Google Apps Script: Access Google Sheets, Maps & Gmail in 4 Lines of Code

GSP235

Google Cloud Self-Paced Labs

Overview

In this lab, you are introduced to one of the easiest ways to write code that accesses Google developer technologies, all by leveraging one of the mainstream web development languages, JavaScript. Using Google Apps Script, you write code to extract an address sitting in a cell in a Google Sheet, generate a Google Map based on that address, and send a link to the map to yourself or a friend using Gmail. The best part? It really takes only 4 lines of code!

Objectives

- Learn a bit about Apps Script... enough to get you going
- Create a new Google Sheets spreadsheet
- Learn how to enter the script editor for any document
- Edit Apps Script code, save, and run it
- Use Gmail to see the fruits of your labor!

Suggested experience

The following experience would enhance your learning experience:

- Basic JavaScript skills (helpful but not required)
- Basic spreadsheet skills

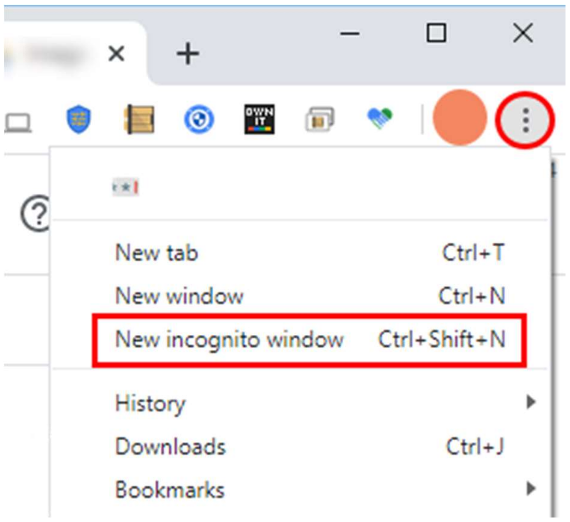
Setup and requirements

What you need

- An internet browser. Chrome browser is recommended.
- Time. 10 minutes to read through steps in the lab. 30 minutes to step through the hands-on practice lab. Once you start this lab, you cannot pause and return later.
  All work for this lab is done in your Qwiklab G Suite account. If you have a different G Suite or Google account open, **sign out** before you start this lab. If you are signed into more than one account, you get the message "Server error is not authorized to access Admin Console of .... Note: Multi-Login is not supported for resellers in Admin Console`".

Note: If you are using a Pixelbook, run this lab in an Incognito window. To do this:

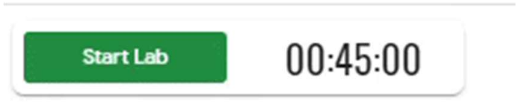
In the Chrome browser, click on the three dots next to your user picture, then select **New incognito window**.



Start your lab

When you are ready, click **Start Lab**. If you need to pay for the lab, a pop-up opens for you to select your payment method.

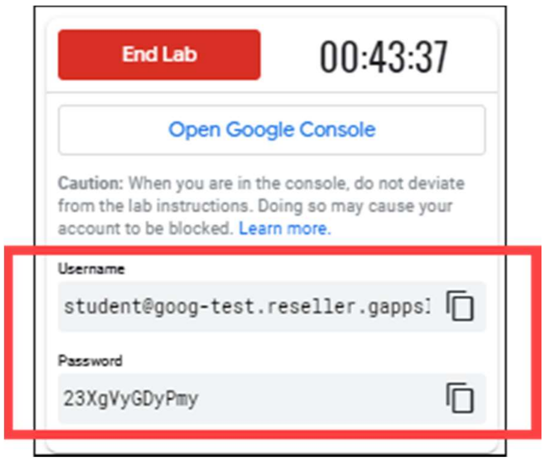
You can track your lab’s setup progress with the status message in the blue box.



When the lab is set up, you will see the button to **Open Google Console** and the temporary credentials you must use for this lab.

Username and Password

Find the Username and Password you need to access your Qwiklab's Google account under the button to **Open Google Console**.



Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

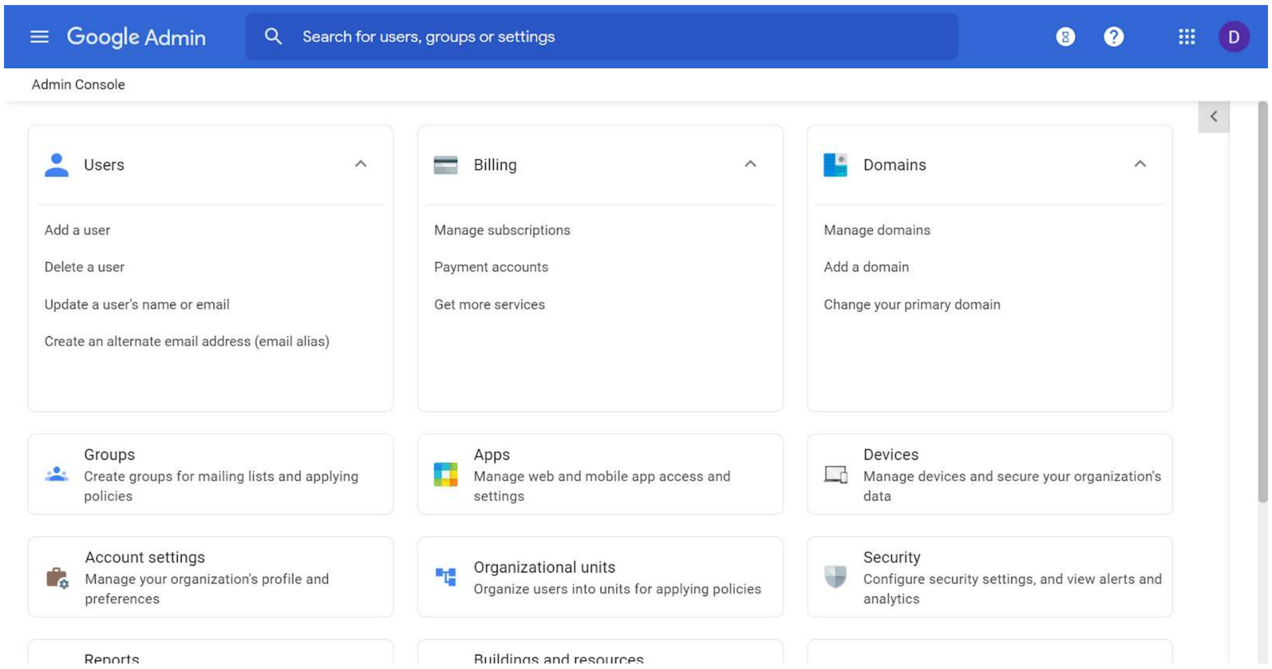
Username
student@goog-test.reseller.gapps

Password
23XgVyGDyPmy

Sign in to Google Workspace Admin Console

Visit admin.google.com or click **Open Google Console** in the upper-left panel and enter the username and password located in the connection details panel
Accept the Google Workspace via Reseller Agreement and **Accept Terms of Service** to accept the Supplemental Terms and Conditions For the Google Workspace Enterprise Plus free trial.

The **Admin console** opens.


Google Admin

Search for users, groups or settings

Admin Console

Users
- Add a user
- Delete a user
- Update a user's name or email
- Create an alternate email address (email alias)

Billing
- Manage subscriptions
- Payment accounts
- Get more services

Domains
- Manage domains
- Add a domain
- Change your primary domain

Groups
Create groups for mailing lists and applying policies

Apps
Manage web and mobile app access and settings

Devices
Manage devices and secure your organization's data

Account settings
Manage your organization's profile and preferences

Organizational units
Organize users into units for applying policies

Security
Configure security settings, and view alerts and analytics

Reports

Buildings and resources

If you get the server error "We are unable to process your request at this time, please try again later", the Admin console is still spinning up. Close the browser window and click **Open Google Console** again.

What is Google Apps Script?

Google Apps Script has a development environment that may be different from what you're used to. With Apps Script, you:

- Develop in a browser-based code editor but can choose to develop locally if using clasp, the command-line deployment tool for Apps Script
- Code in a specialized version of JavaScript customized to access G Suite, and other Google or external services (URLfetch, JDBC, etc.)
- Safely ignore writing authorization code because Apps Script handles it for you
- Do not host your app—it lives and runs on Google servers in the cloud

NOTE: Teaching you Apps Script is outside of the scope of this lab. There are plenty of online resources. The official documentation features an overview with quickstarts, tutorials, as well as videos. This lab introduces you to the Apps Script development environment so you're comfortable creating code and get you thinking about the types of applications you can build with it.

Apps Script applications come in one of two forms:

1. Bound—meaning it's forever, and only tied to one Google document (Doc, Sheet, Slide, Site, or Form)
2. Standalone—an independent script not tied to any G Suite documents

Bound and Standalone apps can also be published to expose more broadly:

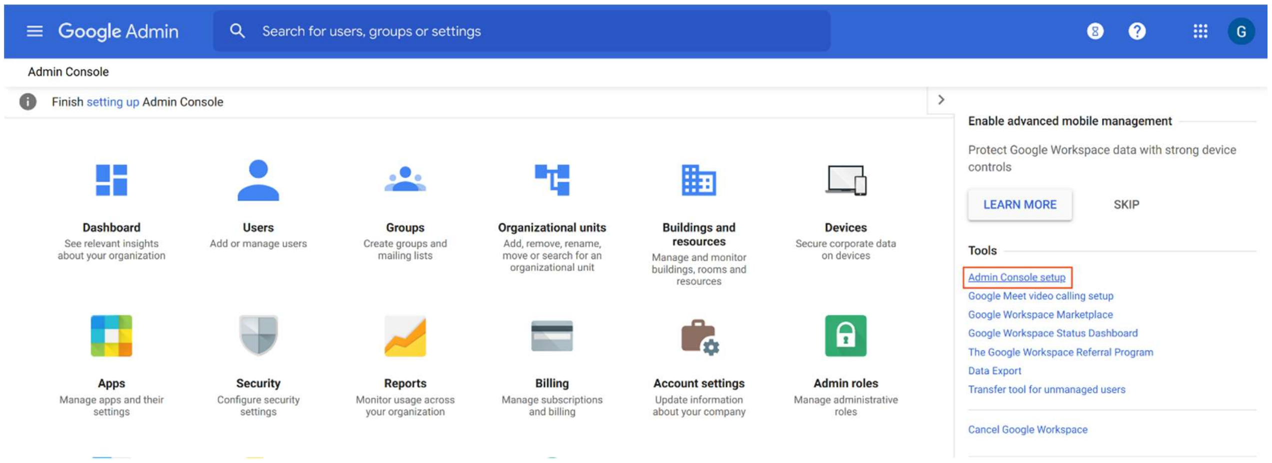
- Not published—remains private, accessible only to project owners
- Published as an add-on—your app can be installed from the add-on store
- Published as web app—your app handles HTTP requests and has web UI components
- Embedded in Google Sites—published web apps can be embedded in either the new Sites or classic Sites pages
- Published as an API executable—your app can be accessed through the Execution API
- Some valid combination of the above

  Your first Apps Script app will be **bound** to a Google Sheet. Time to create a new spreadsheet!

Setup G Suite Admin Console

If you get the server error "We are unable to process your request at this time, please try again later" during these setup step, the Admin console is still spinning up. Close the browser window and click Open Google Console again.

1. From the Admin Console, click **Admin Console setup** from the right under the **Tools** section:

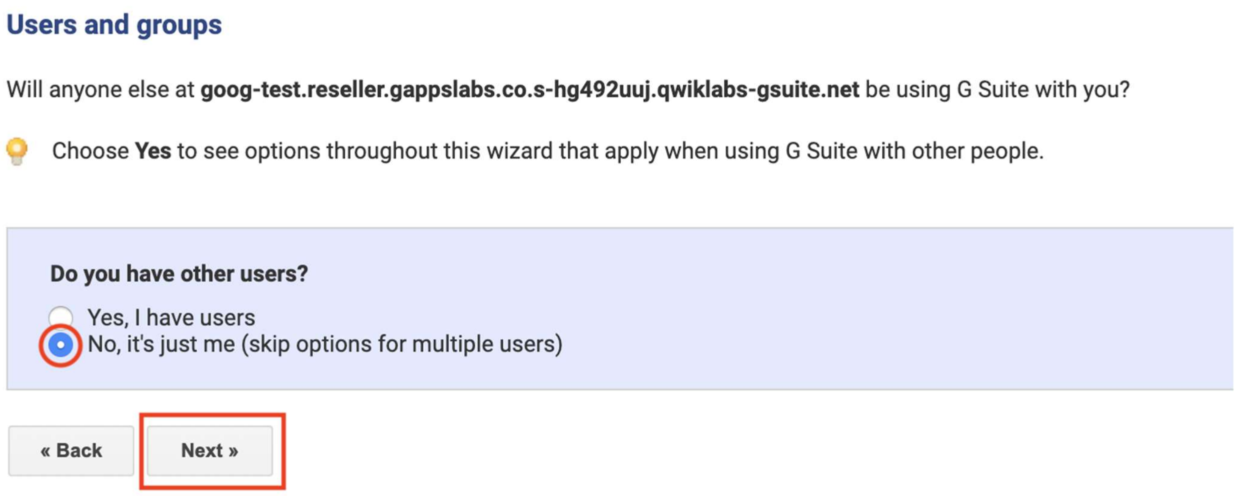

Google Admin
Search for users, groups or settings
Admin Console
Dashboard
See relevant insights about your organization
Users
Add or manage users
Groups
Create groups and mailing lists
Organizational units
Add, remove, rename, move or search for an organizational unit
Buildings and resources
Manage and monitor buildings, rooms and resources
Devices
Secure corporate data on devices
Apps
Manage apps and their settings
Security
Configure security settings
Reports
Monitor usage across your organization
Billing
Manage subscriptions and billing
Account settings
Update information about your company
Admin roles
Manage administrative roles
LEARN MORE
SKIP
Tools
Admin Console setup
Google Meet video calling setup
Google Workspace Marketplace
Google Workspace Status Dashboard
The Google Workspace Referral Program
Data Export
Transfer tool for unmanaged users
Cancel Google Workspace

Finish setting up Admin Console

Enable advanced mobile management

Protect Google Workspace data with strong device controls

2. Click **Next**. When you get to this prompt, select **No, it's just me** and click **Next**:



Users and groups

Will anyone else at **goog-test.reseller.gappslabs.co.s-hg492uuj.qwiklabs-gsuite.net** be using G Suite with you?

Choose **Yes** to see options throughout this wizard that apply when using G Suite with other people.

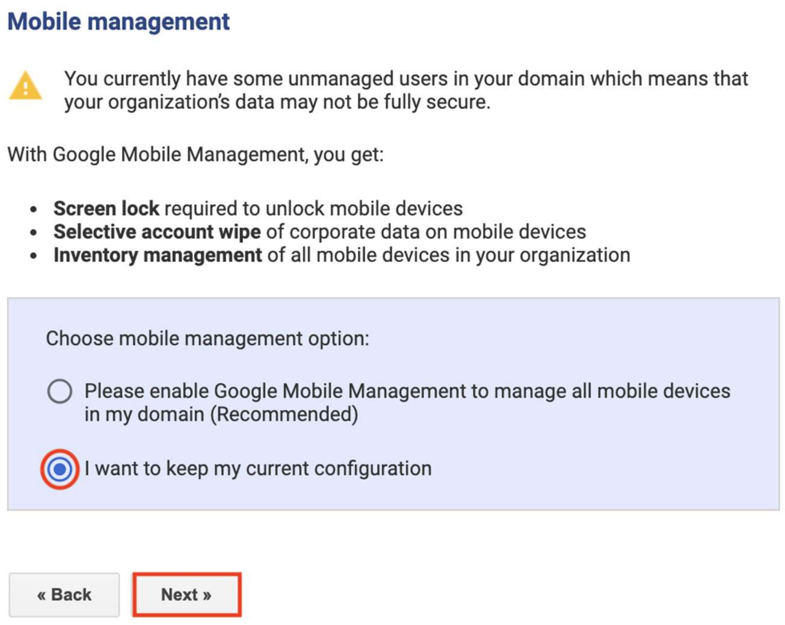
Do you have other users?

Yes, I have users [✓ No, it's just me (skip options for multiple users)]

3. When you land on the "Set up your apps" page, click **Next**. then click **Next** > **Next** > **Next** > **Do this later** > **Next**.

4. When you land on the "Google Calendar" page, click **Do this later**.

5. When you get to the "Mobile management" step, select **I want to keep my current configuration** and click **Next**:



Mobile management

You currently have some unmanaged users in your domain which means that your organization's data may not be fully secure.

With Google Mobile Management, you get:

- **Screen lock** required to unlock mobile devices
- **Selective account wipe** of corporate data on mobile devices
- **Inventory management** of all mobile devices in your organization

Choose mobile management option:

Please enable Google Mobile Management to manage all mobile devices in my domain (Recommended) [✓ I want to keep my current configuration]

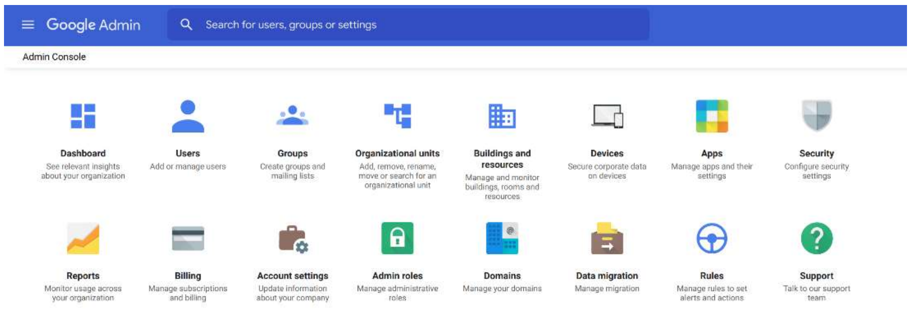
6. Click **Next**. Then click **Next** if you land on the "Extend and customize your Apps" page (this may not appear.)

7. Click **Next** when you land on the "Recommended for G Suite" page.

8. When you land on the "Use the Chrome browser" page, click **Next**.

9. When you land on the "Training and support" page, click **Do this later**.

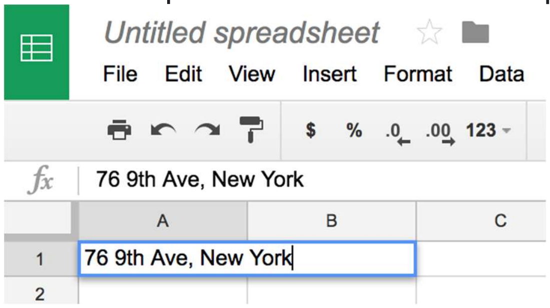
10. You should now be brought back to the G Suite Admin page:


Google Admin
Search for users, groups or settings
Admin Console
Dashboard
See relevant insights about your organization
Users
Add or manage users
Groups
Create groups and mailing lists
Organizational units
Add, remove, rename, move or search for an organizational unit
Buildings and resources
Manage and monitor buildings, rooms and resources
Devices
Secure corporate data on devices
Apps
Manage apps and their settings
Security
Configure security settings
Reports
Monitor usage across your organization
Billing
Manage subscriptions and billing
Account settings
Update information about your company
Admin roles
Manage administrative roles
Domains
Manage your domains
Data migration
Manage migration
Rules
Manage rules to set alerts and actions
Support
Talk to our support team

Create a new Google Sheet and enter a street address

Enter a street address in a new Google Sheet by following these instructions:

1. Click this convenience link to create a new Google Sheet.
2. On the blank spreadsheet, click into the first cell in the upper left-hand corner (A1). It will be in column A and row 1. Now enter an address in that cell—pick any worldwide valid street address with a targeted location such as postal code or city and state/province. Here is an example of entering an address in New York City:



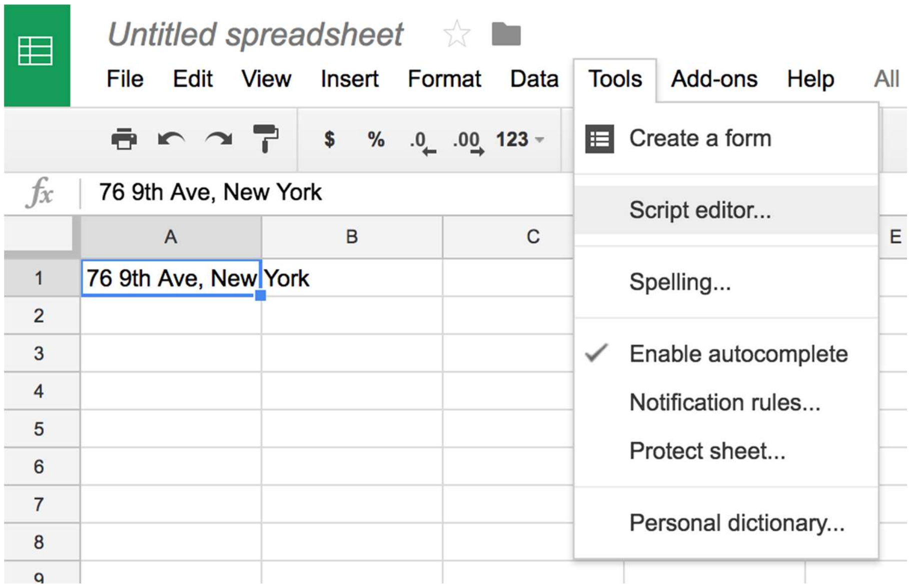
76 9th Ave, New York

That's all you have to do in the Sheet. Now you're ready to enter the editor and write some code!

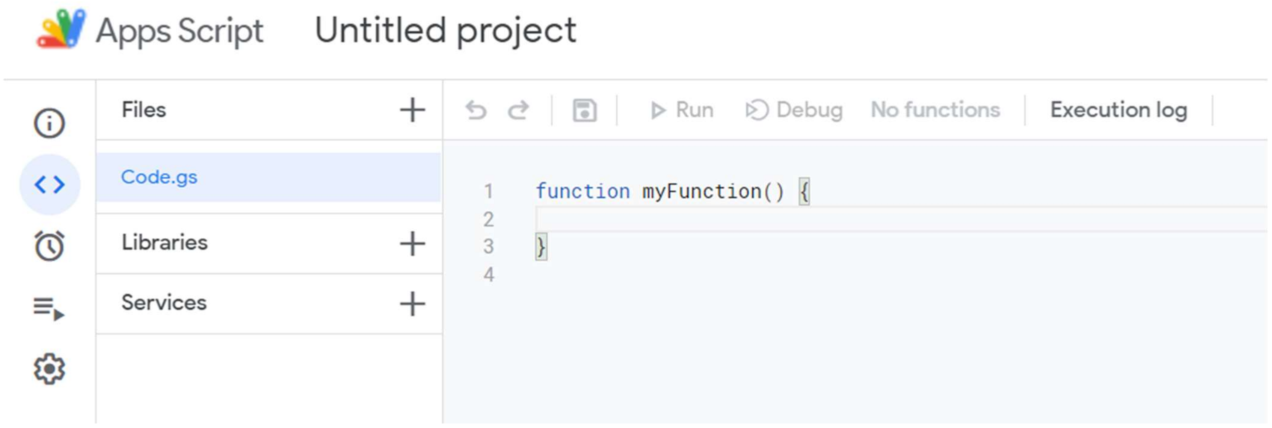
Edit Apps Script code

Now that you have a Google Sheet, it's time to edit its bound script.

1. To open the script editor, select **Tools** from the top menu bar, then click on **Script editor**.



2. What you see now in your browser is the code editor for the bound script:



```
function myFunction() {

}
```

A default function named `myFunction()` is automatically created for you, and in the editor. That's it... you're now ready to write your application.

Edit the (template) code

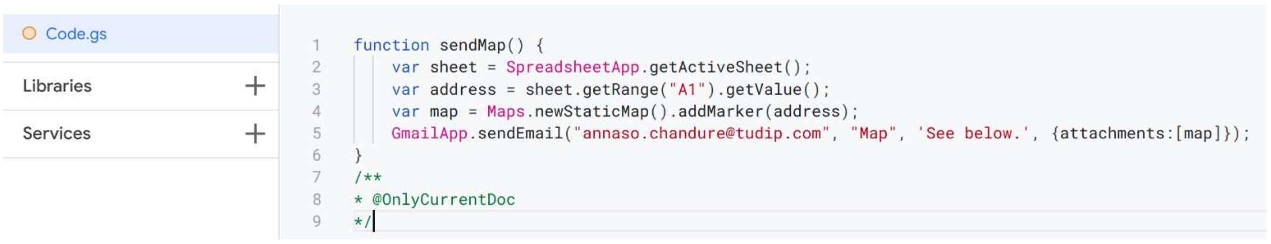
1. The "template" code you're given is empty and doesn't do much. Copy the code below to replace the template code in the editor window. Then update `<YOUR_EMAIL>` with an email address you can access:

```
function sendMap() {
    var sheet = SpreadsheetApp.getActiveSheet();
    var address = sheet.getRange("A1").getValue();
    var map = Maps.newStaticMap().addMarker(address);
    GmailApp.sendEmail("<YOUR_EMAIL>", "Map", 'See below.', {attachments:[map]});
}
```
content_copy

2. To restrict this app to access only the Sheet you're working with (as opposed to all of a user's Sheets), add this annotation as a file-level comment for the peace of mind of your users:

```
/**
* @OnlyCurrentDoc
*/
```
content_copy

Other than this optional annotation, the best part is that the 4 lines of `sendMap()` make up the entire app.
Of course, you need to replace the fake email address `(*friend@example.com*)` with one of yours that you can access during this lab. Did you notice when you replaced the code in the editor, a red circle showed up to the left of the file name?

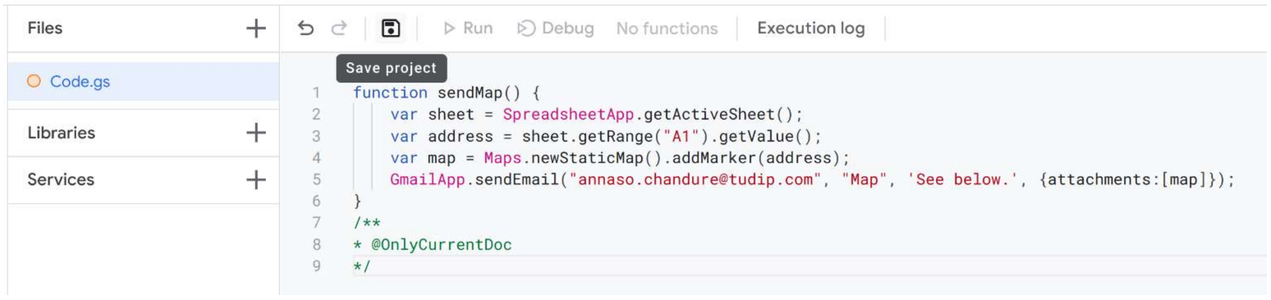


```
function sendMap() {
    var sheet = SpreadsheetApp.getActiveSheet();
    var address = sheet.getRange("A1").getValue();
    var map = Maps.newStaticMap().addMarker(address);
    GmailApp.sendEmail("annaso.chandure@tudip.com", "Map", 'See below.', {attachments:[map]});
}
/**
* @OnlyCurrentDoc
*/
```

That just means you've edited the file which now needs to be saved. You'll see it every time you have an unsaved edit.

3. Save and name your project (call it anything you like—for example, "Hello Maps!").

   Save the file by clicking the small disk icon .

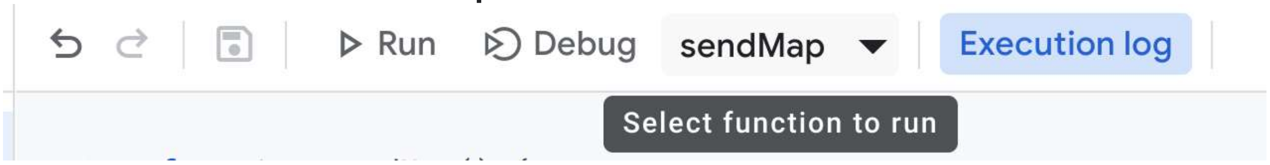


```
function sendMap() {
    var sheet = SpreadsheetApp.getActiveSheet();
    var address = sheet.getRange("A1").getValue();
    var map = Maps.newStaticMap().addMarker(address);
    GmailApp.sendEmail("annaso.chandure@tudip.com", "Map", 'See below.', {attachments:[map]});
}
/**
* @OnlyCurrentDoc
*/
```

Alternatively, you can CTRL+S (PCs, Linux) or Command+S (Mac). If you haven't named your project yet, you must do so before you can proceed.

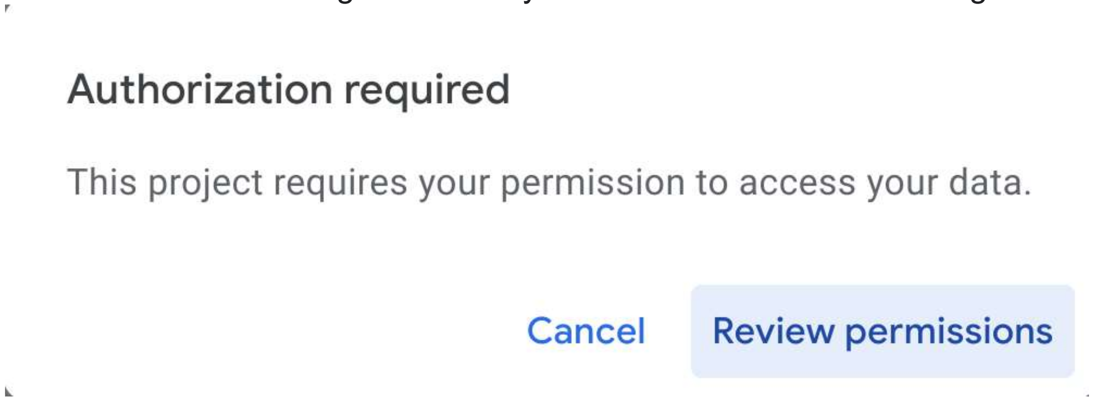
Run the Google Sheets, Maps, and Gmail app

1. Time to run the app. Since the function was renamed to `sendMap()`, Select the function to run as **sendMap**:



Click on the "run" triangle icon and ensure the function to run is `sendMap()`.

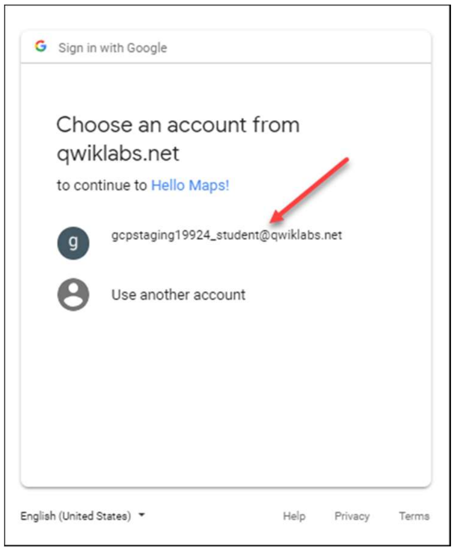
2. One of the Apps Script features that developers appreciate is that you don't have to write the authorization code. Although Apps Script manages this, users (of your app) still need to grant permission (for this script) to access your Sheet and be able to send email through Gmail on your behalf. The first auth dialog looks like this:



Authorization required

This project requires your permission to access your data.

3. Click **Review Permissions**.
4. If prompted, choose your account (your **Username** found in the **Connection Details** section of the lab).



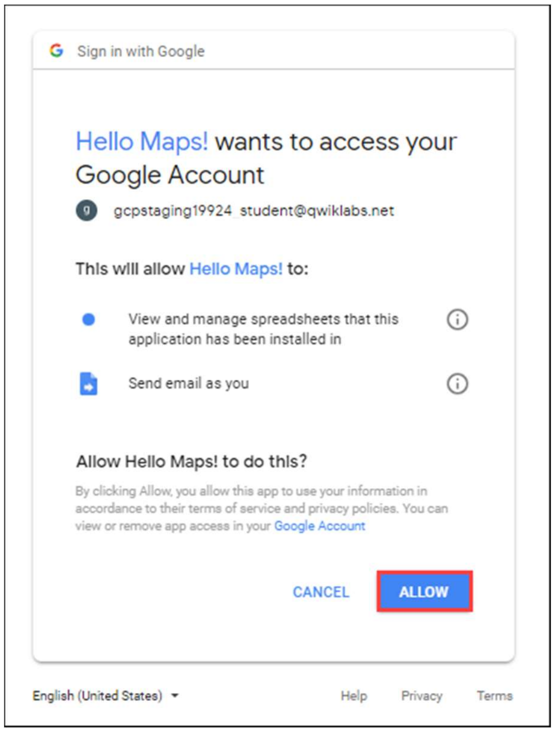
Choose an account from qwiklabs.net

to continue to Hello Maps!

gcpstaging19924_student@qwiklabs.net

Use another account

5. Now you get the *real* OAuth2 dialog window asking for permission to access your Sheet as well as send email on your behalf:



Sign in with Google

Hello Maps! wants to access your Google Account

gcpstaging19924 student@qwiklabs.net

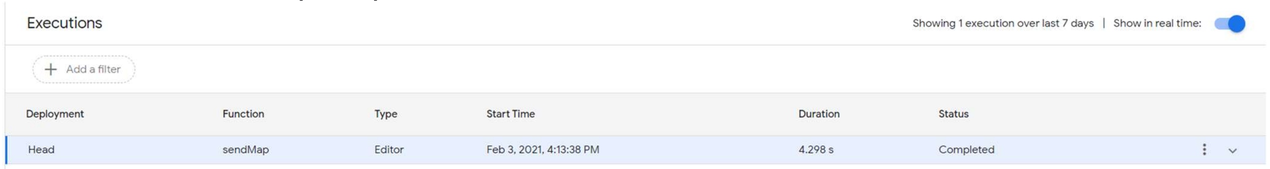
This will allow Hello Maps! to:

- View and manage spreadsheets that this application has been installed in
- Send email as you

Allow Hello Maps! to do this?

By clicking Allow, you allow this app to use your information in accordance to their terms of service and privacy policies. You can view or remove app access in your Google Account

6. After you grant permission, the script runs to completion.
7. Hover over to the left side and click on **Executions** to see `sendMap` listed. Click **View Dashboard** if prompted.

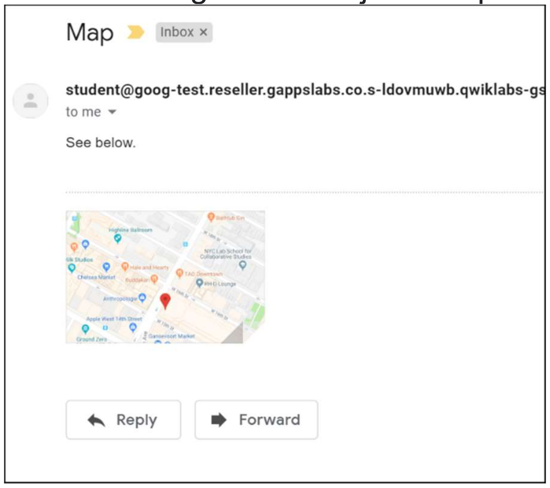


Executions

Showing 1 execution over last 7 days | Show in real time:

Deployment	Function	Type	Start Time	Duration	Status
Head	sendMap	Editor	Feb 3, 2021, 4:13:38 PM	4.298 s	Completed

8. Now check the email account where you sent your message. You should find a message with Subject "Map" and a message body that looks like this:



Map Inbox ×

student@goog-test.reseller.gappslabs.co.s-ldovmuwb.qwiklabs-gs

to me

See below.

Isn't that cool? Just think about it... you have four lines of code that access three different Google products in a meaningful way, even though it's not a complete application by itself. If you're unfamiliar with JavaScript or Apps Script, the code should be readable enough that you should have a rough idea how it works, and perhaps what Apps Script can accomplish for you.

Application - detailed explanation

This section reviews the code in more detail.

Since this application is short, there's no overall code structure to discuss. Instead,this section reviews each line of this app, which touches three different Google products!

1. This is a normal JavaScript function declaration for `sendMap()`.

```
function sendMap() {content_copy
```

2. The first line of code calls the Spreadsheet Service accessible from Apps Script via the SpreadsheetApp object. The returned sheet is assigned to a variable of the same name. The getActiveSheet() method does exactly what it says it does—it returns a "handle" to the current sheet that is active in the user interface (UI).

```
var sheet = SpreadsheetApp.getActiveSheet();content_copy
```

3. With the `sheet` object, reference the cell range (of a single cell) in A1 notation with getRange(). A "range" is a group of cells, including just a single one like ours... cell `A1`, the one we entered the address in. Now let's fetch what's *inside* that range of cells with the getValue() call, and assigned to the address variable upon return. Try adding more addresses and reading from different cells.

```
var address = sheet.getRange("A1").getValue();content_copy
```

4. The 3rd line connects to the Google Maps Service via the Maps object. As soon as we have access to the Maps Service, we request a new static map be created via newStaticMap(). You can then put a "pin" dropped on the address we pulled from the Sheet by using the addMarker() method.

```
var map = Maps.newStaticMap().addMarker(address);content_copy
```

5. The last line uses the Mail Service (via the GmailApp object), calling its sendEmail() method, to send the email which includes both the text "See below." and the map image as an attachment.

```
GmailApp.sendEmail("friend@example.com", "Map", 'See below.', {attachments:[map]});
}content_copy
```

Congratulations!

You used the Google Apps Script to write code that accesses Google developer technologies to extract an address in Google Sheet, generate a Google Map based on that address, and send the map to an email recipient.

Finish Your Quest

This self-paced lab is part of the Workspace Integrations Quest. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge public and link to them in your online resume or social media account. Enroll in this Quest and get immediate completion credit if you've taken this lab. See other available Qwiklabs Quests.

Take your next lab

Continue your Quest, or check out these suggestions:

- Build a Nearby Business Search Service with Google Maps Platform
- The Apps Script CLI - clasp

Next steps / learn more

- Read the Google Sheets API developer documentation.
- Post questions and find answers on Stackoverflow under the google-sheets-api tag.

Additional resources

The code featured in this lab is also available at its GitHub repo at GitHub.com/googlecodelabs/apps-script-intro. (This lab aims to stay in-sync with the repo.) Below are additional resources to help you dig deeper into the material covered in this lab as well as explore other ways of accessing Google developer tools programmatically.

Documentation

- Google Apps Script documentation site
- Apps Script Gmail Service
- Apps Script Spreadsheet Service
- Apps Script Maps Service

Related and general videos

- Others in Google Apps Script video library
- Workspace Dev Show video series

News & updates

- Workspace developers blog
- Workspace developers Twitter (@GSuiteDevs)
- Workspace developers monthly newsletter
- Google Workspace Learning Center

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. Our classes include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. Certifications help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated February 16, 2021

Lab Last Tested February 3, 2021