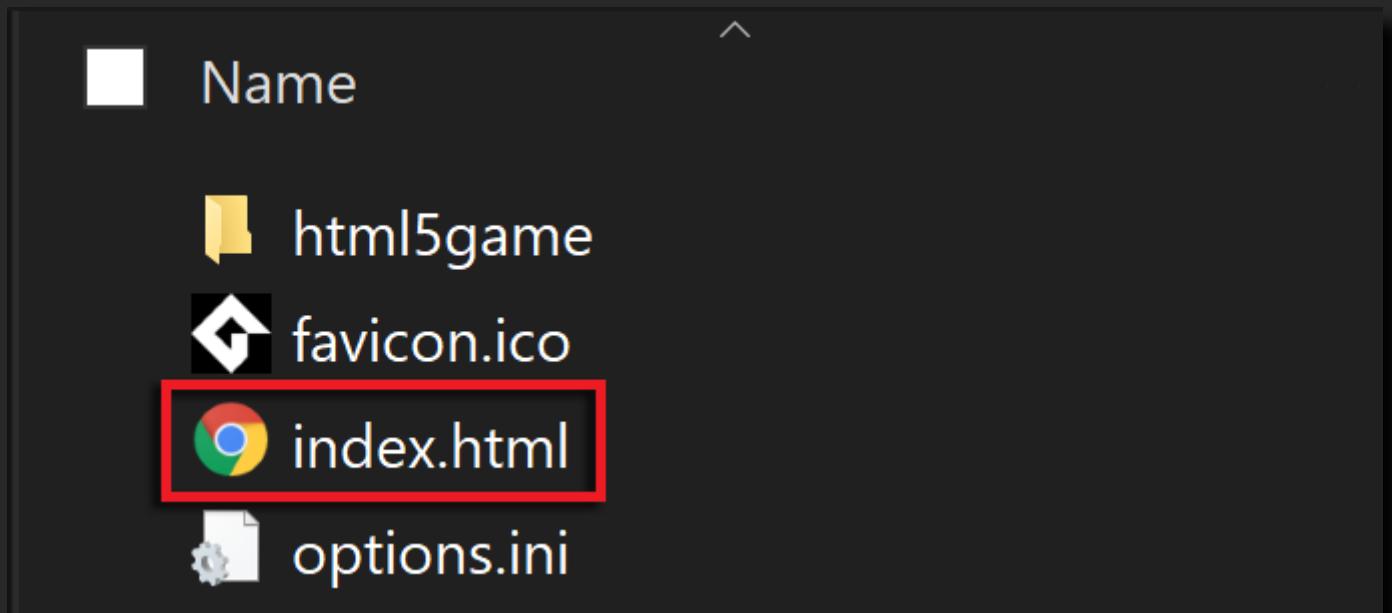
6. Choose the **Package As Zip** option:



7. Locate the created package and **extract it** into a folder.



8. Open the extracted folder and look for an **index.html** file.



9. Open the **index.html** file in Notepad++ or Visual Studio Code (or any other text editor you prefer).

```

65  /* END - Login Dialog Box */
66  :webkit-full-screen {
67      width: 100%;
68      height: 100%;
69  }
70  </style>
71  </head>
72
73 <body>
74     <div class="gm4html5_div_class" id="gm4html5_div_id">
75         <!-- Create the canvas element the game draws to -->
76         <canvas id="canvas" width="1366" height="768" >
77             <p>Your browser doesn't support HTML5 canvas.</p>
78         </canvas>
79     </div>
80
81     <!-- Run the game code -->

```

10. Now we need to add the following code between the `</head>` and `<body>` tags (line 72 in the `html.index` image above):

```

<script src="https://www.gstatic.com/firebasejs/8.9.1.firebaseio.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebaseAnalytics.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebaseAuth.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1.firebaseio.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebaseRestore.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebaseRemoteConfig.js"></script>

<script>
  const firebaseConfig = {
    apiKey: "",
    authDomain: "",
    databaseURL: "",
    projectId: "",
    storageBucket: "",
    messagingSenderId: "",
    appId: "",
    measurementId: ""
  };
  firebase.initializeApp(firebaseConfig);

</script>

```

11. Replace the `const firebaseConfig` part with the one copied in step 4:

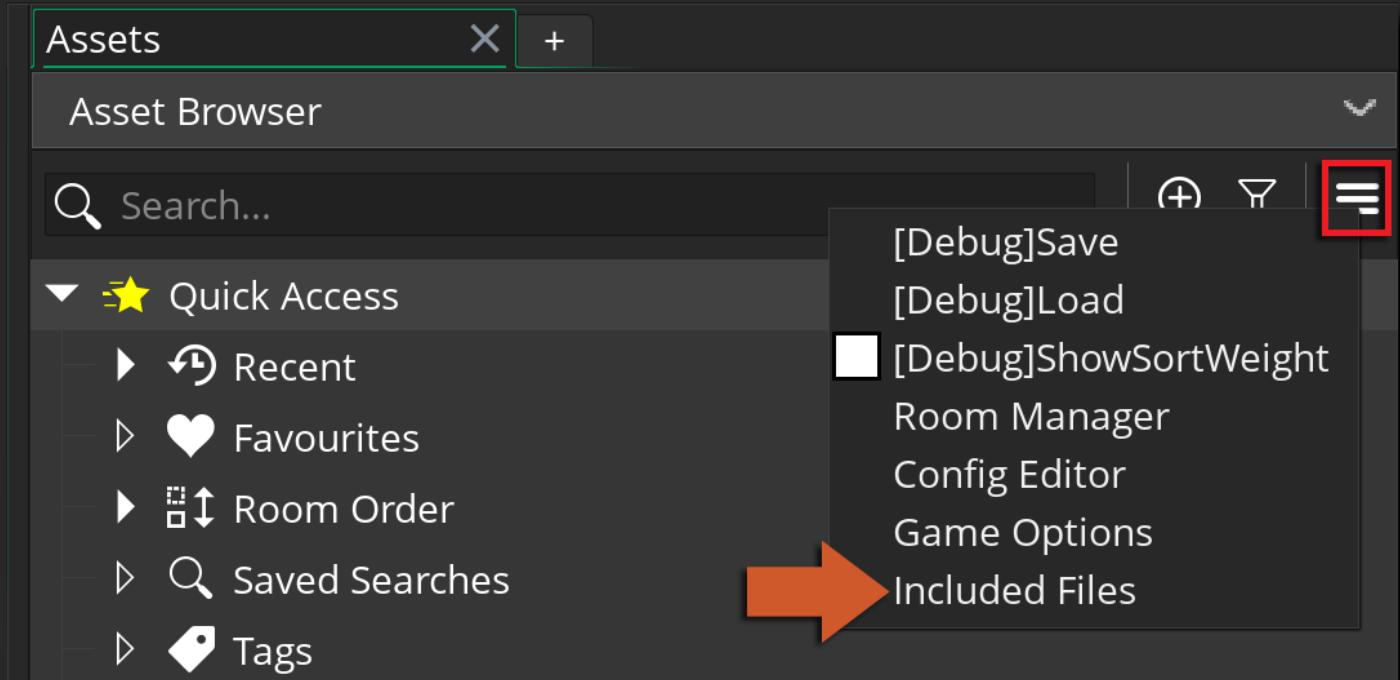
```

68         height: 100%;
69     }
70 
```

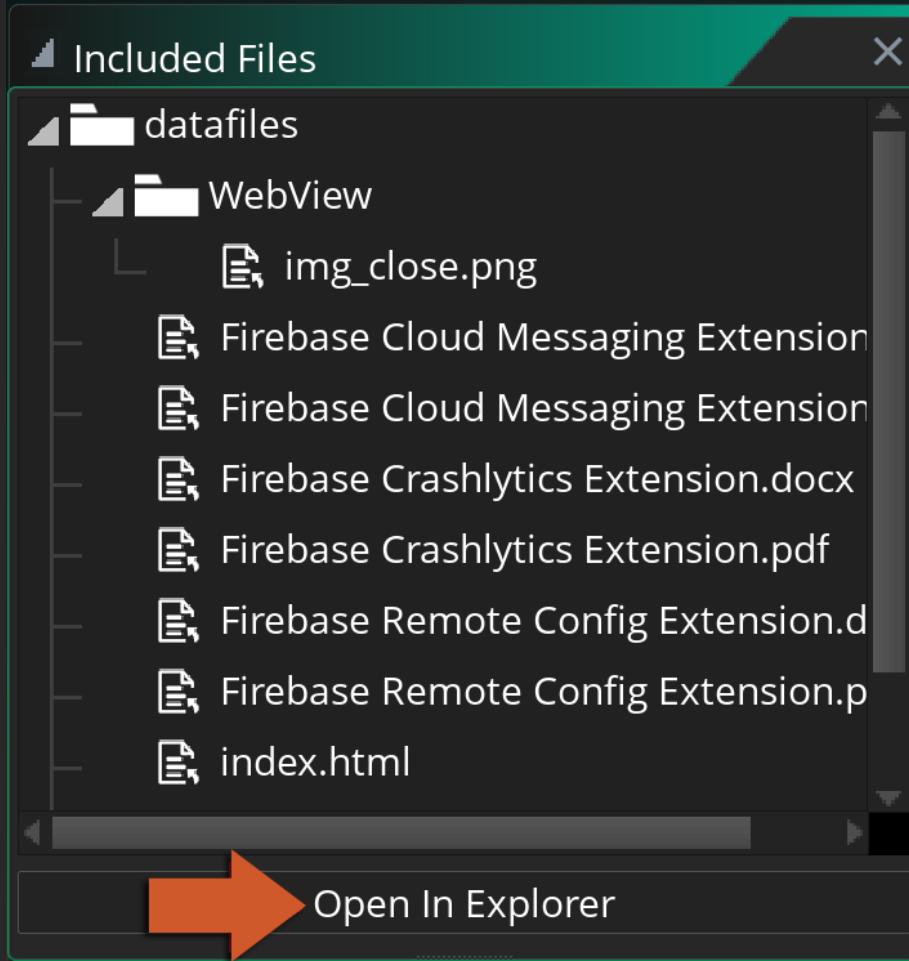
```

71 </style>
72 </head>
73
74 <!-- Firebase -->
75 <script src="https://www.gstatic.com/firebasejs/8.9.1.firebaseio.js"></script>
76 <script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-analytics.js"></script>
77 <script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-auth.js"></script>
78 <script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-database.js"></script>
79 <script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-firebase.js"></script>
80
81
82 <script>
83
84 // Your web app's Firebase configuration
85 // For Firebase JS SDK v7.20.0 and later, measurementId is optional
86 const firebaseConfig = {
87   apiKey: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
88   authDomain: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
89   databaseURL: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
90   projectId: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
91   storageBucket: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
92   messagingSenderId: "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",
93   appId: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
94   measurementId: "XXXXXXXXXXXXXXXXXXXXXXXXXXXX"
95 };
96
97 firebase.initializeApp(firebaseConfig);
98
99 </script>
100
101
102
103 <body>
104     <div class="gm4html5_div_class" id="gm4html5_div_id">
```

12. Go back into GameMaker and open your **Included Files** folder.



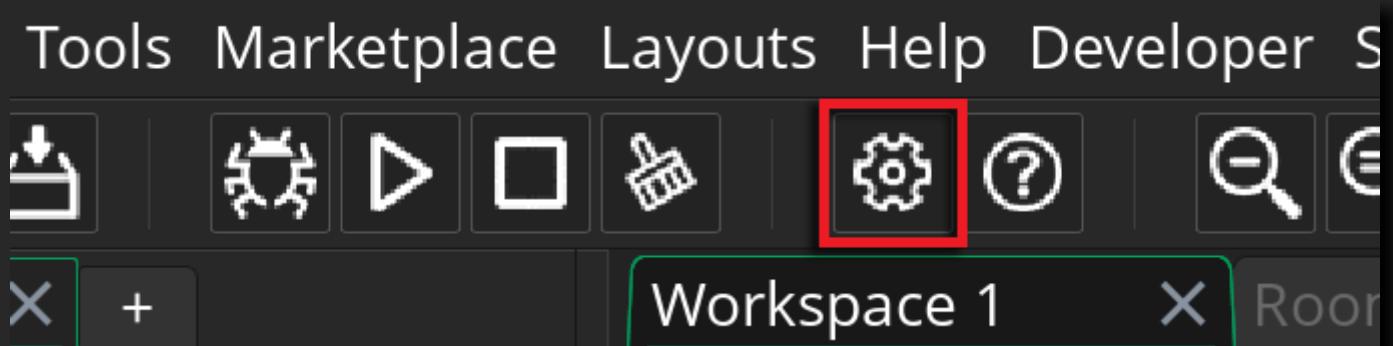
13. Press the Open in Explorer button:



14. Place your **index.html** file inside the folder that opens (/datafiles).

Name	Date modified	Type
WebView	9/14/2021 7:44 AM	File folder
index.html	9/13/2021 9:12 AM	Chrome H

15. Back in GameMaker, click on the **Game Options** button.



16. Go into the **HTML5** platform settings

Game Options - Main

- ▲ Main Options
 - General
- ▲ Platform Settings
 - Opera GX
 - Windows
 - macOS
 - Ubuntu
 - HTML5** ←
 - Android
 - Amazon Fire
 - iOS
 - tvOS

17. In the Advanced section go to the "Include file as index.html" dropdown and select the **index.html** option (this is the file we have just added to the included files).

HTML5 - General

Created with GameMaker Studio 2

Browser Title

1 0 0 0 Version

html5game Folder Name

index.html Output Name

▲ Options

- Output debug to console
- Display cursor
- Display "Running outside server" alert

« ▲ Advanced

Use Default

Included file as index.html

Use Default

index.html

Loading bar extension

18. Press **Apply** and the main setup for all Firebase Web modules is finished!

FirebaseStorage_Cancel

This function is used to cancel an upload (see [FirebaseStorage_Upload](#)) or download (see [FirebaseStorage_Download](#)) process.

Syntax:

```
Fi rebaseStorage_Cancel (I stener)
```

Argument	Description
listener	The upload/download asynchronous listener

Returns:

N/A

Example:

```
I stener = Fi rebaseStorage_Upl oad(I ocal fi le, path);  
  
// After some time  
Fi rebaseStorage_Cancel (I stener);
```

The code above will start a file upload to the given `path` (using the [FirebaseStorage_Upload](#) function) and after a while cancel it.

FirebaseStorage_Delete

This function deletes a path in the Firebase Cloud Storage and returns a listener identifier that can be used for tracking progress on the request.

This is an asynchronous function that will trigger the **Async Social** event during progress and when finished.

Syntax:

```
Fi rebaseStorage_Delete(fi rebasePath, [bucket])
```

Argument	Type	Description
firebasePath	string	The remote path on the Firebase Cloud Storage server
bucket	string	Other Firebase Storage bucket OPTIONAL

Returns:

Real (Asynchronous Listener ID)

Triggers:

Asynchronous Social Event

Key	Type	Description
type	string	The string "Fi rebaseStorage_Delete"
listener	real	The asynchronous listener ID.
path	string	The remote path.
success	bool	Whether or not the download succeeded (if finished) OPTIONAL

Example:

```
var uid = FirebaseAuthAuthentication_GetUID();
listenerId = FirebaseStorage_Delete("UserProfiles/" + uid + "/img.png");
```

The code above uses the user identifier (provided by the function `FirebaseAuthenticati on_GetUID()` from the Firebase Authentication extension) to create a remote path and then deletes a file in that path. The function call will then return a listener ID (`listenerId`) that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "FirebaseStorage_Delete")
{
    if (async_load[? "success"])
    {
        show_debug_message("File was deleted");
    }
    else
    {
        show_debug_message("File could not be deleted");
    }
}
```

The code above matches the response against the correct event **type** and logs whether the task was completed successfully or not.

FirebaseStorage_Download

This function is used to download files from a remote path and returns a listener identifier that can be used for tracking progress on the request.

This is an asynchronous function that will trigger the **Async Social** event during progress and when finished.

IMPORTANT This function is not available on the Web target.

Syntax:

```
FirebaseStorage_Download(localPath, firebasePath, [bucket])
```

Argument	Type	Description
localPath	string	The local path where the files will be downloaded
firebasePath	string	The remote path on the Firebase Cloud Storage server
bucket	string	Other Firebase Storage bucket OPTIONAL

Returns:

Real (Asynchronous Listener ID)

Triggers:

Asynchronous Social Event

Key	Type	Description
type	string	The string "FirebaseStorage_Download"
listener	real	The asynchronous listener ID.
path	string	The remote file path.
localPath	string	The local file path.

success	bool	Whether or not the download succeeded (if finished) OPTIONAL
transferred	real	Number of transferred bytes (if NOT finished) OPTIONAL
total	real	Total number of bytes (if NOT finished) OPTIONAL

Example:

```
var uid = FirebaseAuthentication_GetUID()
listenerId = FirebaseStorage_Download("profilePic.png", "UserProfiles/" + uid + "/img.png")
```

The code above uses the user identifier (provided by the function

`FirebaseAuthenticati on_GetUID()` from the Firebase Authentication extension) to create a remote path string, then downloads a file from the server to a local path. The function call will then return a listener ID (`listenerId`) that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "FirebaseStorage_Download")
{
    if (ds_map_exists(async_load[? "success"]))
    {
        if (async_load[? "success"])
        {
            global.percent = 100;
            img = sprite_add(async_load[? "localPath"], 0, 0, 0, 0, 0);
            show_message_async("Download SUCCESS")
        }
        else
        {
            global.percent = 0;
            show_message_async("Download FAILED")
        }
    }
    else
    {
        global.percent = 100 * async_load[? "transferred"] / async_load[? "total"];
    }
}
```

The code above matches the response against the correct event **type** and if the download has not finished yet (`"success"` key is not present) then calculates the download percentage and stores it inside a global variable (`global.percent`). When the download finishes it checks if it was a successful download and if so it adds the downloaded file as a sprite (using the **sprite_add** function).

NOTE Always remember to [free sprites](#) added this way when you no longer need them, otherwise it can lead to memory leaks.

FirebaseStorage_GetURL

This function asynchronously retrieves a long lived download URL with a revocable token. This can be used to share the file with others, but can be revoked by the developer in the Firebase Console if desired.

This is an asynchronous function that will trigger the **Async Social** event when the task is finished.

Syntax:

```
Fi rebaseStorage_GetURL(fi rebasePath, [bucket])
```

Argument	Type	Description
firebasePath	string	The remote path on Firebase Cloud Storage server.
bucket	string	Other Firebase Storage bucket OPTIONAL

Returns:

Real (Asynchronous Listener ID)

Triggers:

Asynchronous Social Event

Key	Type	Description
type	string	The constant "Fi rebaseStorage_GetURL"
listener	real	The asynchronous listener ID.
path	string	The remote file path.
success	bool	Whether or not the task succeeded
value	string	The URL of the file (if task succeeded) OPTIONAL

Example:

```
listenerId = FirebaseStorage.GetURL(path);
```

The code above tries to retrieve an URL from a remote path on the Firebase Cloud Storage server. The function call will then return a listener ID (`listenerId`) that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "FirebaseStorage.GetURL")
{
    if (async_load[? "success"])
    {
        show_debug_message(async_load[? "path"] + " URL: " + async_load[? "value"]);
    }
    else
    {
        show_debug_message("There was an error retrieving the URL");
    }
}
```

The code above matches the response against the correct event **type** and if the task was successful it logs the retrieved URL for the specific path.

FirebaseStorage_List

This function queries the contents of a Firebase Cloud Storage path, if `pageToken` is provided the query will start after the page number provided. It also returns a listener identifier that can be used for tracking progress on the request.

This is an asynchronous function that will trigger the **Async Social** event when the task is finished.

Syntax:

```
Fi rebaseStorage_List(path, maxResults, [pageToken, bucket])
```

Argument	Description
path	The remote Firebase path to query
maxResults	The max number of results
pageToken	Token of the last request OPTIONAL
bucket	Other Firebase Storage bucket OPTIONAL

Returns:

Real (Asynchronous Listener ID)

Triggers:

Asynchronous Social Event

Key	Type	Description
type	string	The string "Fi rebaseStorage_List"
listener	real	The asynchronous listener ID.
path	string	The remote path.
success	bool	Whether or not the task succeeded
files	string	A JSON formatted string of an array of file paths OPTIONAL

folders	string	A JSON formatted string of an array of folder paths OPTIONAL
pageToken	real	The current query page.

Example:

```
listenerId = FirebaseStorageList("/testing", 5);
```

The code above queries for files and folders inside the path `"/testing"`, returning a maximum of `5` results. The function call will then return a listener ID (`listenerId`) that can be used inside an [Async Social](#) event.

```
if (async_load[? "type"] == "FirebaseStorageList")
{
    if (async_load[? "success"])
    {
        global.arrayOfFiles = json_parse(async_load[? "files"]);
        global.arrayOfFolders = json_parse(async_load[? "folders"]);
    }
}
```

The code above matches the response against the **correct event type** and if the task was successful it stores the files and folders paths parsed as arrays (using the **json_parse** function) inside global variables (`global.arrayOfFiles` and `global.arrayOfFolders`).

FirebaseStorage_ListAll

This function queries all the contents of a Firebase Cloud Storage path and returns a listener identifier that can be used for tracking progress on the request.

This is an asynchronous function that will trigger the **Async Social** event when the task is finished.

Syntax:

```
Fi rebaseStorage_ListAll (path, [bucket])
```

Argument	Description
path	The remote Firebase path to query
bucket	Other Firebase Storage bucket OPTIONAL

Returns:

Real (Asynchronous Listener ID)

Triggers:

Asynchronous Social Event

Key	Type	Description
type	string	The string "Fi rebaseStorage_ListAll"
listener	real	The asynchronous listener ID.
path	string	The remote path.
success	bool	Whether or not the task succeeded
files	string	A JSON formatted string of an array of file paths OPTIONAL
folders	string	A JSON formatted string of an array of folder paths OPTIONAL

Example:

```
listenerId = FirebaseStorageListener("/testing");
```

The code above queries for all files and folders inside the path `"/testing"`. The function call will then return a listener ID (`listenerId`) that can be used inside an [Async Social](#) event.

```
if (async_load[? "type"] == "FirebaseStorageListener")
{
    if (async_load[? "success"])
    {
        global.arrayOfFiles = json_parse(async_load[? "files"]);
        global.arrayOfFolders = json_parse(async_load[? "folders"]);
    }
}
```

The code above matches the response against the **correct event type** and if the task was successful it stores the files and folders paths parsed as arrays (using the [json_parse](#) function) inside global variables (`global.arrayOfFiles` and `global.arrayOfFolders`).

FirebaseStorage_Download

This function uploads files to a given path and returns a listener identifier that can be used for tracking progress on the request.

This is an asynchronous function that will trigger the **Async Social** event during progress and when finished.

IMPORTANT This function is not available on the Web target.

Syntax:

```
FirebaseStorage_Upload(localPath, firebasePath, [bucket])
```

Argument	Type	Description
localPath	string	The local path to a file
firebasePath	string	The remote path on the Firebase Cloud Storage server
bucket	string	Other Firebase Storage bucket OPTIONAL

Returns:

Real (Asynchronous Listener ID)

Triggers:

Asynchronous Social Event

Key	Type	Description
type	string	The string "FirebaseStorage_Upload"
listener	real	The asynchronous listener ID.
path	string	The remote file path.
localPath	string	The local file path.

success	bool	Whether or not the upload succeeded (if finished) OPTIONAL
transferred	real	Number of transferred bytes (if NOT finished) OPTIONAL
total	real	Total number of bytes (if NOT finished) OPTIONAL

Example:

```
var uid = FirebaseAuthentication_GetUID()
listenerId = FirebaseStorage_Upload("profilePic.png", "UserProfiles/" + uid + "/img.png")
```

The code above uses the user identifier (provided by the function

`FirebaseAuthenticati on_GetUID()` from the Firebase Authentication extension) to create a remote path string, and then uploads a file to the remote server. The function call will return a listener ID (`listenerId`) that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "FirebaseStorage_Upload")
{
    if (!ds_map_exists(async_load[? "success"]))
    {
        global.percent = 100 * async_load[? "transferred"] / async_load[? "total"];
    }
    else
    {
        show_debug_message("The upload terminated");
    }
}
```

The code above matches the response against the correct event **type** and if the upload has not finished yet (`"success"` key is not present) then calculates the upload percentage and stores it inside a global variable (`global.percent`).