



Kostenloses eBook

LERNEN

google-apps-script

Free unaffiliated eBook created from
Stack Overflow contributors.

**#google-
apps-script**

Inhaltsverzeichnis

Über.....	1
Kapitel 1: Erste Schritte mit Google-Apps-Skript.....	2
Bemerkungen.....	2
Examples.....	2
Installation oder Setup.....	2
Arten von Skripten.....	3
Skript ausführen / debuggen.....	4
Hallo Welt.....	4
Ein tieferer Blick auf Google Apps Script.....	4
Kapitel 2: Apps Script Web Apps.....	6
Bemerkungen.....	6
Examples.....	6
Web-App-Formular.....	6
Kapitel 3: Client ruft Google-Apps-Skript auf.....	12
Einführung.....	12
Examples.....	12
Dies ist ein Beispiel für einen clientseitigen Aufruf eines Google-App-Skripts.....	12
Kapitel 4: DriveApp.....	13
Examples.....	13
Erstellen Sie einen neuen Ordner in einem Google Drive-Stammverzeichnis.....	13
Erstellen Sie in Google Drive eine neue Datei mit einem bestimmten Mime-Typ.....	13
Erstellen Sie eine neue Textdatei im Google Drive-Stammverzeichnis.....	13
Erstellen Sie eine neue Datei in Google-Laufwerk aus einem Blob.....	13
Alle Ordner abrufen - Ordner in ein Fortsetzungstoken setzen - dann vom Token abrufen.....	14
Holen Sie sich alle Dateien - fügen Sie sie in ein Fortsetzungstoken ein und rufen Sie sie.....	14
Fügen Sie dem Stammlaufwerk einen Ordner hinzu.....	15
Erstellen Sie eine neue Textdatei und fügen Sie sie dem Stammordner hinzu.....	15
Holen Sie sich alle Dateien in einem Laufwerksordner.....	16
Kapitel 5: DriveApp - getFileById (id).....	18
Bemerkungen.....	18

Examples.....	18
Rufen Sie eine Datei mit der Datei-ID von Google Drive ab.....	18
Kapitel 6: DriveApp Service.....	19
Bemerkungen.....	19
Examples.....	19
Erstellen Sie einen neuen Ordner im Google-Stammverzeichnis.....	19
Erstellen Sie in Google Drive eine neue Datei mit einem bestimmten Mime-Typ.....	19
Erstellen Sie eine neue Textdatei im Google-Stammverzeichnis.....	19
Erstellen Sie aus einem Blob eine neue Datei in Google Drive.....	20
Alle Ordner abrufen - Ordner in ein Fortsetzungstoken setzen - dann vom Token abrufen.....	20
Holen Sie sich alle Dateien - fügen Sie sie in ein Fortsetzungstoken ein und rufen Sie sie.....	21
Kapitel 7: DriveApp Service - Dateien nach Typ und Suchstring.....	22
Parameter.....	22
Examples.....	22
Abrufen von Dateien nach Dateityp mit übereinstimmender Zeichenfolge im Dateinamen.....	22
Kapitel 8: Erstellen Sie eine benutzerdefinierte Funktion für Google Sheets.....	24
Einführung.....	24
Examples.....	24
Standardmäßige Schwerkraftkonstante.....	24
Basisbeispiel.....	25
Kapitel 9: Firebase und AppScript: Einführung.....	26
Einführung.....	26
Examples.....	26
Herstellen einer Verbindung zu einem Firebase-Projekt in GAS und Übertragen von Daten von.....	26
Installieren Sie die Firebase-Ressource im AppScript.....	26
Nehmen wir nun ein Beispiel für das Lesen und Schreiben von Daten aus Firebase.....	28
Wie finde ich die firebaseURL und den geheimen Schlüssel?.....	29
Jetzt haben Sie die FirebaseURL und den geheimen Schlüssel eingefügt. Jetzt sind Sie bereit.....	30
Einige weitere Funktionen zum Implementieren von Lesen und Schreiben.....	30
1. Schreiben Sie einfache Daten, um zu testen, ob die Verbindung funktioniert oder nicht.....	30
2. Alle Daten lesen.....	30
3. Einen bestimmten Datensatz lesen.....	30

4. Um einen bestimmten Datensatz zu aktualisieren.....	31
Kapitel 10: Google MailApp.....	32
Bemerkungen.....	32
Examples.....	32
CSV-Datei an eine E-Mail anhängen.....	32
Kapitel 11: Google Sheets MailApp.....	33
Einführung.....	33
Examples.....	33
Ein einfaches MailApp-Beispiel.....	33
Auf Daten aus dem Blatt zugreifen.....	33
Verwenden Sie Blattdaten, um E-Mails zu senden.....	34
Senden von HTML-Inhalten per E-Mail.....	36
Kapitel 12: Google Web App-Skript für den automatischen Download von Google Drive.....	39
Einführung.....	39
Bemerkungen.....	39
Examples.....	39
forms.html.....	39
code.gs.....	40
Wie es funktioniert.....	41
Kapitel 13: SpreadsheetApp Active Sheet.....	44
Bemerkungen.....	44
Examples.....	44
getActive () - Aktive Kalkulationstabelle abrufen.....	44
Kapitel 14: Tabellenkalkulationsmenü hinzufügen.....	45
Syntax.....	45
Parameter.....	45
Bemerkungen.....	45
Examples.....	45
Erstellen Sie ein neues Menü.....	45
Benutzerdefiniertes Menü erstellen.....	46
Kapitel 15: Tabellenkalkulationsservice.....	47

Bemerkungen.....	47
Examples.....	47
Blatt.....	47
Kopieren Sie einen Wert von einem Blatt in das aktuelle Blatt.....	48
Holen Sie sich die letzte Zeile in einer einzelnen Spalte.....	48
Einfügen von Arrays als Zeilen.....	49
Credits.....	50



You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [google-apps-script](#)

It is an unofficial and free google-apps-script ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official google-apps-script.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit Google-Apps-Skript

Bemerkungen

Die offizielle Übersicht zu Google Apps Script wird von dort unter <http://www.google.com/script/start> veröffentlicht

Google Apps Script ist eine JavaScript-Cloud-Skriptsprache, die einfache Methoden zum Automatisieren von Aufgaben für alle Google-Produkte und Drittanbieter-Services sowie zum Erstellen von Webanwendungen bietet.

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

Apps Script basiert auf [JavaScript 1.6](#) sowie einigen Funktionen von [1.7](#) und [1.8](#). Neben den integrierten und [erweiterten Google-Diensten](#) stehen daher viele grundlegende JavaScript-Funktionen zur Verfügung: Sie können allgemeine Objekte wie [Array](#), [Date](#), [RegExp](#) usw. sowie die globalen Objekte [Math](#) und [Object verwenden](#). Da jedoch Apps Script-Code auf Googles Servern (nicht clientseitig, außer für [HTML-Serviceseiten](#)) ausgeführt wird, sind browserbasierte Funktionen wie DOM-Manipulation oder die [Window-API](#) nicht verfügbar.

Examples

Installation oder Setup

Google Apps Script erfordert keine Einrichtung oder Installation. Die einzige Voraussetzung ist ein Google-Konto. Ein Google Mail-Konto funktioniert ebenso wie ein Google Apps for Work / Education / Government-Konto. Sie können ein neues Google-Konto erstellen, indem Sie auf accounts.google.com gehen

Starten Sie Ihr erstes Skript, indem Sie zu script.google.com gehen. Sie können auf Google Apps Script auch über die `tools -> Script editor...` vieler Google Apps zugreifen, z. B. *Docs*, *Sheets*, *Forms* usw. Google Apps Script kann mit der Funktion `Connect more apps...` auch direkt zu Google Drive hinzugefügt werden.

Offizielle Dokumentation finden Sie unter <http://developers.google.com/apps-script/>.

Damit App-Skripts ausgeführt werden können, müssen sie eine `code.gs`-Datei enthalten. Die Datei `code.gs` muss eine Funktion namens `doGet` (eigenständige Skripts) oder eine `onOpen`-Funktion (Addon-Skripts) enthalten. Die Schnellstartanleitungen in der Dokumentation enthalten Beispiele.

Wenn im App-Skript ein API aktiviert ist, muss es auch in der Entwickler-Konsole aktiviert sein. Die Entwicklerkonsole enthält jedoch APIs, die aktiviert werden können, aber nicht in der App-Skript-Oberfläche angezeigt werden. Zum Beispiel muss Marketplace SDK in der Entwicklerkonsole

aktiviert sein, bevor die App im Google Play Store oder in einer domänenweiten Bereitstellung von G Suite veröffentlicht werden kann.

Für Google Apps für Bildung / Arbeit / Verwaltung gibt es Einstellungen in der Domänenadministrationskonsole, die angepasst werden können, um die Ausführung von App-Skripts zuzulassen oder zu untersagen.

Arten von Skripten

Es gibt drei Arten von Google App-Skripts.

- Eigenständige
- An Google Apps gebunden
- Web-Apps

Eigenständiges Skript

Standalone-Skripts sind nicht an Google-Apps, z. B. *Docs, Sheets oder Forms usw.*, gebunden. Standalone-Skripts können entweder über *script.google.com* oder durch Verbinden von Google-App-Skript mit Google-Laufwerk erstellt werden. Eigenständiges Skript kann zum unabhängigen Programmieren von Google-Apps verwendet werden, kann als Webanwendung verwendet werden oder kann so eingerichtet werden, dass es automatisch von einem installierbaren Auslöser ausgeführt wird. Weitere Informationen finden Sie in der [Dokumentation](#) zum eigenständigen Skript.

An Google Apps gebunden

An Google Apps gebundenes Skript, auch als Container-gebundenes Skript bekannt. Im Gegensatz zu Standalone-Skripts sind sie an Google-Apps gebunden, z. B. an *Google Text tools> Script editor Tabellen oder Google Sheets usw.* Container-gebundenes Skript kann durch Auswahl von tools> Script editor aus Google App erstellt werden. Einige [Funktionen](#) wie Dialoge, Eingabeaufforderungen, Menüs und Seitenleisten werden nur von Container-gebundenen Skripts bereitgestellt. Außerdem werden Container-gebundene Skripts zum Erstellen von [Google Add-Ons verwendet](#). In der [Dokumentation finden Sie](#) Container-gebundene Skripts.

Web-Apps

Google App Script kann als Web-App verwendet werden, da diese über den Browser aufgerufen werden können. Web App kann eine Benutzeroberfläche im Browser bereitstellen und Google Apps (z. B. *Dokumente, Tabellen usw.*) verwenden. Sowohl eigenständige Skripts als auch an Google Apps gebundene Skripts können in Web-Apps umgewandelt werden. Damit ein Skript als Web-App funktionieren kann, muss das Skript zwei Anforderungen erfüllen:

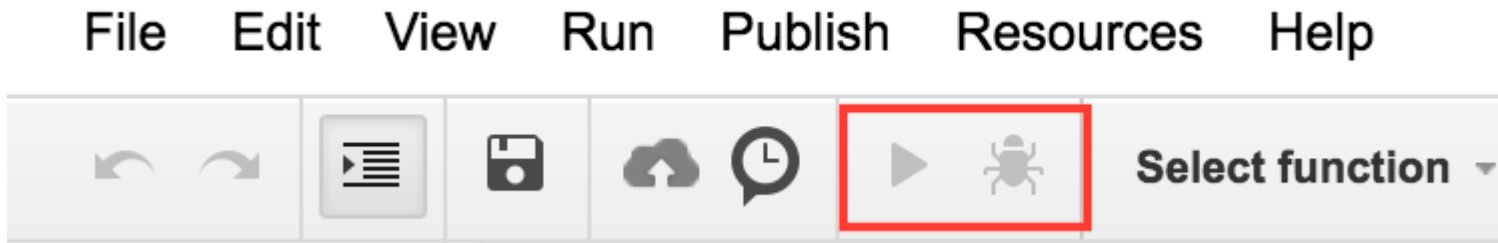
- schließen Sie eine `doGet()` oder eine `doPost()` Funktion ein.
- Die Funktion gibt ein HTML-Service-HtmlOutput-Objekt oder ein Content-Service-TextOutput-Objekt zurück.

Inshort-, `doGet()` und `doPost()` Funktionen funktionieren wie `http-doPost()` und Post-Request-Handler.

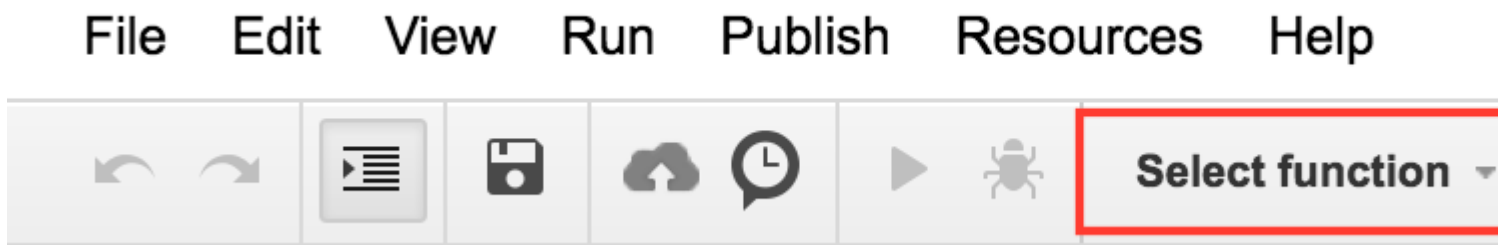
Weitere Informationen zu Web-Apps finden Sie in der offiziellen [Dokumentation](#) .

Skript ausführen / debuggen

Versuchen Sie, Ihren Code wie folgt aus der Symbolleiste auszuführen:



Wenn Sie in Ihrem Code mehr als eine Funktion haben, sollten Sie vor der Ausführung die Funktion erwähnen, mit der Sie arbeiten möchten. Zum Beispiel :



Alternativ können Sie die **Tastenkombination Strg + R** drücken, um den Code auszuführen. Es wird den Code zuerst speichern, falls nicht gespeichert, und dann ausführen. Damit dies funktioniert, müssen Sie die Funktion ausgewählt haben, wie in der Abbildung oben dargestellt.

Wenn Ihr Skript von einigen externen Aktivitäten aufgerufen wird, können Sie weiterhin Protokolle anzeigen, indem Sie auf Ansicht-> Protokolle klicken, wenn Sie nach der Ausführung des Codes etwas protokollieren.

Hallo Welt

Wir werden Hallo als Meldungsfeld sagen.

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

Um das Skript auszuführen, klicken Sie entweder auf ► oder wählen Sie den Menüpunkt **Ausführen -> HalloWelt**

Ein tieferer Blick auf Google Apps Script

Google Apps Script ist eine JavaScript-basierte Plattform, die hauptsächlich zur Automatisierung und Erweiterung von Google Apps verwendet wird. Apps Script wird ausschließlich in der

Infrastruktur von Google ausgeführt und erfordert keine Server-Bereitstellung oder -Konfiguration. Eine Online-IDE dient als Schnittstelle zur gesamten Plattform, die alle für Apps Script verfügbaren Dienste miteinander verbindet. Die Benutzerauthentifizierung wird über OAuth2 in die Plattform eingebettet und erfordert keinen Code oder keine Einrichtung durch den Skriptautor.

Apps Script wird serverseitig ausgeführt, kann jedoch über Benutzeroberflächen verfügen, die mit HTML, CSS, JavaScript oder anderen vom Browser unterstützten Technologien erstellt wurden. Im Gegensatz zu Node.js, das ereignisgesteuert ist, werden App-Skripts in einem Threadmodell ausgeführt. Alle Aufrufe eines Skripts generieren eine eindeutige Instanz dieses Skripts, die isoliert von allen anderen Instanzen ausgeführt wird. Wenn die Ausführung eines Skripts abgeschlossen ist, wird es zerstört.

Funktionen in Apps Script blockieren, sodass Rückruf- und asynchrone Programmiermuster nicht benötigt werden. Durch das Sperren wird verhindert, dass kritische Codeabschnitte, z. B. Datei-E/A, gleichzeitig von verschiedenen Instanzen ausgeführt werden.

In der Praxis werden Apps-Skripte einfach geschrieben. Nachfolgend finden Sie ein einfaches Skript, das eine neue Tabelle aus einer Vorlagen-Tabelle erstellt.

```
// Create a new spreadsheet from a template
function createSpreadsheet() {
  var templateFileId = '1Azcz9GwCeHjG19TXf4aUh6g20Eqmgsd1UMSdNVjzIZPk';
  var sheetName = 'Account Log for:' + new Date();
  SpreadsheetApp.openById(templateFileId).copy(sheetName);
}
```

Erste Schritte mit Google-Apps-Skript online lesen: <https://riptutorial.com/de/google-apps-script/topic/1154/erste-schritte-mit-google-apps-skript>

Kapitel 2: Apps Script Web Apps

Bemerkungen

Dies ist ein Beispiel für eine Formular-Web-App. Das clientseitige Bit zeigt einige grundlegende UX-Designs, z. B. eine deaktivierte Schaltfläche zum Senden, wenn das Formular gesendet wird, oder eine Fehlermeldung, wenn es fehlschlägt

Das Apps Script-Bit ist sehr einfach. Es enthält nur den Code, der zum Bereitstellen der HTML-Datei und zum Überprüfen des Felds erforderlich ist.

Hier ist ein Link zu dieser Beispielanwendung in Aktion: [Beispiel-Anwendungsskriptformular](#)

Hinweis: Sie müssen bei einem Google-Konto angemeldet sein.

Die Dateistruktur des Apps-Skripts lautet wie folgt:

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

Examples

Web-App-Formular

Apps-Skript:

```
//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>
    </div>
  </body>
</html>

```

```

        <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>
```

Apps Script Web Apps online lesen: <https://riptutorial.com/de/google-apps-script/topic/4874/apps-script-web-apps>

Kapitel 3: Client ruft Google-Apps-Skript auf

Einführung

Google appsript funktioniert gut als eigenständige Plattform und im Addon-Format für Google-Dokumente, -Bögen und -Formulare. Es gibt jedoch Situationen, in denen ein Client-Browser möglicherweise eine Google-App aufrufen muss, um eine Aktion auszuführen.

Daher hat Google clientseitige Anforderungen an Google-Apps-Skripts eingeführt. Um dieses Problem zu lösen, hat Google die [clientseitigen Bibliotheken eingeführt](#)

Examples

Dies ist ein Beispiel für einen clientseitigen Aufruf eines Google-App-Skripts

```
<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>
```

Client ruft Google-Apps-Skript auf online lesen: <https://riptutorial.com/de/google-apps-script/topic/8875/client-ruft-google-apps-skript-auf>

Kapitel 4: DriveApp

Examples

Erstellen Sie einen neuen Ordner in einem Google Drive-Stammverzeichnis

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

Verwenden Sie Funktion `createNewFolderInGoogleDrive` Ordner erstellen `Test folder` in einem Google Drive root:

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

`newFolder` hat **Klassen - Ordner - Typ**:

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

Erstellen Sie in Google Drive eine neue Datei mit einem bestimmten Mime-Typ

```
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  
};
```

Erstellen Sie eine neue Textdatei im Google Drive-Stammverzeichnis

```
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  
};
```

Erstellen Sie eine neue Datei in Google-Laufwerk aus einem 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
};

```

Alle Ordner abrufen - Ordner in ein Fortsetzungstoken setzen - dann vom Token abrufen

```

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
};

```

Holen Sie sich alle Dateien - fügen Sie sie in ein Fortsetzungstoken ein und rufen Sie sie ab

```

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);
};

```

Fügen Sie dem Stammlaufwerk einen Ordner hinzu

```

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
};

```

Erstellen Sie eine neue Textdatei und fügen Sie sie dem Stammordner hinzu

```

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
};

```

Holen Sie sich alle Dateien in einem Laufwerksordner

```

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++;
}
}
```

DriveApp online lesen: <https://riptutorial.com/de/google-apps-script/topic/5363/driveapp>

Kapitel 5: DriveApp - getFileById (id)

Bemerkungen

Es ist auch möglich, eine Datei über die URL der Datei abzurufen. Die ID einer Datei befindet sich in der URL. Die Verwendung der ID anstelle der gesamten URL bedeutet, dass der Parameter kürzer ist. Das Speichern der URL anstelle der ID nimmt mehr Speicherplatz in Anspruch.

Examples

Rufen Sie eine Datei mit der Datei-ID von Google Drive ab

```
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
};
```

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

Kapitel 6: DriveApp Service

Bemerkungen

Google MIME-Typen können nicht für den dritten Parameter von MIME-Typen verwendet werden. Die Verwendung eines Google Mime-Typs führt zu einem Fehler, der Folgendes angibt:

"DriveApp.createFile ()" kann nicht zum Erstellen von Google MIME-Typen verwendet werden. Bitte verwenden Sie den Advanced Drive Service

MimeType.GOOGLE_APPS_SCRIPT

MimeType.GOOGLE_DOCS

MimeType.GOOGLE_DRAWINGS

MimeType.GOOGLE_FORMS

MimeType.GOOGLE_SHEETS

MimeType.GOOGLE_SLIDES

Examples

Erstellen Sie einen neuen Ordner im Google-Stammverzeichnis

```
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
};
```

Erstellen Sie in Google Drive eine neue Datei mit einem bestimmten Mime-Typ

```
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
};
```

Erstellen Sie eine neue Textdatei im Google-Stammverzeichnis


```
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
};
```

Erstellen Sie aus einem Blob eine neue Datei in Google Drive

```
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
};
```

Alle Ordner abrufen - Ordner in ein Fortsetzungstoken setzen - dann vom Token abrufen

```
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
};

```

Holen Sie sich alle Dateien - fügen Sie sie in ein Fortsetzungstoken ein und rufen Sie sie ab

```

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);
};

```

DriveApp Service online lesen: <https://riptutorial.com/de/google-apps-script/topic/6395/driveapp-service>

Kapitel 7: DriveApp Service - Dateien nach Typ und Suchstring

Parameter

Parametername	Verwenden für
Suchbegriff	die Zeichenfolge, die im Dateinamen gefunden werden soll

Examples

Abrufen von Dateien nach Dateityp mit übereinstimmender Zeichenfolge im Dateinamen

Rufen Sie alle Google Forms mit dem Wort "Unbenannt" im Dateinamen ab.

```
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;
};

```

DriveApp Service - Dateien nach Typ und Suchstring online lesen:

<https://riptutorial.com/de/google-apps-script/topic/4049/driveapp-service---dateien-nach-typ-und-suchstring>

Kapitel 8: Erstellen Sie eine benutzerdefinierte Funktion für Google Sheets

Einführung

Eine benutzerdefinierte Funktion in Google-Dokumenten ist an ein bestimmtes Dokument gebunden (und kann daher nur in diesem Dokument verwendet werden).

Es muss daher mit der Scrip-Bearbeitung dieses Dokuments erstellt werden (Extras -> Skript-Editor). Nach dem Speichern kann es wie jede andere reguläre Tabellenformel verwendet werden.

Examples

Standardmäßige Schwerkraftkonstante

Diese Funktion gibt die Standardschwerkraftkonstante in den angegebenen Beschleunigungseinheiten zurück (1 für cm / s², 2 für ft / s², 3 für 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;
}
```

Um die Funktion verwenden zu können, muss sie mit dem Skript-Editor (Extras -> Skript-Editor ...) an eine Tabelle gebunden werden. Nachdem die Funktion hinzugefügt wurde, kann sie wie jede andere Google-Tabellenfunktion verwendet werden, indem die Funktion in der Formel einer Zelle

aufgerufen wird.

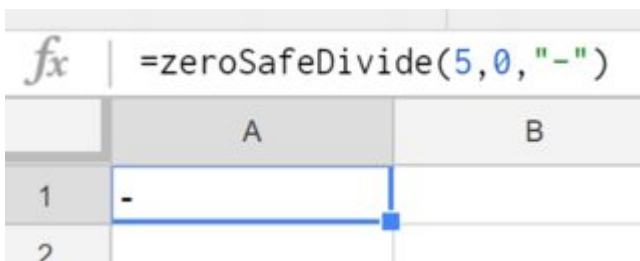
Beachten Sie, wie die Funktion bei der automatischen Eingabe in eine Formel angezeigt wird. Dies ist auf den mehrzeiligen Kommentar über der Funktionsdeklaration zurückzuführen, mit dem beschrieben wird, was die Funktion ähnlich wie JSDoc und Javadoc macht. Damit die Formel in Autocomplete angezeigt wird, muss das `@customfunction`-Tag im Kommentar angegeben werden.

Basisbeispiel

Um unschöne `#DIV/0` Fehler in einer Kalkulationstabelle zu vermeiden, kann eine benutzerdefinierte Funktion verwendet werden.

```
/**
 * 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;
}
```

Um die Funktion verwenden zu können, muss sie mit dem Skript-Editor (**Extras -> Skript-Editor ...**) an eine Tabelle gebunden werden. Nachdem die Funktion hinzugefügt wurde, kann sie wie jede andere Google-Tabellenfunktion verwendet werden, indem die Funktion in der Formel einer Zelle aufgerufen wird.



The screenshot shows a Google Sheet interface. At the top, there is a formula bar with the text `=zeroSafeDivide(5,0,"-")`. Below the formula bar, a table is visible with two columns labeled 'A' and 'B', and two rows labeled '1' and '2'. The cell in row 1, column A contains a minus sign '-'. A blue selection box is around the cell in row 1, column B.

	A	B
1	-	
2		

Beachten Sie, wie die Funktion bei der automatischen Eingabe in eine Formel angezeigt wird. Dies ist auf den mehrzeiligen Kommentar über der Funktionsdeklaration zurückzuführen, mit dem beschrieben wird, was die Funktion ähnlich wie JSDoc und Javadoc macht. Damit die Formel in Autocomplete `@customfunction` muss das `@customfunction` Tag im Kommentar angegeben werden.

Erstellen Sie eine benutzerdefinierte Funktion für Google Sheets online lesen:

<https://riptutorial.com/de/google-apps-script/topic/5572/erstellen-sie-eine-benutzerdefinierte-funktion-fur-google-sheets>

Kapitel 9: Firebase und AppScript: Einführung

Einführung

Integrieren Sie Firebase mit Google AppScript, um Daten in der Firebase-Datenbank zu lesen und zu schreiben.

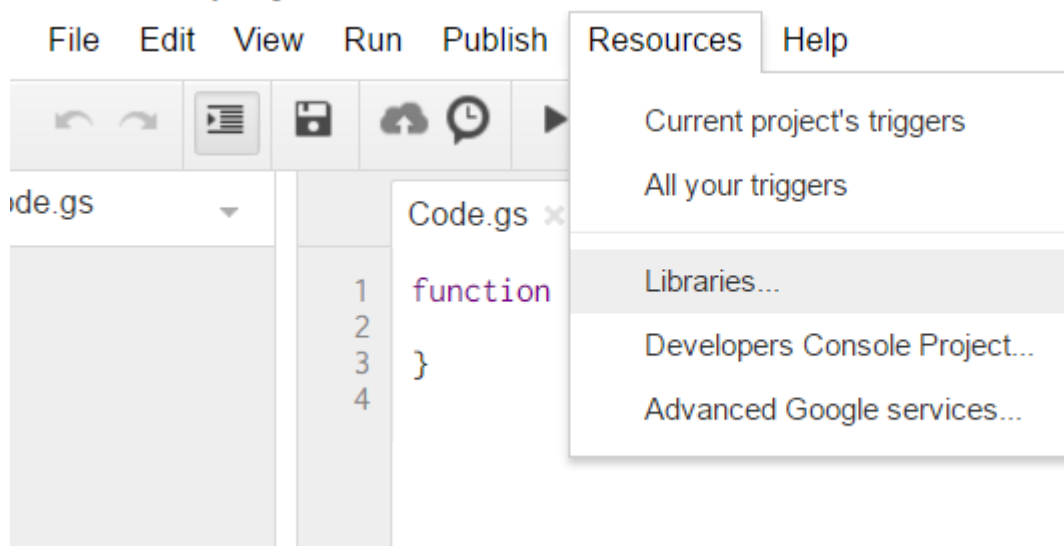
Firebase ist ein NoSQL-Datenbanksystem von Google, das Echtzeitdatenbanken verwendet, um Anwendungen auf mobilen, Desktop- und Tablet-Geräten zu erstellen und zu hosten. NoSQL-Datenbanken verwenden die JSON-Objekte, um die Daten in strukturiertem Format zu speichern.

Examples

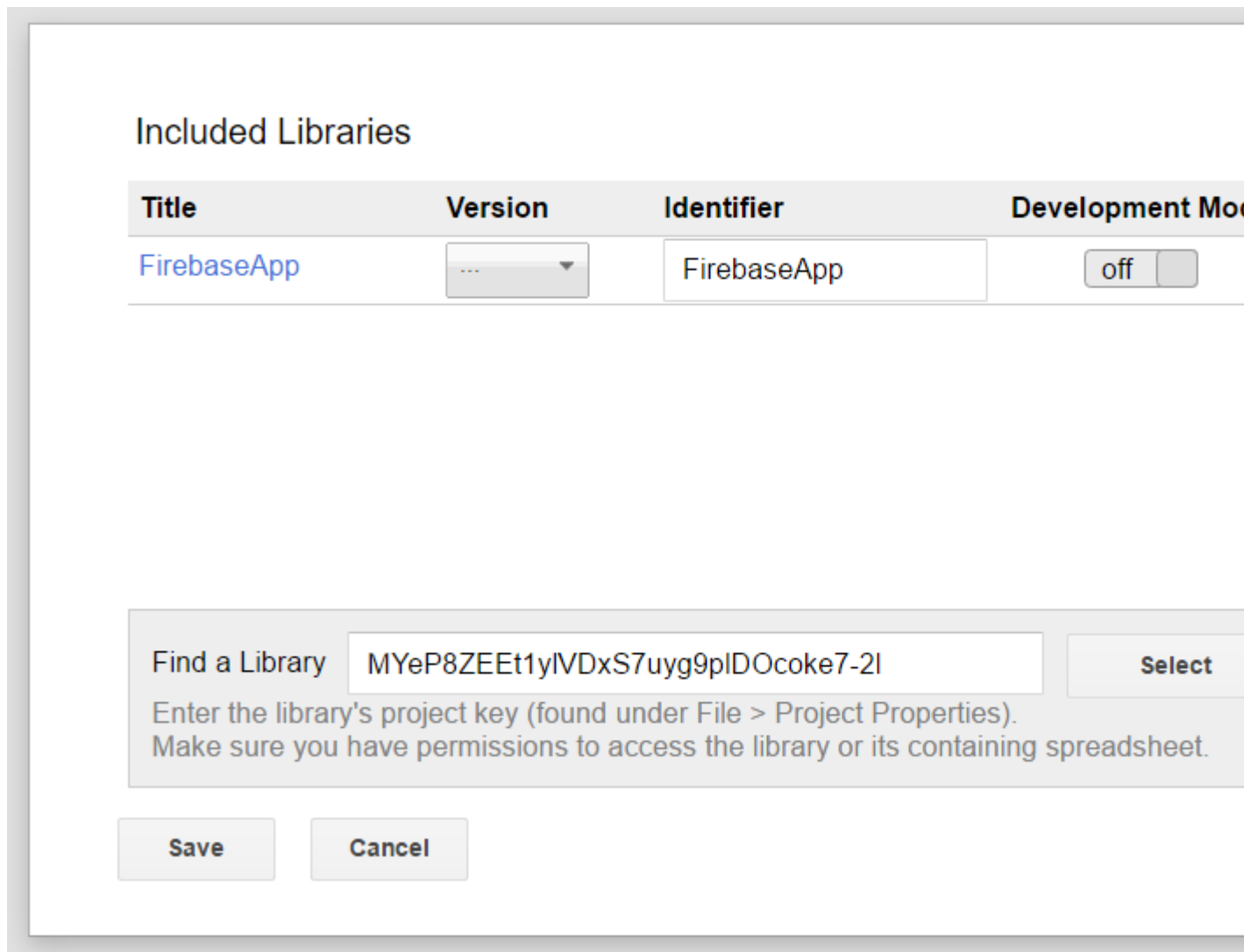
Herstellen einer Verbindung zu einem Firebase-Projekt in GAS und Übertragen von Daten von Google Spreadsheet zu Firebase

Installieren Sie die Firebase-Ressource im AppScript

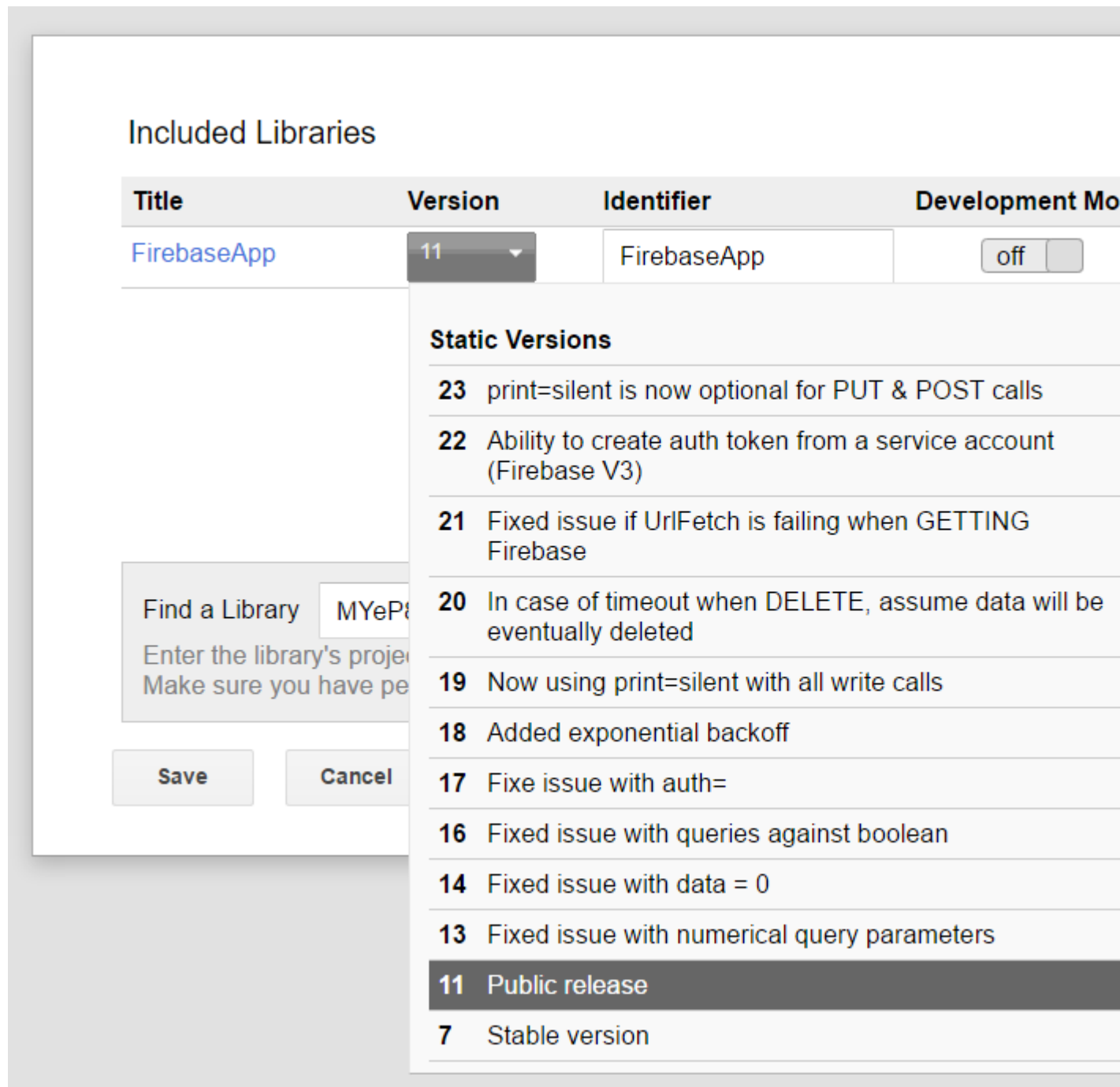
- Klicken Sie dazu auf Ressourcen und dann auf Bibliotheken.
- Firebase verfügt über einen eindeutigen Projektbibliotheksschlüssel, der im AppScript installiert werden muss.



- Klicken Sie auf Bibliotheken. Das folgende Popup-Fenster wird angezeigt. Geben Sie den folgenden Projektschlüssel in das Textfeld ein. **MYeP8ZEEt1yIVDxS7uyg9plDOcoke7-2l** Dies ist der Projektbibliotheksschlüssel für Firebase.



- Wählen Sie jetzt in der Version die stabile öffentliche Release-Version.



- Klicken Sie auf Speichern. Nun ist Firebase erfolgreich in Ihrem AppScript installiert, damit Sie arbeiten können.

Nehmen wir nun ein Beispiel für das Lesen und Schreiben von Daten aus Firebase.

- Jetzt nehmen wir eine in Google Sheets entworfene Probentabelle.

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

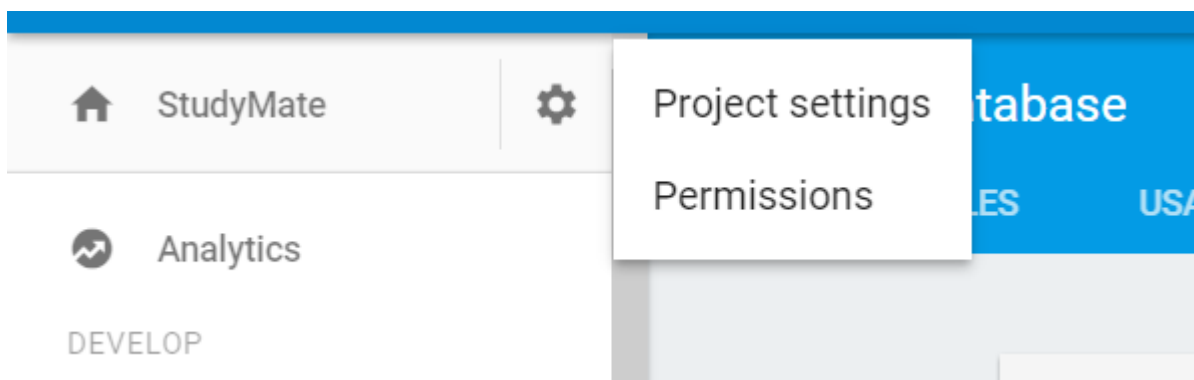
- Erstellen Sie nun die Datenbank in Firebase mithilfe dieser Tabelle in den Arbeitsblättern. Fügen Sie im AppScript den folgenden Code hinzu.

```
function writeToFirebase() {
  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);
}
```

Ersetzen Sie die ID der Kalkulationstabelle, die firebaseURL und den geheimen Schlüssel.

Wie finde ich die firebaseURL und den geheimen Schlüssel?

- Gehen Sie zu Ihrem Firebase Dashboard und klicken Sie oben links auf Einstellungen. Klicken Sie auf Projekteinstellungen.



- Gehen Sie zum Abschnitt Service Accounts, um die Datenbank-URL zu finden. Dies dient als FirebaseURL.
- Klicken Sie nun auf die Registerkarte Database Secrets und Sie können den geheimen Schlüssel finden.

Jetzt haben Sie die FirebaseURL und den geheimen Schlüssel eingefügt. Jetzt sind Sie bereit zu gehen. Klicken Sie in der AppScript-Engine auf Ausführungscodes.

- Sie werden aufgefordert, die Berechtigungen zum ersten Mal zu überprüfen, wenn Sie sie ausführen.
- Klicken Sie auf Berechtigungen überprüfen und zulassen.
- Jetzt führen Sie Ihre Funktion aus und sehen die in Firebase Database erstellte Tabelle.

Um die Datenbank anzuzeigen, gehen Sie zum Firebase-Dashboard. Klicken Sie auf die Datenbank, um die Datenbank anzuzeigen.

Einige weitere Funktionen zum Implementieren von Lesen und Schreiben.

1. Schreiben Sie einfache Daten, um zu testen, ob die Verbindung funktioniert oder nicht.

```
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. Alle Daten lesen

```
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);
  }
}
```

Die gelesenen Daten werden in den Protokollen angezeigt. Um die Protokolle zu prüfen, klicken Sie auf Ansicht → Protokolle oder verwenden Sie einfach Strg + Eingabetaste.

3. Einen bestimmten Datensatz lesen

```
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);
}
```

Die gelesenen Daten werden in den Protokollen angezeigt. Um die Protokolle zu prüfen, klicken Sie auf Ansicht → Protokolle oder verwenden Sie einfach Strg + Eingabetaste.

4. Um einen bestimmten Datensatz zu aktualisieren.

```
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");
}
```

Firestore und AppScript: Einführung online lesen: <https://riptutorial.com/de/google-apps-script/topic/9417/firebase-und-appscript-einfuehrung>

Kapitel 10: Google MailApp

Bemerkungen

Weitere Informationen zu den verfügbaren Methoden finden Sie auch in der offiziellen [API-Referenz](#) für GmailApp.

Examples

CSV-Datei an eine E-Mail anhängen

Angenommen, wir verfügen über ein System, das täglich Berichte in Form von angehängten CSV-Dateien per E-Mail sendet und auf diese zugreifen möchte.

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

Google MailApp online lesen: <https://riptutorial.com/de/google-apps-script/topic/5899/google-mailapp>

Kapitel 11: Google Sheets MailApp

Einführung

Mit diesem Dienst können Benutzer E-Mails mit vollständiger Kontrolle über den Inhalt der E-Mail senden. Im Gegensatz zu GmailApp dient MailApp ausschließlich zum Versenden von E-Mails. MailApp kann nicht auf den Google Mail-Posteingang eines Benutzers zugreifen.

Mit Google MailApp geschriebene Änderungen an Skripten lösen bei einem Benutzer mit größerer Wahrscheinlichkeit eine erneute Autorisierung aus als bei MailApp-Skripten.

Examples

Ein einfaches MailApp-Beispiel

MailApp ist die API von Google App Script, die zum Senden von E-Mails verwendet werden kann

```
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);  
}
```

Die MailApp-Klasse ist auf [Kontingente](#) beschränkt, die auf Ihrem Google-Konto basieren:

- Verbraucher (z. B. persönliches Google Mail-Konto): 100 Empfänger / Tag
- Kunde von Google Apps (Legacy): 100 Empfänger / Tag
- GSuite (basic / Gov / Edu / Business): 1500 Empfänger / Tag

Sie können Ihr E-Mail-Kontingent in `MailApp`

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

Auf Daten aus dem Blatt zugreifen

```
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;
}

```

Sie können die obige Funktion auch wie folgt ändern, um den dynamischen Datenbereich aus dem Inhalt der Tabelle zu erhalten:

```

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;
}

```

Verwenden Sie Blattdaten, um E-Mails zu senden

Gegeben - A haben ein Blatt von Mitarbeitern, die eine Erstattung beantragt haben.

Anforderung - Wir sollten eine E-Mail an den Mitarbeiter senden, wenn die Erstattung bearbeitet wird

Das Blatt ist also wie folgt:

A	B	C
Name	Email Address	Reimberseme amount
Ramesh	ramesh@sample.com	200
Vidhita	vidhita@sample.com	50
Jhanvi	jhanvi@sample.com	30

Die Funktion zum Versenden einer E-Mail lautet wie folgt:

```

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

Senden von HTML-Inhalten per E-Mail

Wenn Sie im obigen Beispiel HTML-Inhalt als Nachricht in der E-Mail versenden möchten, erstellen Sie eine HTML-Datei, indem Sie auf **Datei -> Neu -> HTML-Datei gehen**

Jetzt können Sie eine HTML-Datei neben Ihrer GS-Datei wie folgt anzeigen:

```
Code.gs x Message.html x
Code.gs
Message.html
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <base target="_top">
5   </head>
6   <body>
7     <div>
8       <span> <b> Hel
9     </div>
10
11   </body>
12 </html>
13
14
15
```

Aktualisieren Sie nun die *getMessage ()* -Methode aus dem obigen Beispiel wie folgt:

```
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;
}
```

Der Aufruf an *MailApp* api muss ebenfalls geändert werden

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

Der ganze Code wird also wie folgt aussehen:

```

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);
        }
    }
}

```

Google Sheets MailApp online lesen: <https://riptutorial.com/de/google-apps-script/topic/5298/google-sheets-mailapp>

Kapitel 12: Google Web App-Skript für den automatischen Download von Google Drive

Einführung

Mit diesem einfachen Google App-Web Skript (Standalone) kann Google Drive wiederholt aufgefordert werden, Dateien auf den lokalen PC des Benutzers herunterzuladen. Veranschaulicht die Verwendung eines App-Skripts zum Bereitstellen der Funktionen der beiden: 1. Benutzeroberfläche (in diesem Beispiel eine einfache) 2. Die Seite zum Herunterladen von Dateien. Für eine ausführlichere Erklärung der Funktionsweise lesen Sie das Beispiel "Funktionsweise".

Bemerkungen

Das Web Script muss veröffentlicht werden, damit es funktioniert.

Popups müssen für <https://script.google.com> aktiviert sein

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')
  }
}

```

Wie es funktioniert

Google Drive (Standalone) Web App zum automatischen Herunterladen (Poll) von Dateien auf den lokalen PC des Benutzers (Download Folder).

DriveApp bietet Mechanismen zum Suchen und Herunterladen von Dateien. Der Downloadmechanismus hat jedoch einige gravierende Einschränkungen aufgrund der von Google Apps übernommenen Client / Server-Architektur. (Keine Schuld von Google)

Die serverseitige DriveApp bietet keine direkte Funktion zum Herunterladen auf den lokalen PC, da der Server keine Vorstellung davon hat, wo sich der Client befindet. Das Herunterladen der Datei auf den Server selbst wäre bedeutungslos.

Der serverseitige Code benötigt einen Mechanismus, um dem Client-seitigen Code die Dateidaten oder einen Link zu einer Datei bereitzustellen. Beide Mechanismen sind vorgesehen, aber die Daten aus dem ersteren sind darauf beschränkt, vom clientseitigen Code direkt verwendet zu werden. Der Client hat keinen Mechanismus zum Speichern der Daten, die er einmal erhalten hat, auf dem lokalen PC. So können die Daten auf der Webseite selbst angezeigt werden.

Der zweite Mechanismus ermöglicht die Rückgabe der URL des Skripts (selbst) oder der URL der Drive-Datei. Die Laufwerksdatei-URL ist nicht sehr nützlich, da sie nicht direkt im Client-Browser zum Herunterladen der Datei verwendet werden kann. Wenn Sie diese URL in den Anker setzen (und darauf klicken), wird nur eine Webseite angezeigt, die sich öffnet, aber tatsächlich nichts tut (außer möglicherweise die Datei online anzeigen).

Das verlässt die Skript-URL. Die Skript-URL enthält jedoch nur das Skript und nicht die Datei.

Um einen Download zu initiieren, muss die Datei vom Drive-Dienst von der doGet / doPost-Funktion des serverseitigen Skripts mithilfe von ContentService createTextOutput genau wie in den Online-Handbüchern von Google angezeigt werden. Dies bedeutet jedoch, dass sich auf der Webseite kein anderes Oberflächenelement befinden kann, das durch die von doGet / doPost

zurückgegebenen Ergebnisse generiert wird.

Dies lässt uns eine sehr unattraktive Lösung. Eine leere Webseite ohne Benutzeroberflächenelemente, die eine Seite herunterladen, wird geschlossen und muss manuell geöffnet werden, wenn ein weiterer Download erforderlich ist.

Offensichtlich könnte eine andere Hosting-Webseite die Benutzeroberfläche und den Link zum Skript für den Web App-Download bereitstellen, um dieses Problem zu beheben.

Dieses Skript verwendet einen Ansatz von Dr. Jekyll und Mr. Hyde, um dieses Problem zu lösen.

Wenn das Skript ohne Parameter für GET (doGet) geöffnet wird, zeigt es standardmäßig ein Formular an. Dies ist die Bedingung, wenn die veröffentlichte App zum ersten Mal von einem Benutzer geöffnet wird. Das in diesem Beispiel bereitgestellte Formular ist extrem einfach.

Wenn das Skript mit dem Parameter `servefile = true` geöffnet wird, verhält sich das Skript wie ein Download der Laufwerksdatei.

Das clientseitige Javascript enthält einen Abfragemechanismus (event timer setInterval), der in regelmäßigen Abständen ein serverseitiges Skript aufruft, um zu prüfen, ob eine andere Datei heruntergeladen werden kann.

Wenn das serverseitige Skript ausgeführt wird, wenn es eine Drive-Datei findet, die den Suchkriterien entspricht *, wird die URL des Skripts zurückgegeben, das die Parameter enthält:

```
? servefile = true & id = the_id_of_the_google_drive_file
```

(* Die Suchkriterien in diesem einfachen Beispiel sind hart in das serverseitige Skript codiert. Sie können bei Bedarf problemlos vom Client an den Server übergeben werden.)

Diese Informationen werden über den erkannten withSuccessHandler-Mechanismus als Zeichenfolge an den Client zurückgegeben.

Das Java-Skript des Clients aktualisiert dann die HREF eines verborgenen Ankers mit diesen zurückgegebenen Informationen und klickt dann automatisch auf den Anker.

Dies bewirkt, dass ein weiterer Aufruf der App / des Skripts gestartet wird. Wenn der neue Aufruf der App gestartet wird, erkennt doGet den Parameter `servefile` und gibt die Datei nicht an die Benutzeroberfläche zurück, sondern an den Browser. Die zurückgegebene Datei wird durch den angegebenen ID-Parameter identifiziert, der zuvor von der oben beschriebenen Suche zurückgegeben wurde.

Da die Datei mit der angegebenen ID noch vorhanden ist, wird sie heruntergeladen und der neue Aufruf der App wird geschlossen, wobei der erste Aufruf zum Wiederholen dieses Vorgangs verbleibt.

Eine Schaltfläche steht auf der einfachen Oberfläche zur Verfügung, wenn der Benutzer / Tester ungeduldig auf den Timer wartet, aber nicht benötigt wird und ansonsten entfernt werden kann.

Das einfache Formular kann natürlich erweitert werden, um bei Bedarf eine umfassendere

Benutzeroberfläche bereitzustellen. B. die Dateisuchkriterien bereitstellen.

Google Web App-Skript für den automatischen Download von Google Drive online lesen:
<https://riptutorial.com/de/google-apps-script/topic/8212/google-web-app-skript-fur-den-automatischen-download-von-google-drive>

Kapitel 13: SpreadsheetApp Active Sheet

Bemerkungen

Methode: `getActive ()`

Rückgabety: [Tabellenkalkulation](#)

Examples

`getActive ()` - Aktive Kalkulationstabelle abrufen

Dies gibt die derzeit aktive Kalkulationstabelle oder null zurück, wenn keine vorhanden ist.

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

[SpreadsheetApp Active Sheet online lesen: https://riptutorial.com/de/google-apps-script/topic/5861/spreadsheetapp-active-sheet](https://riptutorial.com/de/google-apps-script/topic/5861/spreadsheetapp-active-sheet)

Kapitel 14: Tabellenkalkulationsmenü hinzufügen

Syntax

1. addMenu (Name, SubMenus)

Parameter

Name	Beschreibung
Name	der Name des zu erstellenden Menüs
subMenus	ein Array von JavaScript-Karten

Bemerkungen

Normalerweise möchten Sie addMenu über die Funktion onOpen aufrufen, damit das Menü beim Laden der Tabelle automatisch erstellt wird.

```
// 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

Erstellen Sie ein neues Menü

Erstellt ein neues Menü in der Tabellenkalkulationsoberfläche. Jeder Menüeintrag führt eine benutzerdefinierte Funktion aus.

```
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);
```

Benutzerdefiniertes Menü erstellen

/*

Methode: Benutzerdefiniertes Menü erstellen Dies ist die erste Funktion, die beim Laden der App aufgerufen wird

*/

```
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();  
}
```

Tabellenkalkulationsmenü hinzufügen online lesen: <https://riptutorial.com/de/google-apps-script/topic/4253/tabellenkalkulationsmenu-hinzufugen>

Kapitel 15: Tabellenkalkulationsservice

Bemerkungen

Die offizielle API-Referenz für den Spreadsheet Service finden Sie unter <https://developers.google.com/apps-script/reference/spreadsheet/>.

Examples

Blatt

Verweis auf eine Registerkarte mit dem Namen "Name" erhalten

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

Aktive Tabellenregisterkarte erhalten

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

Spalte einfügen

```
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
```

Zeile einfügen

```
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
```

Zellenwert

```
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);
```

Zellen kopieren

```
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));
```

Formel

```
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)");
```

Kopieren Sie einen Wert von einem Blatt in das aktuelle Blatt

Stellen Sie sich vor, wir haben eine separate Google-Tabelle und müssen den B2-Zellenwert in Zelle D5 in Ihrem aktuellen Arbeitsblatt angeben.

```
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  
}
```

Sie finden die ID der Tabelle in Ihrer URL.

Holen Sie sich die letzte Zeile in einer einzelnen Spalte

```
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;  
    }  
  }  
}
```

Einfügen von Arrays als Zeilen

Das Einfügen einer Zeile am unteren Rand einer Tabelle ist einfach:

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

Beachten Sie, dass dies die Zeile nach der letzten *nicht leeren* Zeile hinzufügt.

Das Einfügen einer Reihe irgendwo in der Mitte ist etwas mehr Arbeit:

```
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!
```

Viele dieser nutzlosen Codes können in eine Hilfsfunktion abstrahiert werden:

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

Was unser Beispiel auf nur reduziert:

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

Tabellenkalkulationsservice online lesen: <https://riptutorial.com/de/google-apps-script/topic/2688/tabellenkalkulationsservice>

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Google-Apps-Skript	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 ruft Google-Apps-Skript auf	Supertopoz
4	DriveApp	Brian , Kos , nibarius , Sandy Good , Wolfgang
5	DriveApp - getFileById (id)	Sandy Good
6	DriveApp Service	Sandy Good
7	DriveApp Service - Dateien nach Typ und Suchstring	nibarius , Sandy Good
8	Erstellen Sie eine benutzerdefinierte Funktion für Google Sheets	Francky_V , Joshua Dawson , Pierre-Marie Richard , Rubén
9	Firebase und AppScript: Einführung	Joseba , Vishal Vishwakarma
10	Google MailApp	nibarius
11	Google Sheets MailApp	Bhupendra Piprava , Brian , Jordan Rhea , Kos , nibarius , Saloni Vithalani
12	Google Web App-Skript für den automatischen Download von Google Drive	Walter
13	SpreadsheetApp Active Sheet	iJay
14	Tabellenkalkulationsmenü hinzufügen	Bishal , iJay , nibarius
15	Tabellenkalkulationsservice	cdrini , iJay , nibarius , Sandy Good , sudo bangbang