LEARNING google-apps-script

#google-apps-script

Table of Contents

About

Chapter 1: Getting started with google-apps-script

Remarks

The official overview for Google Apps Script is published at http://www.google.com/script/start, from there

> Google Apps Script is a JavaScript cloud scripting language that provides easy ways to automate tasks across Google products and third party services and build web applications.

From https://developers.google.com/apps-script/guides/services/#basic_javascript_features

> Apps Script is based on JavaScript 1.6, plus a few features from 1.7 and 1.8. Many basic JavaScript features are thus available in addition to the built-in and advanced Google services: you can use common objects like Array, Date, RegExp, and so forth, as well as the Math and Object global objects. However, because Apps Script code runs on Google's servers (not client-side, except for HTML-service pages), browser-based features like DOM manipulation or the Window API are not available.

Examples

Installation or Setup

Google Apps Script does not require setup or installation. The only requirement is a Google Account. A Gmail account works as well as a Google Apps for Work/Education/Government account. You can create a new Google account by going to accounts.google.com

Start your first script by going to script.google.com. You can also access Google Apps Script under the `tools -> Script editor...` of many Google Apps i.e *Docs, Sheets, Forms etc*. Google Apps Script can also be added directly to your Google Drive with the `Connect more apps..` feature.

Official documentation can be found at developers.google.com/apps-script/.

For app-scripts to run they must contain a code.gs file. The code.gs file must contain a function named doGet (standalone scripts) or an onOpen function (addon scripts). The quick starts in the documentation contain examples.

If an api is turned on in the app-script it must also be turned on in the developers-console. However, the developers console contains api's that can be turned on but do not appear in the app-script interface. For example, Marketplace SDK needs to be turned on in the developers console before the app can be published to the Google play store or to a G suite domain wide deployment.

For Google apps for education/work/government there are settings in the domain admin console that can be adjusted to allow or disallow app-scripts to run.

Types of Scripts

Google App scripts are of three types.

- Standalone
- Bound to Google Apps
- Web Apps

Standalone script

Standalone scripts are not bound to any Google apps i.e *Docs, Sheets or Forms etc.* Standalone script can either be created by visiting *script.google.com* or by connecting Google app script with Google drive. Standalone script can be used to program Google apps independently, can be used as a Web App or can be set up to run automatically from an installable trigger. See the documentation for standalone script.

Bound to Google Apps

Script bound to Google Apps also known as container-bound script; unlike standalone scripts, are bound to Google apps i.e *Google Docs or Google Sheets etc.* Container bound script can be created by selecting `tools> Script editor` from Google App. Some features like dialogs, prompts, menus and sidebar are only provided by container-bound scripts. Furthermore, container-bound script is used to create Google Add-ons. See the documentation for container-bound scripts.

Web Apps

Google App Script can be used as web app as they can be accessed by browser. Web App can provide user interface on the browser and can make use of google apps i.e *docs, sheets etc.* Both standalone scripts and scripts bound to Google Apps can be turned into web apps. For any script to work as a web app, script has to meet two requirements:

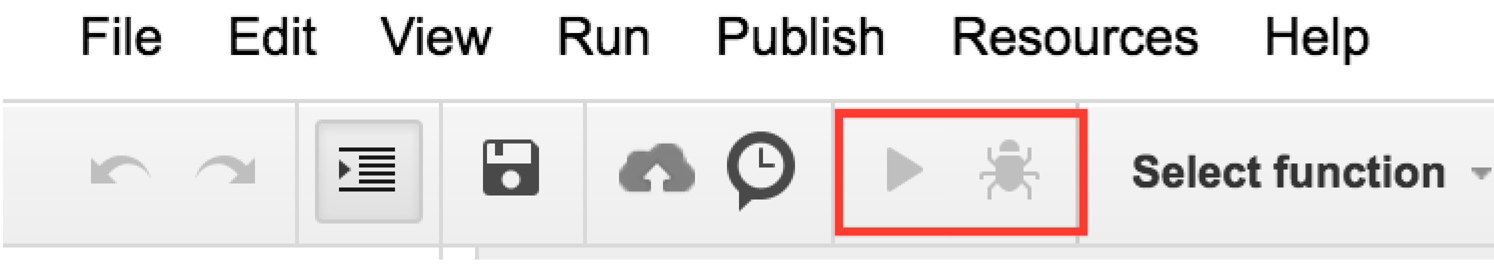
- include a `doGet()` or `doPost()` function.
- The function returns an HTML service HtmlOutput object or a Content service TextOutput object.

Inshort, `doGet()` and `doPost()` functions works like http get and post request handlers respectively.

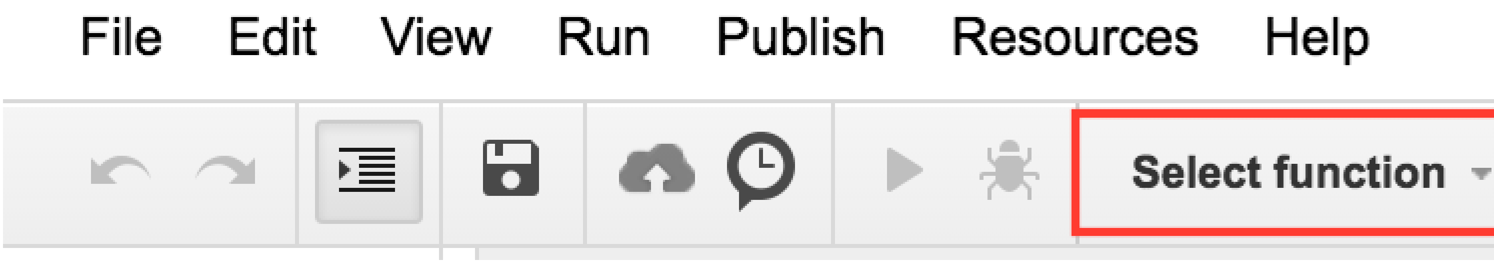
For more details on Web Apps, see the official documentation.

Running/Debugging your script

Try to run your code from the tool bar as shown below :



In your code, if you have more than one function then, before running it you should mention the function you want to run with. For example :



Alternatively, you can press **ctrl + r** from your keyboard to run the code. It will save the code first, if not saved, and then run it. But, for that to work, you must have selected the function, as seen in the image above.

Also, if your script is called by some external activities, still you will be able to see logs by clicking view->logs if you are logging anything after the code is executed.

Hello World

We are going to say Hello as a message box.

```
function helloWorld()
{
  Browser.msgBox("Hello World");
}
```

To execute the script, either click ▶ or select the menu item **Run** -> **helloWorld**

A deeper look at Google Apps Script

Google Apps Script is a JavaScript based platform-as-a-service primarily used to automate and extend Google Apps. Apps Script runs exclusively on Google's infrastructure requiring no server provisioning or configuration. An online IDE serves as the interface to the entire platform connecting all the services that are available to Apps Script. User authentication is baked into the platform via OAuth2 and requires no code or setup by the script author.

Apps Script runs server-side, but can have user interfaces built with Html, CSS, JavaScript, or any other browser supported technologies. Unlike Nodejs, which is event driven, App Scripts runs in a threaded model. All calls to a script generate a unique instance of that script which runs in isolation of all other instances. When an instance of a script finishes execution it is destroyed.

Functions in Apps Script are blocking so callback and async programming patterns are not needed. Locking is used to prevent critical sections of code, such as file IO, from being executed simultaneously by different instances.

In practice writing Apps Scripts are simple. Below is a simple script that creates a new spreadsheet from a template spreadsheet.

```
// Create a new spreadsheet from a template
function createSpreadsheet(){
   var templateFileId = '1Azcz9GwCeHjGl9TXf4aUh6g20Eqmgd1UMSdNVjzIZPk';
   var sheetName = 'Account Log for:' + new Date();
   SpreadsheetApp.openById(templateFileId).copy(sheetName);
}
```

Read Getting started with google-apps-script online: https://riptutorial.com/google-apps-script/topic/1154/getting-started-with-google-apps-script

Chapter 2: Apps Script Web Apps

Remarks

This is an example form web app, the client-side bit shows some basic UX design, such as a disabled submit button when the form is submitting, or an error message if it fails...etc

The Apps Script bit is very basic. It contains just the code necessary to serve up the html, and to validate the field.

Here is a link to this example app in action: Example Apps Script Form

Note: You must be signed into a Google account.

The Apps Script file structure is as so:

- Code.gs
- index.html
- Stylesheet.html
- JavaScript.html

Examples

Web App Form

Apps Script:

```
//Triggered when the page is navigated to, serves up HTML
function doGet(){
  var template = HtmlService.createTemplateFromFile('index');
  return template.evaluate()
      .setTitle('Example App')
      .setSandboxMode(HtmlService.SandboxMode.IFRAME);
}

//Called from the client with form data, basic validation for blank values
function formSubmit(formData){
  for(var field in formData){
    if(formData[field] == ''){
      return {success: false, message: field + ' Cannot be blank'}
    }
  }
  return {success: true, message: 'Sucessfully submitted!'};
}
```

HTML

```
<!DOCTYPE html>
<html>
```

```
    <head>
        <base target="_top">
        <link href="https://ssl.gstatic.com/docs/script/css/add-ons1.css" rel="stylesheet">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js"
type="text/javascript"></script>
    </head>

    <body>
        <div id="mainForm">
            <h1>Example Form</h1>
            <form>
                <div>
                    <div class="inline form-group">
                        <label for="name">Name</label>
                        <input id="nameInput" style="width: 150px;" type="text">
                    </div>
                </div>
                <div>
                    <div class="inline form-group">
                        <label for="city">City</label>
                        <input id="cityInput" style="width: 150px;" type="text">
                    </div>
                    <div class="inline form-group">
                        <label for="state">State</label>
                        <input id="stateInput" style="width: 40px;" type="text">
                    </div>
                    <div class="inline form-group">
                        <label for="zip-code">Zip code</label>
                        <input id="zip-codeInput" style="width: 65px;" type="number">
                    </div>
                </div>
                <div class="block form-group">
                    <label for="typeSelect">Type</label>
                    <select id="typeSelect">
                        <option value="">
                        </option>
                        <option value="Type 1 ">
                            Type 1
                        </option>
                        <option value="Type 2 ">
                            Type 2
                        </option>
                        <option value="Type 3 ">
                            Type 3
                        </option>
                        <option value="Type 4 ">
                            Type 4
                        </option>
                    </select>
                </div>
                <button class="action" id="submitButton" type="button">Submit</button>
                <button class="clear" id="clearFormButton" type="button">Clear Form</button>
            </form>
            <div class="hidden error message">
                <div class="title">Error:</div>
                <div class="message"></div>
            </div>
            <div class="hidden success message">
                <div class="title">Message:</div>
                <div class="message">Sucessfully submitted</div>
            </div>
        </div>
        <?!= HtmlService.createHtmlOutputFromFile('JavaScript').getContent(); ?>
        <?!= HtmlService.createHtmlOutputFromFile('Stylesheet').getContent(); ?>
    </body>

</html>
```

CSS

```
<style>
.hidden {
    display: none;
}

.form-group {
    margin: 2px 0px;
}

#submitButton {
    margin: 4px 0px;
}

body {
    margin-left: 50px;
}

.message {
   padding: 2px;
   width: 50%;
}

.message > * {
   display: inline-block;
}

.message .title {
    font-weight: 700;
    font-size: 1.1em;
}

.success.message {
    border: 1px solid #5c9a18;
    background: #e4ffe4;
    color: #2a8e2a;
}

.error.message {
    background: #f9cece;
    border: 1px solid #7d2929;
}

.error.message .title {
    color: #863030;
}

button.clear {
    background: -moz-linear-gradient(top, #dd6e39, #d17636);
    background: -ms-linear-gradient(top, #dd6e39, #d17636);
    background: -o-linear-gradient(top, #dd6e39, #d17636);
    background: -webkit-linear-gradient(top, #dd6e39, #d17636);
    background: linear-gradient(top, #dd6e39, #d17636);
    border: 1px solid transparent;
    color: #fff;
    text-shadow: 0 1px rgba(0, 0, 0, .1);
}

button.clear:hover {
    background: -moz-linear-gradient(top, #ca602e, #bd6527);
    background: -ms-linear-gradient(top, #ca602e, #bd6527);
    background: -o-linear-gradient(top, #ca602e, #bd6527);
    background: -webkit-linear-gradient(top, #ca602e, #bd6527);
    background: linear-gradient(top, #ca602e, #bd6527);
    border: 1px solid transparent;
    color: #fff;
    text-shadow: 0 1px rgba(0, 0, 0, .1);
}
</style>
```

JavaScript

```
<script>
var inputs = [
  'nameInput',
  'cityInput',
  'stateInput',
  'zip-codeInput',
  'typeSelect'
];

$(function(){
  var pageApp = new formApp();
  $('#submitButton').on('click', pageApp.submitForm);
  $('#clearFormButton').on('click', pageApp.clearForm);
});

var formApp = function(){
  var self = this;

  //Clears form input fields, removes message, enables submit
  self.clearForm = function(){
    for(var i = 0; i < inputs.length; i++){
        $('#'+inputs[i]).val('');
    }
    toggleSubmitButton(false);
    setErrorMessage(false);
    setSuccessMessage(false);
  }

  //Submits the form to apps script
  self.submitForm = function(){
    toggleSubmitButton(true);
    setSuccessMessage(false);
    setErrorMessage(false);

    google.script.run
        .withSuccessHandler(self.sucessfullySubmitted)
        .withFailureHandler(self.failedToSubmit)
        .formSubmit(self.getFormData());
  };
  //Retrieves the form data absed on the input fields
  self.getFormData = function(){
    var output = {};
    for(var i = 0; i < inputs.length; i++){
        output[inputs[i]] = $('#'+inputs[i]).val();
    }
    console.log(output)
    return output;
  }

  //When the apps script sucessfully returns
  self.sucessfullySubmitted = function(value){
    if(value.success){
      setSuccessMessage(true, value.message);
    } else {
      setErrorMessage(true, value.message);
      toggleSubmitButton(false);
    }
  }

  //When the apps script threw an error
  self.failedToSubmit = function(value){
    toggleSubmitButton(false);
    setErrorMessage(true, value.message);
  }
}

//Disables/enables the submit button
function toggleSubmitButton(disabled){
  $('#submitButton').prop('disabled', disabled);
}

//Sets the general message box's message and enables or disabled the error box
function setSuccessMessage(show, message){
  if(show){
    $('.success.message').removeClass('hidden');
    $('.success.message .message').text(message);
  } else {
    $('.success.message').addClass('hidden');
    $('.success.message .message').text('');
  }
}

//Sets the error message box's message and enables or disabled the error box
function setErrorMessage(show, message){
  if(show){
    $('.error.message').removeClass('hidden');
    $('.error.message .message').text(message);
  } else {
    $('.error.message').addClass('hidden');
    $('.error.message .message').text('');
  }
}

function getFormData(){
  var output = {};
  for(var i = 0; i < inputs.length; i++){
      output[inputs[i]] = $('#'+inputs[i]).val();
  }
  return output;
}
```

```
</script>
```

Read Apps Script Web Apps online: https://riptutorial.com/google-apps-script/topic/4874/apps-script-web-apps

Chapter 3: Client calls to Google apps-script

Introduction

Google appscript runs well as a stand alone platform and in the addon format for Google docs, sheets and forms. However, there are times when a client browser may need to call to a Google app to perform some action.

Therefore, Google introduced client side requests to Google apps-scripts. To solve this problem, Google introduced the client side libraries

Examples

This is an example of a client side call to a Google app-script

```
<script src="https://apis.google.com/js/api.js"></script>
<script>
function start() {
 // 2. Initialize the JavaScript client library.
 gapi.client.init({
'apiKey': 'YOUR_API_KEY',
// clientId and scope are optional if auth is not required.
'clientId': 'YOUR_WEB_CLIENT_ID.apps.googleusercontent.com',
'scope': 'profile',
}).then(function() {
// 3. Initialize and make the API request.
return gapi.client.request({
  'path': 'https://people.googleapis.com/v1/people/me',
  })
 }).then(function(response) {
console.log(response.result);
}, function(reason) {
console.log('Error: ' + reason.result.error.message);
});
};
// 1. Load the JavaScript client library.
 gapi.load('client', start);
</script>
```

Read Client calls to Google apps-script online: https://riptutorial.com/google-apps-script/topic/8875/client-calls-to-google-apps-script

Chapter 4: Create a custom function for Google Sheets

Introduction

A custom function in google docs is tied to a specific document (and thus can only be used in that document).

It must therefore be created with that document's scrip edit (Tools -> Script Editor). Once saved, it can then be used like any other regular spreadsheet formula.

Examples

Standard gravity custom constant

This function return the standart gravity constant in the specified acceleration units (1 for cm/s², 2 for ft/s², 3 for m/s²)

```
/**
 * Returns the standard gravity constant in the specified acceleration units
 * Values taken from https://en.wikipedia.org/wiki/Standard_gravity on July 24, 2016.
 *
 * @param  {number}  input  1 for cm/s², 2 for ft/s², 3 for m/s²
 *
 * @customfunction
 */
function sg(units_key) {
  var value;
  switch(units_key) {
    case 1:
      value = 980.665;
      break;
    case 2:
      value = 32.1740;
      break;
    case 3:
      value = 9.80665;
      break;
    default:
      throw new Error('Must to specify 1, 2 or 3');
  }
  return value;
}
```

To use the function, it needs to be bound to a spreadsheet using the script editor (Tools -> Script editor...). Once the function is added, it can be used like any other google sheets function by calling the function in a cell's formula.

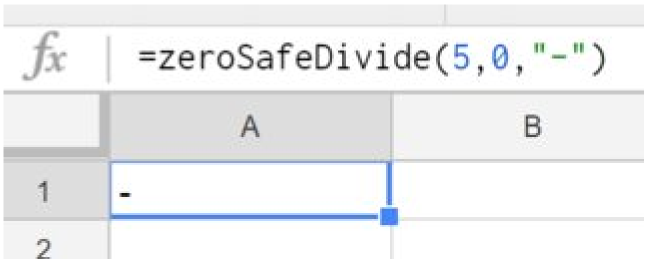
Note how the function shows up in autocomplete when typed into a formula. This is due to the multi-line comment above the function declaration which is used to describe what the function does similar to JSDoc and Javadoc. To have the formula show up in autocomplete, the @customfunction tag must be specified in the comment.

Basic Example

To avoid unsightly `#DIV/0` errors in a spreadsheet, a custom function can be used.

```
/**
 * Divides n by d unless d is zero, in which case, it returns
 * the given symbol.
 *
 * @param  {n}  number The numerator
 * @param  {d}  number The divisor
 * @param  {symbol}  string The symbol to display if `d == 0`
 * @return {number or string} The result of division or the given symbol
 *
 * @customfunction
 */
function zeroSafeDivide(n, d, symbol) {
  if (d == 0)
    return symbol;
  else
    return n / d;
}
```

To use the function, it needs to be bound to a spreadsheet using the script editor (**Tools -> Script editor...**). Once the function is added, it can be used like any other google sheets function by calling the function in a cell's formula.



```
=zeroSafeDivide(5,0,"-")
```

	A	B
1	-	
2		

Note how the function shows up in autocomplete when typed into a formula. This is due to the multi-line comment above the function declaration which is used to describe what the function does similar to JSDoc and Javadoc. To have the formula show up in autocomplete, the `@customfunction` tag must be specified in the comment.

Read Create a custom function for Google Sheets online: https://riptutorial.com/google-apps-script/topic/5572/create-a-custom-function-for-google-sheets

Chapter 5: DriveApp

Examples

Create a new folder in a Google Drive root

```
function createNewFolderInGoogleDrive(folderName) {
  return DriveApp.createFolder(folderName);
}
```

Use function `createNewFolderInGoogleDrive` to create folder named `Test folder` in a Google Drive root:

```
 var newFolder = createNewFolderInGoogleDrive('Test folder');
```

`newFolder` has Class Folder type:

```
// output id of new folder to log
Logger.log(newFolder.getId());
```

Create new file in Google Drive of a certain Mime type

```
 function createGoogleDriveFileOfMimeType() {
  var content,fileName,newFile;//Declare variable names

  fileName = "Test File " + new Date().toString().slice(0,15);//Create a new file name with
date on end
  content = "This is the file Content";

  newFile = DriveApp.createFile(fileName,content,MimeType.JAVASCRIPT);//Create a new file in
the root folder
};
```

Create a new text file in Google Drive root folder

```
 function createGoogleDriveTextFile() {
  var content,fileName,newFile;//Declare variable names

  fileName = "Test Doc " + new Date().toString().slice(0,15);//Create a new file name with
date on end
  content = "This is the file Content";

  newFile = DriveApp.createFile(fileName,content);//Create a new text file in the root folder
};
```

Create a new file in Google drive from a blob

```
 function createGoogleDriveFileWithBlob() {
  var blob,character,data,fileName,i,L,max,min,newFile,randomNmbr;//Declare variable names

  fileName = "Test Blob " + new Date().toString().slice(0,15);//Create a new file name with
date on end

  L = 500;//Define how many times to loop
  data = "";
  max = 126;
  min = 55;

  for (i=0;i<L;i+=1) {//Loop to create data
    randomNmbr = Math.floor(Math.random()*(max-min+1)+min);//Create a random number
    //Logger.log('randomNmbr: ' + randomNmbr);
    character = String.fromCharCode(randomNmbr);

    //Logger.log('character: ' + character);//Print the character to the Logs
    data = data + character;
  };

  blob = Utilities.newBlob(data, MimeType.PLAIN_TEXT, fileName);//Create a blob with random
characters

  newFile = DriveApp.createFile(blob);//Create a new file from a blob

  newFile.setName(fileName);//Set the file name of the new file
};
```

Get all folders - put folders into a continuation token - then retrieve from token

```
 function processGoogleDriveFolders() {
  var arrayAllFolderNames,continuationToken,folders,foldersFromToken,thisFolder;//Declare
variable names

  arrayAllFolderNames = [];//Create an empty array and assign it to this variable name

  folders = DriveApp.getFolders();//Get all folders from Google Drive in this account
  continuationToken = folders.getContinuationToken();//Get the continuation token

  Utilities.sleep(18000);//Pause the code for 3 seconds

  foldersFromToken = DriveApp.continueFolderIterator(continuationToken);//Get the original
folders stored in the token
  folders = null;//Delete the folders that were stored in the original variable, to prove that
the continuation token is working

  while (foldersFromToken.hasNext()) {//If there is a next folder, then continue looping
    thisFolder = foldersFromToken.next();//Get the next folder
    arrayAllFolderNames.push(thisFolder.getName());//Get the name of the next folder
  };

  Logger.log(arrayAllFolderNames);//print the folder names to the Logs
};
```

Get all files - put them into a continuation token - then retrieve them

```
 function processGoogleDriveFiles() {
  var arrayAllFileNames,continuationToken,files,filesFromToken,fileIterator,thisFile;//Declare
```

```
variable names

  arrayAllFileNames = [];//Create an empty array and assign it to this variable name

  files = DriveApp.getFiles();//Get all files from Google Drive in this account
  continuationToken = files.getContinuationToken();//Get the continuation token

  Utilities.sleep(18000);//Pause the code for 3 seconds

  filesFromToken = DriveApp.continueFileIterator(continuationToken);//Get the original files
stored in the token
  files = null;//Delete the files that were stored in the original variable, to prove that the
continuation token is working

  while (filesFromToken.hasNext()) {//If there is a next file, then continue looping
    thisFile = filesFromToken.next();//Get the next file
    arrayAllFileNames.push(thisFile.getName());//Get the name of the next file
  };

  Logger.log(arrayAllFileNames);
};
```

Add a folder to the root drive

```
function DriveAppAddFolder(child) {//Adds file to the root drive in Google Drive
  var body,returnedFolder;//Declare variable names

  if (!child) {
    body = "There is no folder";
    MailApp.sendEmail(Session.getEffectiveUser().getEmail(), "", "Error Adding Folder!", body)
    return;
  };

  returnedFolder = DriveApp.addFolder(child);//Add a folder to the root drive

  Logger.log('returnedFolder: ' + returnedFolder);//Print the folder results to the Logs
};


function createNewFolderInGoogleDrive() {
  var folder,newFolderName,timeStamp,dateTimeAsString;

  timeStamp = new Date();//Create a new date
  dateTimeAsString = timeStamp.toString().slice(0,15);

  newFolderName = 'Test Folder Name ' + dateTimeAsString;//Create new folder name with
date/time appended to name

  folder = DriveApp.createFolder(newFolderName);//Create a new folder
  DriveAppAddFolder(folder);//Call a function and pass a folder to the function
};
```

Create a new text file and add it to the root folder

```
 function DriveAppAddFile(child) {//Adds file to the root drive in Google Drive
  var body,returnedFolder;//Declare variable names
  if (!child) {
    body = "There is no file";
    MailApp.sendEmail(Session.getEffectiveUser().getEmail(), "", "Error Adding File!", body)
    return;
  };

  returnedFolder = DriveApp.addFile(child);

  Logger.log('returnedFolder: ' + returnedFolder);
};


function createNewFileInGoogleDrive() {
  var content,file,newFileName,timeStamp,dateTimeAsString;

  timeStamp = new Date();//Create a new date
  dateTimeAsString = timeStamp.toString().slice(0,15);

  content = "This is test file content, created at: " + dateTimeAsString;//Create content for
new file
  newFileName = 'Test File ' + dateTimeAsString;//Create new file name with date/time appended
to name

  file = DriveApp.createFile(newFileName, content);//Create a new file
  DriveAppAddFile(file);//Call a function and pass a file to the function
};
```

Get all Files in a Drive Folder

```
function onOpen() {

  // Add a custom menu to run the script
  var ss = SpreadsheetApp.getActiveSpreadsheet();
  var searchMenuEntries = [ {name: "Run", functionName: "search"}];
  ss.addMenu("Get Files", searchMenuEntries);
}

function getFiles() {

  // Get the active spreadsheet and the active sheet
  var ss = SpreadsheetApp.getActiveSpreadsheet();
  var ssid = ss.getId();

  // Look in the same folder the sheet exists in. For example, if this template is in
  // My Drive, it will return all of the files in My Drive.
  var ssparents = DriveApp.getFileById(ssid).getParents();
  var sheet = ss.getActiveSheet();

  // Set up the spreadsheet to display the results
  var headers = [["Last Updated", "File Owner", "File Name", "File URL"]];
  sheet.getRange("A1:D").clear();
  sheet.getRange("A1:D1").setValues(headers);

  // Loop through all the files and add the values to the spreadsheet.
  var folder = ssparents.next();
  var files = folder.getFiles();
  var i=1;
  while(files.hasNext()) {
    var file = files.next();
    if(ss.getId() == file.getId()){
      continue;
    }
    sheet.getRange(i+1, 1, 1,
4).setValues([[file.getLastUpdated(),file.getOwner().getName(),file.getName(),
file.getUrl()]]);
    i++;
  }
}
```

Read DriveApp online: https://riptutorial.com/google-apps-script/topic/5363/driveapp

Chapter 6: DriveApp - getFileById(id)

Remarks

It is also possible to get a file by the file's URL. The ID of a file is in the url, so using the ID instead of the entire URL means that the parameter is shorter. Storing the URL rather than the ID takes up more space.

Examples

Get a file from Google Drive using the file ID

```
function getGoogleDriveFileById(id) {
  var file;

  file = DriveApp.getFileById(id);//Returns a file - The "id" must be a string

  //One way to manually get a file ID
  // - Open the file from Google Drive
  // - The file ID is in the URL in the browsers address bar
  //https://docs.google.com/spreadsheets/d/File_ID_is_here/edit#gid=0
};
```

Read DriveApp - getFileById(id) online: https://riptutorial.com/google-apps-script/topic/6087/driveapp---getfilebyid-id-

Chapter 7: DriveApp Service

Remarks

Google Mime types can not be used for the third parameter of Mime Types. Using a Google Mime Type will result in an error that states:

Cannot use "DriveApp.createFile()" to create Google MIME types. Please use Advanced Drive Service

MimeType.GOOGLE_APPS_SCRIPT

MimeType.GOOGLE_DOCS

MimeType.GOOGLE_DRAWINGS

MimeType.GOOGLE_FORMS

MimeType.GOOGLE_SHEETS

MimeType.GOOGLE_SLIDES

Examples

Create a new folder in Google root drive

```
function createNewFolderInGoogleDrive() {
  var folderName,newFolder;//Declare variable names

  folderName = "Test Folder " + new Date().toString().slice(0,15);//Create a new folder name
with date on end
  newFolder = DriveApp.createFolder(folderName);//Create a new folder in the root drive
};
```

Create new file in Google Drive of a certain Mime type

```
function createGoogleDriveFileOfMimeType() {
  var content,fileName,newFile;//Declare variable names

  fileName = "Test File " + new Date().toString().slice(0,15);//Create a new file name with
date on end
  content = "This is the file Content";

  newFile = DriveApp.createFile(fileName,content,MimeType.JAVASCRIPT);//Create a new file in
the root folder
};
```

Create a new text file in Google root drive folder

```
function createGoogleDriveTextFile() {
  var content,fileName,newFile;//Declare variable names

  fileName = "Test Doc " + new Date().toString().slice(0,15);//Create a new file name with
date on end
  content = "This is the file Content";

  newFile = DriveApp.createFile(fileName,content);//Create a new text file in the root folder
};
```

Create a new file in Google Drive from a blob

```
function createGoogleDriveFileWithBlob() {
  var blob,character,data,fileName,i,L,max,min,newFile,randomNmbr;//Declare variable names

  fileName = "Test Blob " + new Date().toString().slice(0,15);//Create a new file name with
date on end

  L = 500;//Define how many times to loop
  data = "";
  max = 126;
  min = 55;

  for (i=0;i<L;i+=1) {//Loop to create data
    randomNmbr = Math.floor(Math.random()*(max-min+1)+min);//Create a random number
    //Logger.log('randomNmbr: ' + randomNmbr);
    character = String.fromCharCode(randomNmbr);

    //Logger.log('character: ' + character);//Print the character to the Logs
    data = data + character;
  };

  blob = Utilities.newBlob(data, MimeType.PLAIN_TEXT, fileName);//Create a blob with random
characters

  newFile = DriveApp.createFile(blob);//Create a new file from a blob

  newFile.setName(fileName);//Set the file name of the new file
};
```

Get all folders - put folders into a continuation token - then retrieve from token

```
function processGoogleDriveFolders() {
  var arrayAllFolderNames,continuationToken,folders,foldersFromToken,thisFolder;//Declare
variable names

  arrayAllFolderNames = [];//Create an empty array and assign it to this variable name

  folders = DriveApp.getFolders();//Get all folders from Google Drive in this account
  continuationToken = folders.getContinuationToken();//Get the continuation token

  Utilities.sleep(18000);//Pause the code for 3 seconds

  foldersFromToken = DriveApp.continueFolderIterator(continuationToken);//Get the original
folders stored in the token
  folders = null;//Delete the folders that were stored in the original variable, to prove that
the continuation token is working
```

```
  while (foldersFromToken.hasNext()) {//If there is a next folder, then continue looping
    thisFolder = foldersFromToken.next();//Get the next folder
    arrayAllFolderNames.push(thisFolder.getName());//Get the name of the next folder
  };

  Logger.log(arrayAllFolderNames);//print the folder names to the Logs
};
```

Get all files - put them into a continuation token - then retrieve them

```
function processGoogleDriveFiles() {
  var arrayAllFileNames,continuationToken,files,filesFromToken,fileIterator,thisFile;//Declare
variable names

  arrayAllFileNames = [];//Create an empty array and assign it to this variable name

  files = DriveApp.getFiles();//Get all files from Google Drive in this account
  continuationToken = files.getContinuationToken();//Get the continuation token

  Utilities.sleep(18000);//Pause the code for 3 seconds

  filesFromToken = DriveApp.continueFileIterator(continuationToken);//Get the original files
stored in the token
  files = null;//Delete the files that were stored in the original variable, to prove that the
continuation token is working

  while (filesFromToken.hasNext()) {//If there is a next file, then continue looping
    thisFile = filesFromToken.next();//Get the next file
    arrayAllFileNames.push(thisFile.getName());//Get the name of the next file
  };

  Logger.log(arrayAllFileNames);
};
```

Read DriveApp Service online: https://riptutorial.com/google-apps-script/topic/6395/driveapp-service

Chapter 8: DriveApp Service - Files by type and search string

Parameters

Parameter Name	Use For
searchString	the string to be found in the file name

Examples

Get files by file type with matching string in file name

Get all Google Forms with the word "Untitled" in the file name.

```
function mainSearchFunction(searchStr) {
  var fileInfo,arrayFileIDs,arrayFileNames,arrayOfIndexNumbers,
      allFileIDsWithStringInName,i,searchStr,thisID;//Declare variables

  if (!searchStr) {
    searchStr = "Untitled";//Assign a string value to the variable
  };

  fileInfo = getFilesOfType();//Run a function that returns files information
  arrayFileNames = fileInfo[1];//Get the array of file names
  arrayOfIndexNumbers = searchFileNamesForString(arrayFileNames,searchStr);

  //Logger.log('searchStr: ' + searchStr)
  //Logger.log(arrayOfIndexNumbers)

  allFileIDsWithStringInName = [];
  arrayFileIDs = fileInfo[0];

  for (i=0;i<arrayOfIndexNumbers.length;i+=1) {
    thisID = arrayFileIDs[arrayOfIndexNumbers[i]];
    allFileIDsWithStringInName.push(thisID);
  };

  Logger.log(allFileIDsWithStringInName)
};

function getFilesOfType() {
  var allFormFiles,arrFileName,arrFileID,arrFileUrls,thisFile;

  allFormFiles = DriveApp.getFilesByType(MimeType.GOOGLE_FORMS);
  arrFileName = [];
  arrFileID = [];
  arrFileUrls = [];

  while (allFormFiles.hasNext()) {
    thisFile=allFormFiles.next();
    arrFileName.push(thisFile.getName());
    arrFileID.push(thisFile.getId());
    arrFileUrls.push(thisFile.getUrl());
  };

  //Logger.log(arrFileName)
  return [arrFileID,arrFileName];
};


function searchFileNamesForString(arrayFileNames,searchStr) {
  var arrayIndexNumbers,i,L,thisName;

  arrayIndexNumbers = [];

  L = arrayFileNames.length;

  for (i=0;i<L;i+=1){
    thisName = arrayFileNames[i];
    Logger.log(thisName);
    Logger.log('thisName.indexOf(searchStr): ' + thisName.indexOf(searchStr));

    if (thisName.indexOf(searchStr) !== -1) {
      arrayIndexNumbers.push(i);
    };
  };

  return arrayIndexNumbers;
};
```

Read DriveApp Service - Files by type and search string online: https://riptutorial.com/google-apps-script/topic/4049/driveapp-service---files-by-type-and-search-string

Chapter 9: Firebase and AppScript : Introduction

Introduction

Integrate Firebase with Google AppScript to Read and Write Data in the Firebase Database.

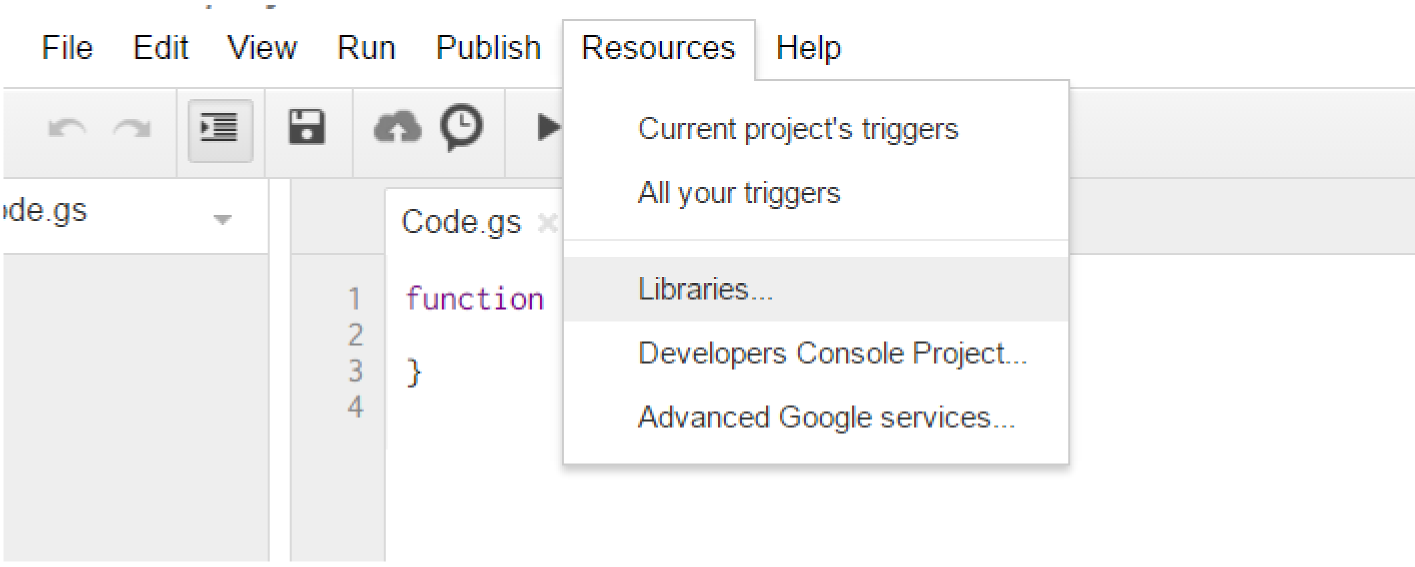
Firebase is a NoSQL database system by Google that uses realtime database to help create and host applications on mobile, desktop and tablet devices. NoSQL databases use the JSON objects to store the data in structured format.

Examples

Connecting to a Firebase project in GAS and transferring data from Google Spreadsheet to Firebase

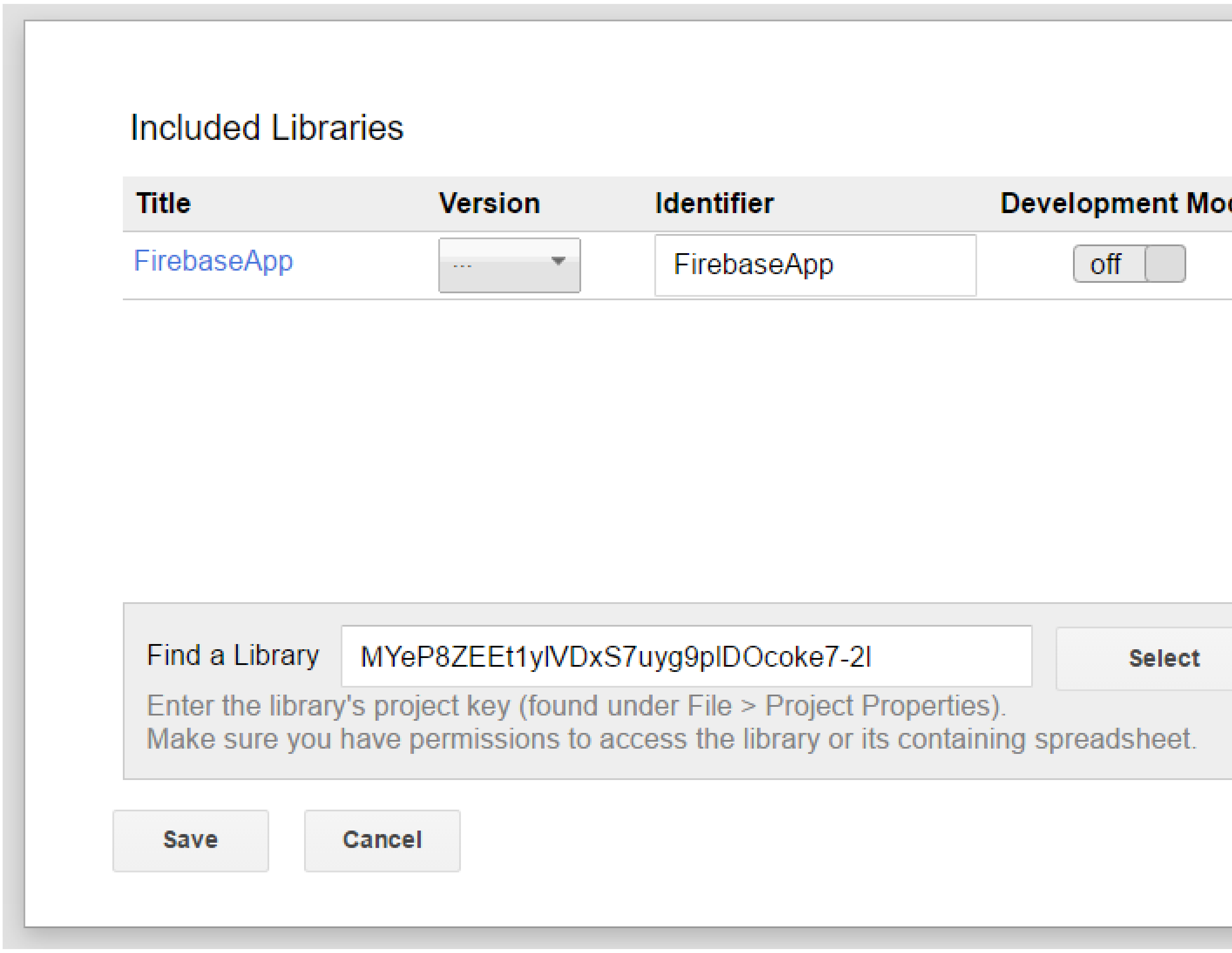
Install Firebase resource in the the AppScript

- To do that click on Resources and then on Libraries.
- Firebase has a unique project library key that need to be installed in the AppScript.


File
Edit
View
Run
Publish
Resources
Help
Current project's triggers
All your triggers
Libraries...
Developers Console Project...
Advanced Google services...
ıde.gs
Code.gs

```
function
}
```

- Click on Libraries The following pop-up appears. Enter the following project key in the textbox. **MYeP8ZEEt1ylVDxS7uyg9plDOcoke7-2l** This is the project library key for Firebase.



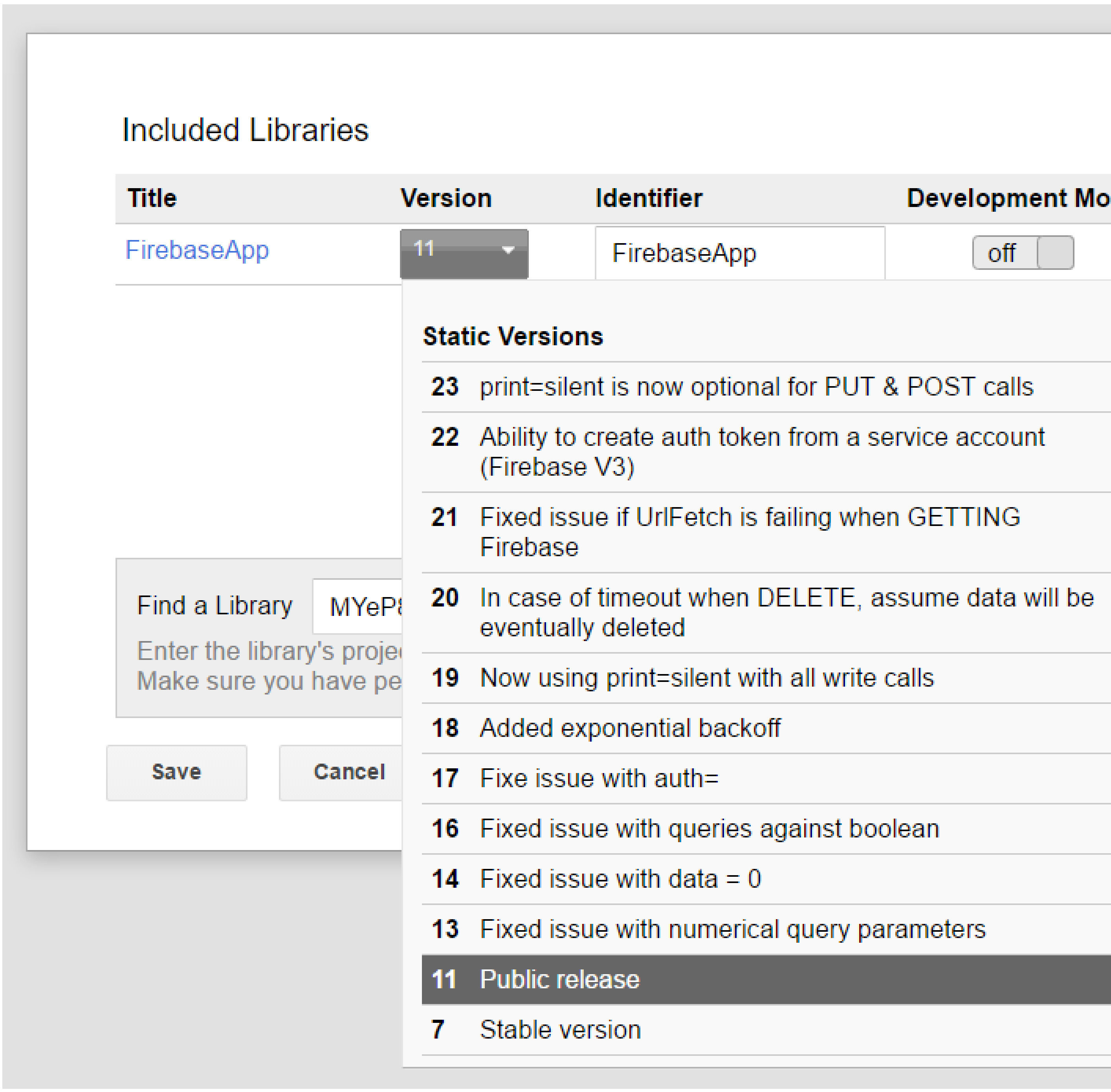
Included Libraries

Title	Version	Identifier	Development Mo
FirebaseApp	---	FirebaseApp	off

Find a Library: MYeP8ZEEt1ylVDxS7uyg9plDOcoke7-2l

Enter the library's project key (found under File > Project Properties).
Make sure you have permissions to access the library or its containing spreadsheet.

- Now in the version choose the stable public release version.


Included Libraries

Title	Version	Identifier	Development Mo
FirebaseApp	11	FirebaseApp	off

Find a Library MYeP

Enter the library's proje
Make sure you have pe

Save Cancel

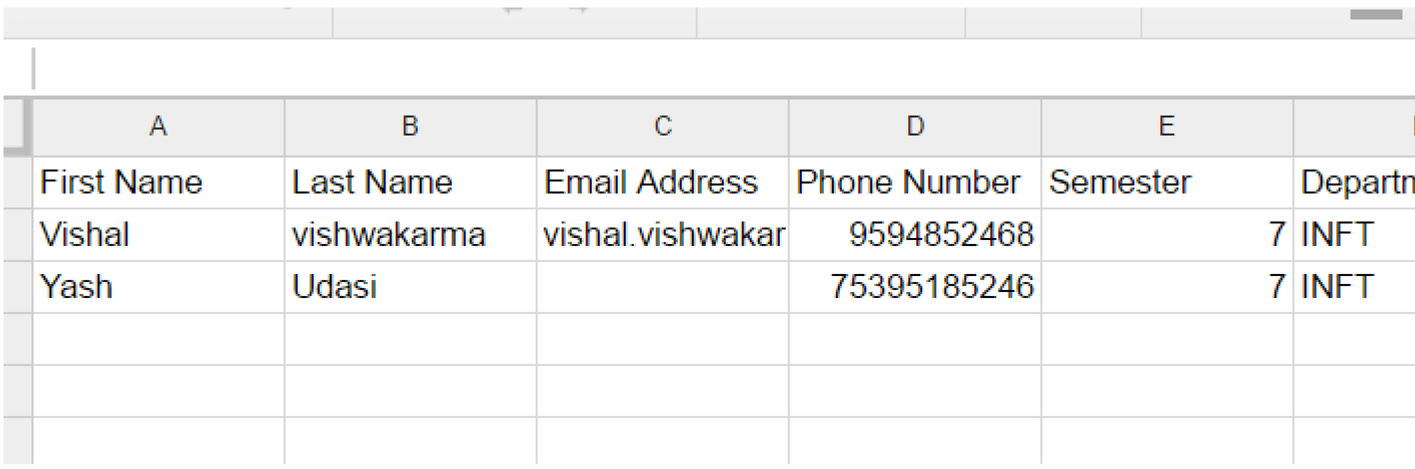
Static Versions

23	print=silent is now optional for PUT & POST calls
22	Ability to create auth token from a service account (Firebase V3)
21	Fixed issue if UrlFetch is failing when GETTING Firebase
20	In case of timeout when DELETE, assume data will be eventually deleted
19	Now using print=silent with all write calls
18	Added exponential backoff
17	Fixe issue with auth=
16	Fixed issue with queries against boolean
14	Fixed issue with data = 0
13	Fixed issue with numerical query parameters
11	Public release
7	Stable version

- Click on Save. Now Firebase is successfully installed in your AppScript for you to work.

Now let's take an example for reading and writing data from Firebase.

- Now we take a sample table designed in Google Sheets.



	A	B	C	D	E	
	First Name	Last Name	Email Address	Phone Number	Semester	Departn
	Vishal	vishwakarma	vishal.vishwakar	9594852468	7	INFT
	Yash	Udasi		75395185246	7	INFT

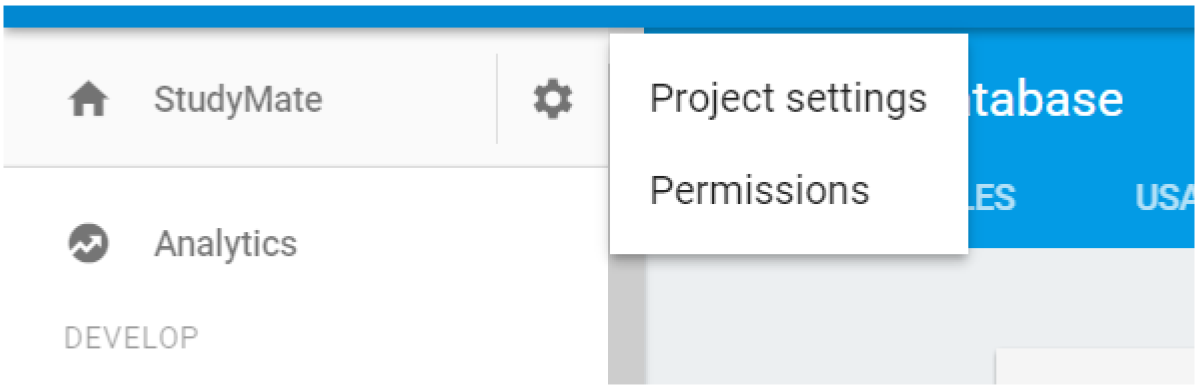
- Now to build the database in the Firebase using this table in the sheets. Add the following code in the AppScript.

```
function writeDataToFirebase() {
  var ss = SpreadsheetApp.openById("1LACsj0s3syAa9gvORdRWBhJ_YcXHybjQfHPgw3TLQ6g");
  var sheet = ss.getSheets()[0];
  var data = sheet.getDataRange().getValues();
  var dataToImport = {};
  for(var i = 1; i < data.length; i++) {
    var firstName = data[i][0];
    var lastName = data[i][1];
    dataToImport[firstName + '-' + lastName] = {
      firstName:firstName,
      lastName:lastName,
      emailAddress:data[i][2],
      semester:data[i][4],
      department:data[i][5],
    };
  }
  var firebaseUrl = "https://example-app.firebaseio.com/";
  var secret = "secret-key";
  var base = FirebaseApp.getDatabaseByUrl(firebaseUrl, secret);
  base.setData("", dataToImport);
}
```

Replace the spreadsheet ID and the firebaseURL and the secret key.

How to find the firebaseURL and the secret key?

- Go to your Firebase Dashboard and click on settings gear at top left corner. Click on Project Settings.



- Go to Service Accounts section you can find the databaseURL. This serves as the firebaseURL.
- Now click on Database Secrets tab and you can find the secret key.

Now you have inserted the firebaseURL and the secret key. Now you are all set to go. Click on run code in the AppScript engine.

- It will ask to review Permissions first time when you run.
- Click Review Permissions and Allow.
- Now you run your function and you can see the table created in Firebase Database.

To see the database go to the Firebase dashboard and Click on the Database you can view the database.

Some more functions to implement read and write.

1. To write a simple data to test whether connection is working or not.

```
function myFunction(){
 var firebaseUrl = "https://example-app.firebaseio.com/";
 var secret = "secret-key";
 var base = FirebaseApp.getDatabaseByUrl(firebaseUrl, secret);
 base.setData("test", "Hello Firebase");
}
```

2. To read all the Data

```
function getAllData() {
         var firebaseUrl = "https://example-app.firebaseio.com/";
         var secret = "secret-key";
var base = FirebaseApp.getDatabaseByUrl(firebaseUrl, secret);
 var data = base.getData();
          for(var i in data) {
            Logger.log(data[i].firstName + ' ' + data[i].lastName);
          }
}
```

The data read is displayed in the Logs. To check logs click on click on View → Logs or simply use Control + Enter.

3. To read a specific record

```
function getContact() {
  var firebaseUrl = "https://example-app.firebaseio.com/";
  var secret = "secret-key";
  var base = FirebaseApp.getDatabaseByUrl(firebaseUrl, secret);
  var contact = base.getData("Yash-Udasi");
  Logger.log(contact);
}
```

The data read is displayed in the Logs. To check logs click on click on View → Logs or simply use Control + Enter.

4. To update a specific record.

```
function updateData() {
  var firebaseUrl = "https://example-app.firebaseio.com/";
  var secret = "secret-key";
  var base = FirebaseApp.getDatabaseByUrl(firebaseUrl, secret);
  base.updateData("Yash-Udasi/emailAddress", "yash.udasi@fyuff.com");
}
```

Read Firebase and AppScript : Introduction online: https://riptutorial.com/google-apps-script/topic/9417/firebase-and-appscript---introduction

Chapter 10: GmailApp

Remarks

See also the official API reference for the GmailApp for further details on the available methods.

Examples

Get CSV file attached to a mail

Assume that we have a system that sends out daily reports over email in the form of attached CSV files and that we want to access these.

```
function getCsvFromGmail() {
  // Get the newest Gmail thread based on sender and subject
  var gmailThread = GmailApp.search("from:noreply@example.com subject:\"My daily report\"", 0,
1)[0];

  // Get the attachments of the latest mail in the thread.
  var attachments = gmailThread.getMessages()[gmailThread.getMessageCount() -
1].getAttachments();

  // Get and and parse the CSV from the first attachment
  var csv = Utilities.parseCsv(attachments[0].getDataAsString());
  return csv;
}
```

Read GmailApp online: https://riptutorial.com/google-apps-script/topic/5899/gmailapp

Chapter 11: Google sheets MailApp

Introduction

This service allows users to send emails with complete control over the content of the email. Unlike GmailApp, MailApp's sole purpose is sending email. MailApp cannot access a user's Gmail inbox.

Changes to scripts written using GmailApp are more likely to trigger a re-authorization request from a user than MailApp scripts.

Examples

A basic MailApp Example

MailApp is the api from Google App Script that can be used to send mail

```
function sendEmails() {

  var subject = "A subject for your new app!";
  var message = "And this is the very first message"
  var recipientEmail = "abc@example.com";

  MailApp.sendEmail(recipientEmail, subject, message);
}
```

The MailApp Class is limited to quotas based on your Google Account:

- Consumer user (ie, personal Gmail account): 100 recipients/day
- Google Apps (legacy) customer: 100 recipients/day
- GSuite (basic/Gov/Edu/Business): 1500 recipients/day

You can check your email quota within `MailApp`

```
function checkQuota() {
  Logger.log(MailApp.getRemainingDailyQuota());
}
```

Access data from sheet

```
function getSheetData() {

  var sheet = SpreadsheetApp.getActiveSheet();

  var startRow = 2;  // First row of data to process
  var numRows = 100;   // Number of rows to process
  var startCol = 1;  //First column of data to process
  var numCols = 15;    // Number of columns to process
  var dataRange = sheet.getRange(startRow, startCol, numRows, numCols);

  // Fetch values for each row in the Range.
  var data = dataRange.getValues();

  return data;
}
```

You can also modify the above function as follows to get data range dynamic from the content present in sheet:

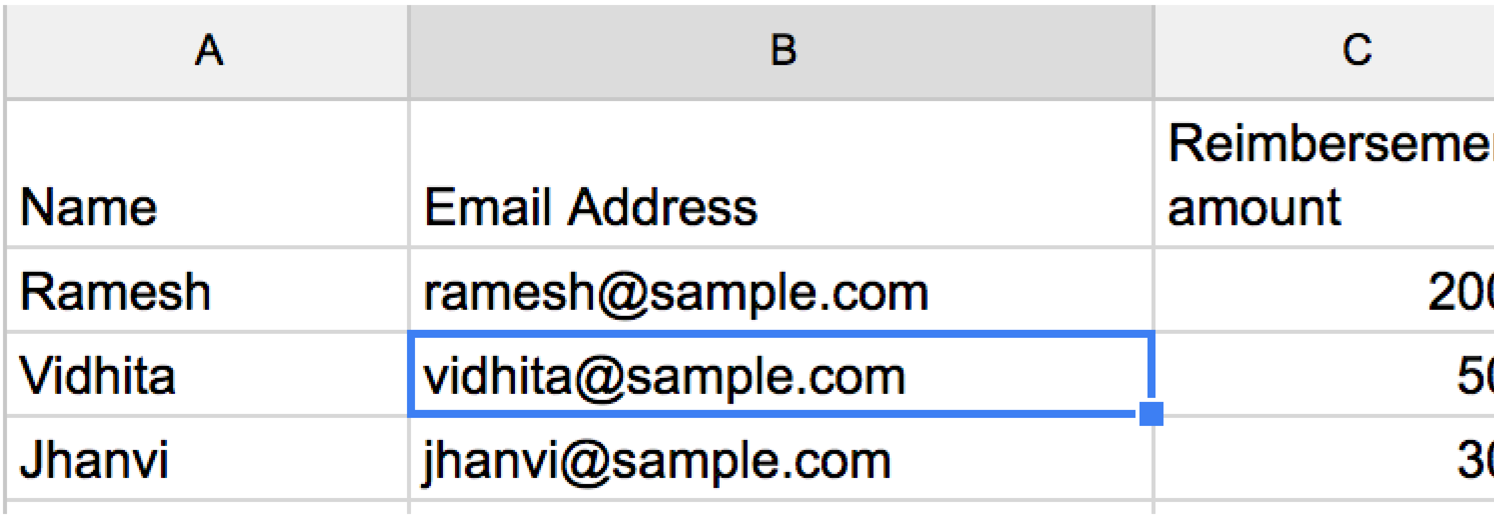
```
function getDataSheet() {

  sheet = SpreadsheetApp.getActiveSheet();

   //Get data range based on content
  var dataRange = sheet.getDataRange();

  // Fetch values for each row in the Range.
  var data = dataRange.getValues();

  return data;
}
```

Use Sheet data to send email

Given - A have sheet of employees who have requested for reimbursement.

Requirement - We should sent out an email to the employee when their reimbursement is processed

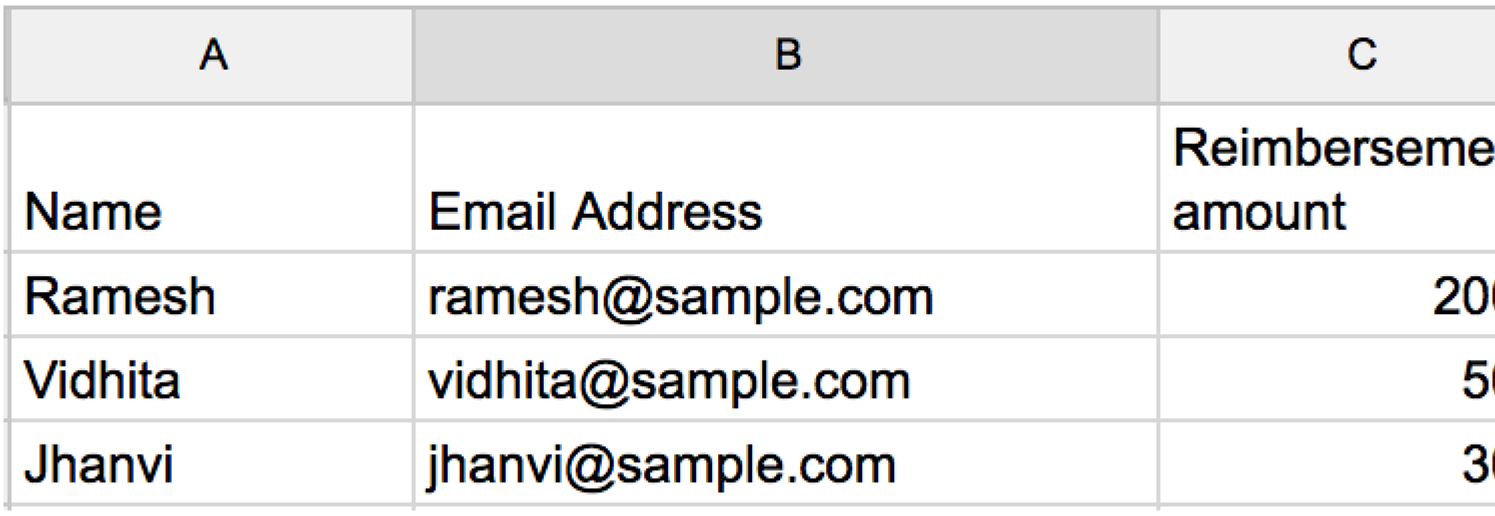
So, the sheet is as follows:



A	B	C
Name	Email Address	Reimberseme amount
Ramesh	ramesh@sample.com	20
Vidhita	vidhita@sample.com	5
Jhanvi	jhanvi@sample.com	3

The function for sending out an email is as follows:

```
function getDataSheet() {

  sheet = SpreadsheetApp.getActiveSheet();
  startRow = 2;  // First row of data to process
  numRows = 100;   // Number of rows to process
  startCol = 1;  //First column of data to process
  numCols = 15;    // Number of columns to process

  var dataRange = sheet.getRange(startRow, startCol, numRows, numCols);

  // Fetch values for each row in the Range.
  var data = dataRange.getValues();

  return data;
}

function getMessage(name, amount) {
  return "Hello " + name + ", Your reimbursement for amount " + amount + " is processed
successfully";
}

function sendEmail() {

  var emailSent = "Yes";
  var reimbursed = "Yes";
  var emailCol = 5;

  var data = getDataSheet();

  for (var i = 0; i < data.length; i++) {

    var row = data[i];

    var isReimbursed = row[3];
    var isEmailSent = row[4];
    var name = row[0];
    var amount = row[2];

    if(isReimbursed == reimbursed && isEmailSent != emailSent) {

      var subject = "Reimbursement details";
      var message = getMessage(name, amount);

      var recipientEmail = row[1];

      MailApp.sendEmail(recipientEmail, subject, message);

      //Sheet range starts from index 1 and data range starts from index 0
      sheet.getRange(1 + i, emailCol).setValue(emailSent);
    }
  }
}
```

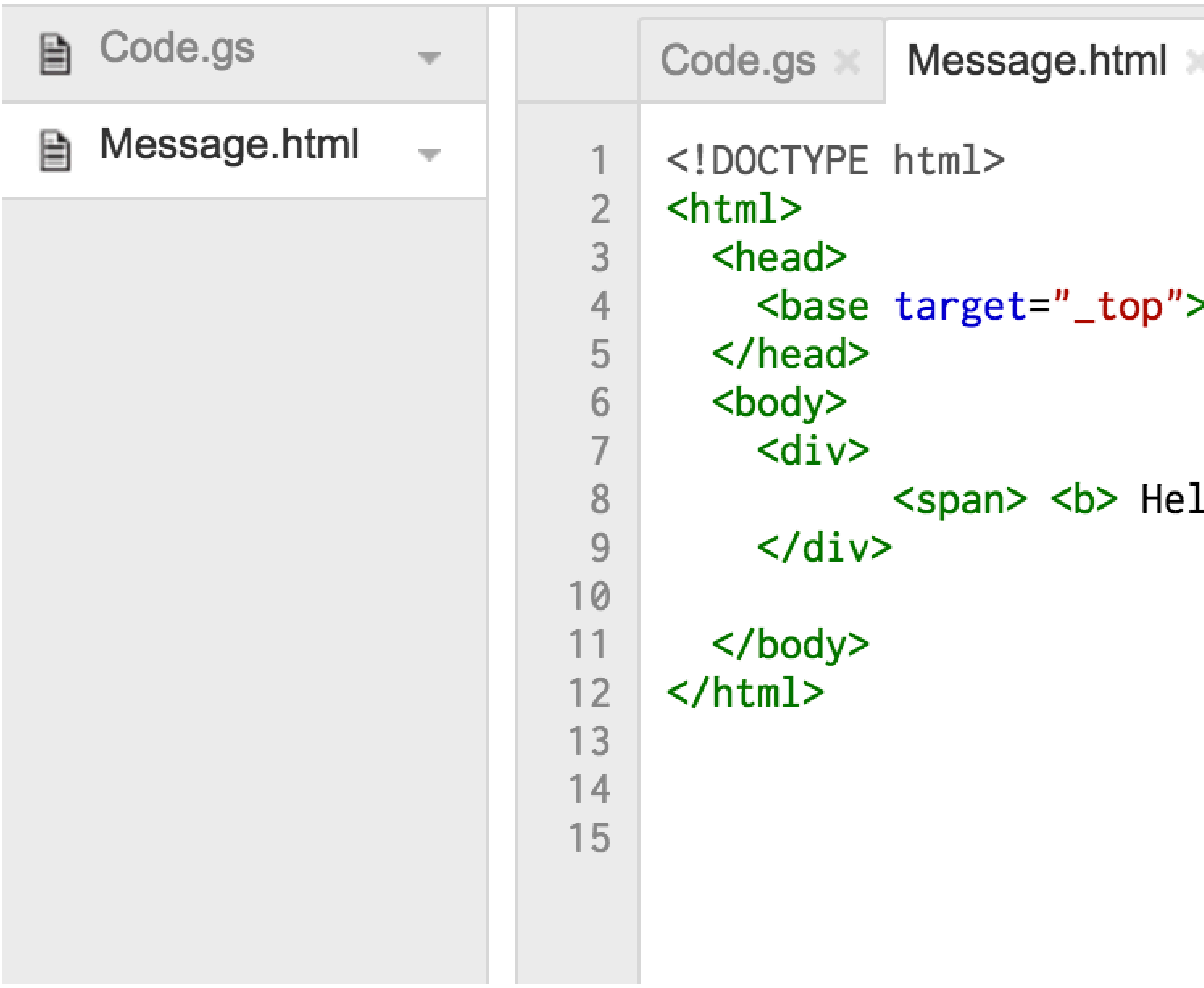


	A	B	C
	Name	Email Address	Reimberseme amount
	Ramesh	ramesh@sample.com	20
	Vidhita	vidhita@sample.com	5
	Jhanvi	jhanvi@sample.com	3

Sending HTML content in mail

In the above example, if we want to send out HTML content as message in the email, then create a HTML file by going to **File -> New -> HTML file**

Now you can see a HTML file besides your gs file as follows :


Code.gs
Message.html

```
<!DOCTYPE html>
<html>
  <head>
    <base target="_top">
  </head>
  <body>
    <div>
          <span> <b> Hel
    </div>

  </body>
</html>
```

Now, update the *getMessage()* method from above example as follows :

```
function getMessage(name, amount) {
  var htmlOutput = HtmlService.createHtmlOutputFromFile('Message'); // Message is the name of
the HTML file

  var message = htmlOutput.getContent()
  message = message.replace("%name", name);
  message = message.replace("%amount", amount);

  return message;
}
```

The call to *MailApp* api needs to be changed as well

```
MailApp.sendEmail(recipientEmail, subject, message, {htmlBody : message});
```

So the whole code will be as follows :

```
function getDataSheet() {

  sheet = SpreadsheetApp.getActiveSheet();

  startRow = 2;  // First row of data to process
  numRows = 100;   // Number of rows to process
  startCol = 1;  //First column of data to process
  numCols = 15;    // Number of columns to process

  var dataRange = sheet.getRange(startRow, startCol, numRows, numCols);

  // Fetch values for each row in the Range.
  var data = dataRange.getValues();

  return data;
}

function getMessage(name, amount) {
  var htmlOutput = HtmlService.createHtmlOutputFromFile('Message');

  var message = htmlOutput.getContent()
  message = message.replace("%name", name);
  message = message.replace("%amount", amount);

  return message;
}

function sendEmail() {

  var emailSent = "Yes";
  var reimbursed = "Yes";
  var emailCol = 5;

  var data = getDataSheet();

  for (var i = 0; i < data.length; i++) {

    var row = data[i];

    var isReimbursed = row[3];
    var isEmailSent = row[4];
    var name = row[0];
    var amount = row[2];

    if(isReimbursed == reimbursed && isEmailSent != emailSent) {

      var subject = "Reimbursement details";
      var message = getMessage(name, amount);

      var recipientEmail = row[1];

      MailApp.sendEmail(recipientEmail, subject, message, {htmlBody : message});

      sheet.getRange(startRow + i, emailCol).setValue(emailSent);
    }
  }
}
```

Read Google sheets MailApp online: https://riptutorial.com/google-apps-script/topic/5298/google-sheets-mailapp

Chapter 12: Google Web App Script To Auto Download From Google Drive

Introduction

This Simple Google App Web Script (Standalone) allows Google Drive to be repeated polled for files to be downloaded to the user's local PC. Shows how to use one app script to provide the function of both the: 1. User interface (a simple one in this example) 2. The file download page. For a fuller explanation of how it works read the Example "How it works".

Remarks

The Web Script must be published in order to work.

Pops ups must be enabled for https://script.google.com

Examples

forms.html

```
<!DOCTYPE html>
<html>
  <head>
    <base target="_top">
    <script>

    setInterval(
    function ()
    {
      document.getElementById('messages').innerHTML = 'Event Timer Fetching';
      google.script.run
        .withSuccessHandler(onSuccess)
        .withFailureHandler(onFailure)
        .fetchFromGoogleDrive();
    }, 60000);

    function callFetchGoogleDrive() {
      document.getElementById('messages').innerHTML = 'Fetching';
      google.script.run
        .withSuccessHandler(onSuccess)
        .withFailureHandler(onFailure)
        .fetchFromGoogleDrive();
    }

    function onSuccess(sHref)
    {
      if(new String(sHref).valueOf() == new String("").valueOf())
      {
        document.getElementById('messages').innerHTML = 'Nothing to download';
      }
      else
      {
        document.getElementById('messages').innerHTML = 'Success';
        document.getElementById('HiddenClick').href = sHref;
        document.getElementById('HiddenClick').click(); // Must enable Pop Ups for
https://script.google.com
      }
    }

    function onFailure(error)
    {
      document.getElementById('messages').innerHTML = error.message;
    }

    </script>
  </head>
  <body>
    <div id="messages">Waiting to DownLoad!</div>
    <div>
      <a id="HiddenClick" href="" target="_blank" onclick="google.script.host.close"
style="visibility: hidden;">Hidden Click</a>
    </div>
    <div>
      <button type="button" onclick='callFetchGoogleDrive();' id="Fetch">Fetch Now!</button>
    </div>
  </body>
</html>
```

code.gs

```
function doGet(e){
   var serveFile = e.parameter.servefile;
   var id = e.parameter.id;

   if(serveFile)
   {
     return downloadFile(id); // and Hyde
   }

  return HtmlService.createHtmlOutputFromFile('form.html'); // Jekyll
}

function fetchFromGoogleDrive() { // Jekyll
  var fileslist = DriveApp.searchFiles("your search criteria goes here + and trashed =
false"); // the 'and trashed = false' prevents the same file being download more than once

  if (fileslist.hasNext()) {
    var afile = fileslist.next();
    var html = ScriptApp.getService().getUrl()+"?servefile=true&id="+afile.getId();
    return html;
  }
  else
  {
    return '';
  }
}

function downloadFile(id){ // and Hyde
  try
  {
    var afile = DriveApp.getFileById(id);

    var aname = afile.getName();
    var acontent = afile.getAs('text/plain').getDataAsString();

    var output = ContentService.createTextOutput();
    output.setMimeType(ContentService.MimeType.CSV);
    output.setContent(acontent);
    output.downloadAsFile(aname);
    afile.setTrashed(true);
    return output;
  }
  catch (e) {
    return ContentService.createTextOutput('Nothing To Download')
  }
}
```

How it works

Google Drive (Standalone) Web App to automatically download (Poll) files from Drive to user's local PC (Download Folder).

DriveApp provides mechanisms for searching and downloading files. However the download mechanism has some serious limitations due to the client/server architecture Google Apps inherited. (No fault of Google)

The server side DriveApp does not provide a direct function to download to the local PC because the server has no concept of where the client is and downloading the file to the server itself would be meaningless.

The server side code needs a mechanism to provide the client side code the file data or a link to file. Both these mechanisms are provided but the data from the former is limited to being used by the client side code directly. The client has no mechanism of saving the data, once obtained, to the local PC. So it can be used to display the data on the web page itself.

The second mechanism allows the url of the script (itself) or the url of the Drive file to be passed back. The Drive file url is not very useful as it cannot be directly used in the client browser to download the file. Placing this url in anchor (and clicking it) only results in a web page that opens but does not actually do anything (except possibly view the file online).

That leaves the script url. However the script url only provides the script and not the file.

In order to initiate a download the file from the Drive service must be returned from the doGet / doPost function of the server side script using ContentService createTextOutput exactly as shown in the Google online guides. However this implies that there can be no other UI element on the web page generated by the results returned by doGet/doPost.

This leaves us with a very unattractive solution. A blank web page with no user UI elements that downloads a page, closes and that requires manually opening when ever another download is required.

Obviously another hosting web page could provide the UI and the link to the Web App Download script to resolve this issue.

This script uses a Dr Jekyll and Mr Hyde approach to resolve this issue.

If the script is opened with no parameters to the GET (doGet) then it defaults to displaying a form. This will be the condition when the published app is first opened by a user. The form provided in this example is extremely simple.

If the script is opened with the parameter servefile=true then the script behaves as a Drive file download.

The client side javascript contains a polling mechanism (event timer setInterval) that periodically calls server side script to check for the availability of another file to download.

When the server side script is executed if it finds any Drive file that matches the search criteria* it returns the url of the script itself appended with the parameters:

?servefile=true&id=the_id_of_the_google_drive_file

(* The search criteria in this simple example is hard coded into the server side script. It could easily be passed from the client to the server if required.)

This information is returned as a string to the client via the recognised withSuccessHandler mechanism.

The client java script then updates the HREF of a hidden anchor with this returned information and then clicks the anchor automatically.

This causes another invocation of the app/script to be launched. When the new invocation of the app launches the doGet will detect the servefile parameter and instead of returning the user interface it will return the file, to the browser. The file returned will be that identified by the provided ID parameter that was previously returned by the search described above.

Given that the file with the provided ID still exists it will be downloaded and the new invocation of the app will close leaving the first invocation to repeat this process.

A button is provided on the simple interface if the user/tester becomes impatient with waiting for the timer but it is not required and can otherwise be removed.

The simple form can of course be extended to provide a richer user interface if required. Such as providing the file search criteria.

Read Google Web App Script To Auto Download From Google Drive online: https://riptutorial.com/google-apps-script/topic/8212/google-web-app-script-to-auto-download-from-google-drive

Chapter 13: Spreadsheet Add Menu

Syntax

1. addMenu(name, subMenus)

Parameters

Name	Description
name	the name of the menu to be created
subMenus	an array of JavaScript maps

Remarks

Usually, you will want to call addMenu from the onOpen function so that the menu is automatically created when the Spreadsheet is loaded.

```
 // The onOpen function is executed automatically every time a Spreadsheet is loaded
 function onOpen() {
   var activeSheet = SpreadsheetApp.getActiveSpreadsheet();
   var menuItems = [];
   // When the user clicks on "addMenuExample" then "Menu 1", the function Myfunction1 is
executed.
   menuItems.push({name: "Menu 1", functionName: "Myfunction1"});
   menuItems.push(null); // adding line separator
   menuItems.push({name: "Menu 2", functionName: "Myfunction2"});

   activeSheet.addMenu("addMenuExample", menuEntries);
 }
```

Examples

Create a new menu

Creates a new menu in the Spreadsheet UI. Each menu entry runs a user-defined function.

```
   var activeSheet = SpreadsheetApp.getActiveSpreadsheet();
   var menuItems = [];
   // When the user clicks on "addMenuExample" then "Menu 1", the function Myfunction1 is
executed.
   menuItems.push({name: "Menu 1", functionName: "Myfunction1"});
   menuItems.push(null); // adding line separator
   menuItems.push({name: "Menu 2", functionName: "Myfunction2"});

   activeSheet.addMenu("addMenuExample", menuEntries);
```

Create Custom Menu

/*

Method: To Create Custom Menu This is First Function to be called when App Loads

---

*/

```
function onOpen() {
  var ui = SpreadsheetApp.getUi();
  // Or DocumentApp or FormApp.
  ui.createMenu('My HR')
      .addItem('Send Form to All', 'sendIDPForm_All')
      .addItem('Trigger IDP System', 'applyCategory')
      .addToUi();
}
```

Read Spreadsheet Add Menu online: https://riptutorial.com/google-apps-script/topic/4253/spreadsheet-add-menu

Chapter 14: Spreadsheet Service

Remarks

The official API reference for the Spreadsheet Service can be found at https://developers.google.com/apps-script/reference/spreadsheet/.

Examples

Sheet

Getting a reference to a named sheet tab

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();//Get active spreadsheet
var sheet_with_name_a = spread_sheet.getSheetByName("sheet_tab_name");
```

Getting active sheet tab

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();
var active_sheet = spread_sheet.getActiveSheet();
```

Insert Column

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();
var active_sheet = spread_sheet.getActiveSheet();
active_sheet.insertColumnAfter(1);  // This inserts a column after the first column position
active_sheet.insertColumnBefore(1);  // This inserts a column in the first column position
active_sheet.insertColumns(1);  // Shifts all columns by one
active_sheet.insertColumns(1, 3);  // Shifts all columns by three
active_sheet.insertColumnsAfter(1);  // This inserts a column in the second column position
active_sheet.insertColumnsBefore(1, 5);  // This inserts five columns before the first column
```

Insert row

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();
var active_sheet = spread_sheet.getActiveSheet();
active_sheet.insertRowAfter(1);  // This inserts a row after the first row position
active_sheet.insertRowBefore(1);  // This inserts a row in the first row position
active_sheet.insertRows(1);  // Shifts all rows by one
active_sheet.insertRows(1, 3);  // Shifts all rows by three
active_sheet.insertRowsAfter(1);  // This inserts a row in the second row position
active_sheet.insertRowsBefore(1, 5);  // This inserts five rows before the first row
```

Cell value

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();
var active_sheet = spread_sheet.getActiveSheet();
var cell = range.getCell(1, 1);
var cell_value = cell.getValue();
```

```
cell.setValue(100);
```

Copy cells

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();
var active_sheet = spread_sheet.getActiveSheet();
var rangeToCopy = active_sheet.getRange(1, 1, sheet.getMaxRows(), 5);
rangeToCopy.copyTo(sheet.getRange(1, 6));
```

Formula

```
var spread_sheet = SpreadsheetApp.getActiveSpreadsheet();
var active_sheet = spread_sheet.getActiveSheet();
var range = active_sheet.getRange("B5");
var formula = range.getFormula()
range.setFormula("=SUM(B3:B4)");
```

Copy a value from one sheet to current sheet

Imagine that we have a separate Google spreadsheet, and we need to get the B2 cell value to cell D5 on your current sheet.

```
function copyValueandPaste()
{
    var source = SpreadsheetApp.openById('spread sheet id is here'); //Separate spreadsheet
book
    var sourcesheet = source.getSheetByName('Sheet1'); //Sheet tab with source data
    var sourceCellValue = sourcesheet.getRange('B2').getValue(); // get B2 cell value

    var thisBook = SpreadsheetApp.getActive(); // Active spreadsheet book
    var thisSheet = thisBook.getSheetByName('Sheet1'); // Target sheet
    thisSheet.getRange('D5').setValue(sourceCellValue); //Set value to target sheet D5 cell
}
```

You can find spreadsheet id from your URL.

Get the last row in a single column

```
function lastRowForColumn(sheet, column){
  // Get the last row with data for the whole sheet.
  var numRows = sheet.getLastRow();

  // Get all data for the given column
  var data = sheet.getRange(1, column, numRows).getValues();

  // Iterate backwards and find first non empty cell
  for(var i = data.length - 1 ; i >= 0 ; i--){
    if (data[i][0] != null && data[i][0] != ""){
      return i + 1;
    }
  }
}
```

Inserting Arrays as Rows

Inserting a row at the bottom of a spreadsheet is easy:

```
var someSheet = SpreadsheetApp.getActiveSpreadsheet().getSheets()[0];
someSheet.appendRow(["Frodo", "Baggins", "Hobbit", "The Shire", 33]);
```

Note this will add the row after the last *non-empty* row.

Inserting a row somewhere in the middle is a bit more work:

```
var someSheet = SpreadsheetApp.getActiveSpreadsheet().getSheets()[0];

var newRowIndex = 2;
var row = ["Gandalf", "?", "Wizard", "?", 2019];
someSheet.insertRowBefore(newRowIndex);
// getRange(row, col, numRows, numCols)
someSheet.getRange(newRowIndex, 1, 1, row.length).setValues([row]); // Note 2D array!
```

A lot of this useless code can be abstracted into a helper function:

```
function insertRowBefore(sheet, rowIndex, rowData) {
  sheet.insertRowBefore(rowIndex);
  sheet.getRange(rowIndex, 1, 1, rowData.length).setValues([rowData]);
}
```

Which reduces our example to just:

```
var someSheet = SpreadsheetApp.getActiveSpreadsheet().getSheets()[0];
insertRowBefore(someSheet, 2, ["Gandalf", "?", "Wizard", "?", 2019]);
```

Read Spreadsheet Service online: https://riptutorial.com/google-apps-script/topic/2688/spreadsheet-service

Chapter 15: SpreadsheetApp Active Sheet

Remarks

Method: getActive()

Return Type: Spreadsheet

Examples

getActive() - Get active spreadsheet

This returns the currently active spreadsheet, or null if there is none.

```
var currentSheet = SpreadsheetApp.getActive();
var url = currentSheet.getUrl();
Logger.log( url );
```

Read SpreadsheetApp Active Sheet online: https://riptutorial.com/google-apps-script/topic/5861/spreadsheetapp-active-sheet

Credits

S. No	Chapters	Contributors
1	Getting started with google-apps-script	Albert Portnoy, Community, Douglas Gaskell, iJay, MShoaib91, Rubén, Saloni Vithalani, Shyam Kansagra, Spencer Easton, sudo bangbang, Supertopoz
2	Apps Script Web Apps	Douglas Gaskell
3	Client calls to Google apps-script	Supertopoz
4	Create a custom function for Google Sheets	Francky_V, Joshua Dawson, Pierre-Marie Richard, Rubén
5	DriveApp	Brian, Kos, nibarius, Sandy Good, Wolfgang
6	DriveApp - getFileById(id)	Sandy Good
7	DriveApp Service	Sandy Good
8	DriveApp Service - Files by type and search string	nibarius, Sandy Good
9	Firebase and AppScript : Introduction	Joseba, Vishal Vishwakarma
10	GmailApp	nibarius
11	Google sheets MailApp	Bhupendra Piprava, Brian, Jordan Rhea, Kos, nibarius, Saloni Vithalani
12	Google Web App Script To Auto Download From Google Drive	Walter
13	Spreadsheet Add Menu	Bishal, iJay, nibarius
14	Spreadsheet Service	cdrini, iJay, nibarius, Sandy Good, sudo bangbang

15	SpreadsheetApp Active Sheet	iJay