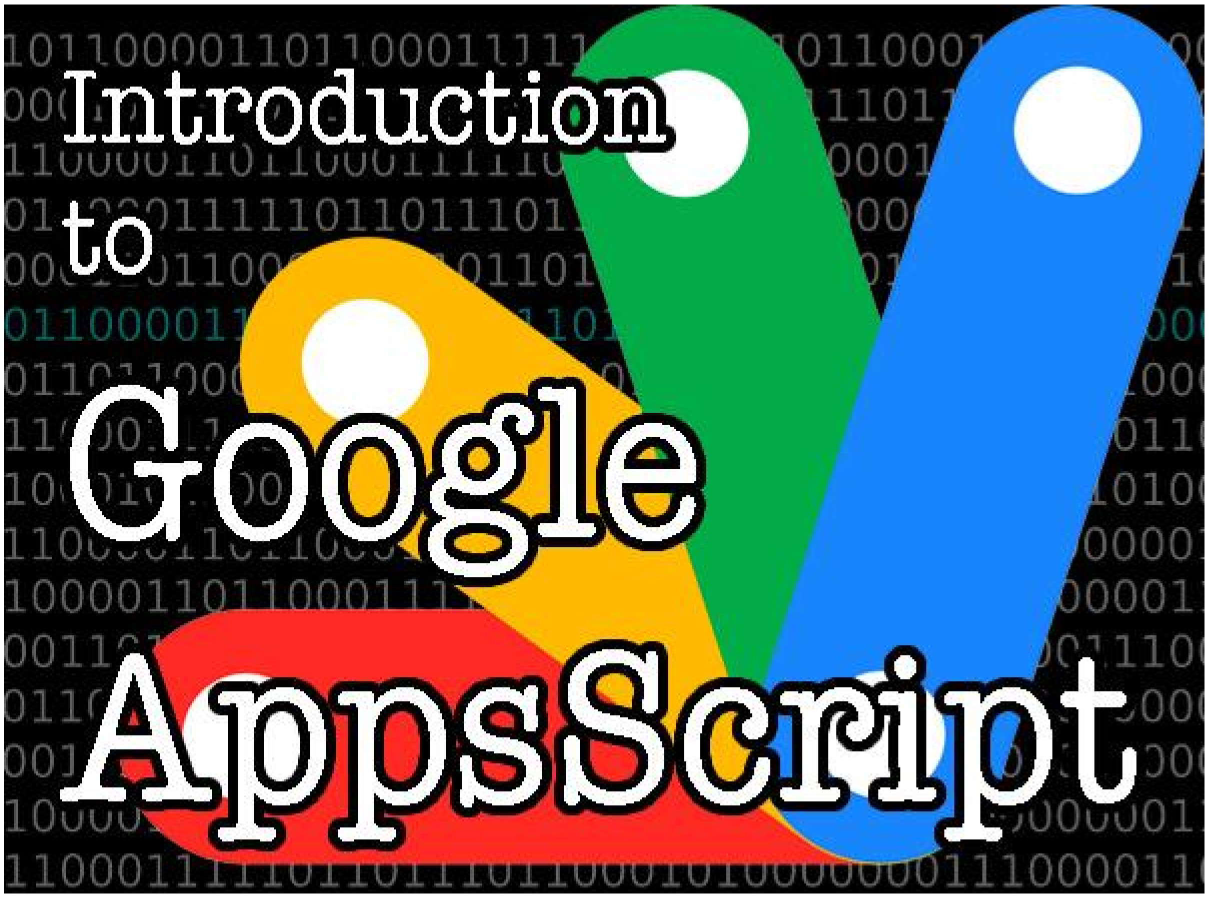
Unlocking the Power of Automation: A Beginner's Guide to Google Apps Script



In today's digital age, where efficiency and productivity are paramount, the ability to automate repetitive tasks can be a game-changer. Enter Google Apps Script, an incredibly versatile tool that empowers you to automate and extend the functionality of Google Workspace applications like Google Sheets, Google Docs, and Gmail.

In this beginner's guide, we'll take a journey into the world of Google Apps Script, unlocking its potential to simplify and enhance your daily workflow. Whether you're a business professional, educator, or simply an avid Google user, Google Apps Script has something to offer everyone.

What is Google Apps Script?

Google Apps Script is a server-side scripting language and runtime environment developed by Google. It allows users to create custom automation, build add-ons, and interact with Google Workspace applications programmatically. The beauty of Google Apps Script lies in its seamless integration with Google's suite of products, making it an invaluable resource for streamlining tasks and enhancing your digital experiences.

Why Learn Google Apps Script?

Automation: With Google Apps Script, you can automate repetitive tasks, such as generating reports, sending customized emails, and updating spreadsheet data. Say goodbye to manual drudgery!

- Customization: Tailor your Google Workspace applications to your specific needs. Create custom functions, menus, and dialogs to enhance your productivity.
- Integration: Seamlessly integrate Google Apps Script with other Google services and third-party APIs. Connect your Google Workspace applications to external data sources and services for a truly customized experience.
- Collaboration: Share your Google Apps Script projects with colleagues and collaborate in real-time. Extend the power of automation to your entire team.

What Can You Do with Google Apps Script?

The possibilities with Google Apps Script are nearly limitless. Here are just a few examples of what you can achieve:

- Automate Email Responses: Create custom email responders, schedule emails, or process incoming emails to streamline communication.
- Data Manipulation: Perform complex data transformations, data cleansing, and analysis in Google Sheets.
- Document Automation: Generate personalized documents, reports, or certificates in Google Docs.
- Forms Enhancement: Customize Google Forms with dynamic features, validation, and integration with external databases.
- Chatbots: Build interactive chatbots for Google Chat or Gmail to provide instant responses to common queries.
- Workflow Automation: Create workflows that trigger actions based on specific events, such as form submissions or changes in a spreadsheet.
- Add-ons: Develop and publish Google Workspace add-ons to enhance the functionality of Google Workspace applications for yourself or others.

Getting Started

In the coming articles of this series, we will delve into the fundamentals of Google Apps Script. You'll learn how to write your first scripts, understand key concepts, and gradually explore more advanced topics. Whether you have a programming background or are a complete novice, this guide is designed to help you harness the power of automation with Google Apps Script.

10 Coding Exercises for Google Apps Script

Exercise 1: Create a Custom Menu in Google Sheets

Description: Create a custom menu in Google Sheets that, when clicked, runs a Google Apps Script function to display a message.

Steps:

Open a Google Sheets document.

Click on "Extensions" in the menu bar, then select "Apps Script."

Replace any existing code with the following script:

```
function onOpen() {
  var ui = SpreadsheetApp.getUi();
  ui.createMenu('Custom Menu')
      .addItem('Show Message', 'showMessage')
      .addToUi();
}

function showMessage() {
  SpreadsheetApp.getUi().alert('Hello, Google Apps
Script!');
}
```

Save the script and return to your Google Sheets document.

You should now see a "Custom Menu" option in the menu bar. Click it, and you'll find the "Show Message" option that displays a message when clicked.

Exercise 2: Automate Sending Emails from Google Sheets

Description: Write a Google Apps Script that reads data from a Google Sheets spreadsheet and sends email notifications based on certain conditions.

Steps:

Open a Google Sheets document with data.

Click on "Extensions" in the menu bar, then select "Apps Script."

Replace any existing code with the following script:

```
function sendEmails() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  var data = sheet.getDataRange().getValues();

  for (var i = 1; i < data.length; i++) {
    var name = data[i][0];
    var email = data[i][1];
    var score = data[i][2];

    if (score < 70) {
      var subject = 'Exam Results';
      var message = 'Hi ' + name + ', your score of ' +
score + ' is below passing.';
      MailApp.sendEmail(email, subject, message);
    }
  }
}
```

Save the script and return to your Google Sheets document.

In the script editor, click the "play" button to run the sendEmails function. It will send emails to students with scores below 70.

Exercise 3: Create a Google Form Using Google Apps Script

Description: Write a Google Apps Script that generates a Google Form with specific questions.

Steps:

Open a Google Sheets document.

Click on "Extensions" in the menu bar, then select "Apps Script."

Replace any existing code with the following script:

```
function createForm() {
  var form = FormApp.create('My Sample Form');

  var item = form.addTextItem();
  item.setTitle('What is your name?');
  var multipleChoiceItem =
form.addMultipleChoiceItem();
  multipleChoiceItem.setTitle('What is your favorite
color?')
      .setChoiceValues(['Red', 'Blue', 'Green',
'Yellow']);

  Logger.log('Form URL: ' + form.getPublishedUrl());
  Logger.log('Form ID: ' + form.getId());
}
```

Save the script and run the createForm function.

Check the "Logs" in the script editor for the form URL and ID.

Open the generated form using the URL.

Exercise 4: Create a Google Document with Google Apps Script

Description: Write a Google Apps Script that creates a new Google Docs document and adds content to it.

Steps:

Open a Google Docs document.

Click on "Extensions" in the menu bar, then select "Apps Script."

Replace any existing code with the following script:

```
function createDocument() {
  var doc = DocumentApp.create('My Sample Document');
  var body = doc.getBody();

  body.appendParagraph('This is a sample document 
created with Google Apps Script.');
  body.appendParagraph('You can add text, lists, 
images, and more!');

  Logger.log('Document URL: ' + doc.getUrl());
}
```

Save the script and run the createDocument function.

Check the "Logs" in the script editor for the document URL.

Open the generated document using the URL.

Exercise 5: Create a Custom Sidebar in Google Sheets

Description: Write a Google Apps Script that creates a custom sidebar in Google Sheets with interactive elements.

Steps:

Open a Google Sheets document.

Click on "Extensions" in the menu bar, then select "Apps Script."

Replace any existing code with the following script:

```
function showSidebar() {
  var htmlOutput = 
HtmlService.createHtmlOutputFromFile('Sidebar')
```

```
      .setTitle('Custom Sidebar')
      .setWidth(300);
  SpreadsheetApp.getUi().showSidebar(htmlOutput);
}

function doGet() {
  return
HtmlService.createHtmlOutputFromFile('Sidebar');
}
```

Create an HTML file named "Sidebar" by clicking on the "+" button in the script editor and selecting "HTML." Add the following HTML content:

```
<!DOCTYPE html>
<html>
  <head>
    <base target="_top">
  </head>
  <body>
    <h1>Welcome to the Custom Sidebar</h1>
    <button onclick="showMessage()">Show
Message</button>
    <script>
      function showMessage() {
        google.script.run.showDialog();
      }
    </script>
  </body>
</html>
```

Save both the script and the HTML file.

In the script editor, run the showSidebar function to open the custom sidebar in Google Sheets.

Click the "Show Message" button to trigger the showDialog function.

Exercise 6: Create a Custom Dialog Box in Google Sheets

Description: Write a Google Apps Script that displays a custom dialog box in Google Sheets with user input fields.

Steps:

Open a Google Sheets document.

Click on "Extensions" in the menu bar, then select "Apps Script."

Replace any existing code with the following script:

```
function showDialog() {
  var htmlOutput =
HtmlService.createHtmlOutputFromFile('Dialog')
      .setWidth(300)
      .setHeight(200);
  SpreadsheetApp.getUi().showModalDialog(htmlOutput,
'Custom Dialog');
}

function doGet() {
  return
HtmlService.createHtmlOutputFromFile('Dialog');
}
```

Create an HTML file named "Dialog" by clicking on the "+" button in the script editor and selecting "HTML." Add the following HTML content:

```
<!DOCTYPE html>
<html>
  <head>
    <base target="_top">
  </head>
  <body>
    <h1>Custom Dialog</h1>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name"><br><br>
    <button onclick="submitForm()">Submit</button>
    <script>
      function submitForm() {
        var name =
document.getElementById("name").value;
        google.script.run.processForm(name);
        google.script.host.close();
      }
    </script>
  </body>
</html>
```

Save both the script and the HTML file.

In the script editor, run the showDialog function to open the custom dialog box in Google Sheets.

Enter a name and click "Submit" to trigger the processForm function.

Exercise 7: Create a Time-Driven Trigger in Google Apps Script

Description: Write a Google Apps Script that runs on a time-driven trigger to send daily email reminders.

Steps:

Open a Google Sheets document.

Click on "Extensions" in the menu bar, then select "Apps Script."

Replace any existing code with the following script:

```
function sendDailyReminder() {
  var email = Session.getActiveUser().getEmail();
  var subject = 'Daily Reminder';
  var message = 'Don\'t forget to complete your tasks 
today!';
  MailApp.sendEmail(email, subject, message);
}
```

Save the script.

In the script editor, click the clock icon to open the triggers dashboard.

Click "Add Trigger" to create a time-driven trigger for the sendDailyReminder function. Set it to run daily at a specific time.

The script will send a daily reminder email to the user.

Exercise 8: Create a Google Add-On

Description: Write a Google Apps Script that turns into a Google Workspace add-on to extend Google Workspace applications.

Steps:

Open a Google Sheets document.

Click on "Extensions" in the menu bar, then select "Apps Script."

Replace any existing code with the following script:

```
function onOpen() {
  var ui = SpreadsheetApp.getUi();
  ui.createAddonMenu()
      .addItem('Open Sidebar', 'showSidebar')
      .addToUi();
}

function showSidebar() {
  var htmlOutput =
HtmlService.createHtmlOutputFromFile('Sidebar')
      .setTitle('Custom Sidebar')
      .setWidth(300);
  SpreadsheetApp.getUi().showSidebar(htmlOutput);
}

function doGet() {
  return
HtmlService.createHtmlOutputFromFile('Sidebar');
}
```

Create an HTML file named "Sidebar" (similar to Exercise 5).

Save both the script and the HTML file.

In the script editor, click the puzzle piece icon to configure the add-on.

Create an add-on by providing a name, version, and description.

Test the add-on by opening a Google Sheets document and using the "Add-ons" menu to trigger the "Open Sidebar" action.

Exercise 9: Create a Google Workspace Document from Google Forms Submission

Description: Write a Google Apps Script that creates a new Google Docs document when a Google Form is submitted and populates it with form responses.

Steps:

Create a Google Form with fields you want to capture.

Open the associated Google Sheets document where form responses are collected.

Click on "Extensions" in the menu bar, then select "Apps Script."

Replace any existing code with the following script:

```
function createDocument(e) {
  var formResponses = e.values;
  var name = formResponses[1]; // Assuming name is the second question
  var doc = DocumentApp.create('Response Document: ' + name);
  var body = doc.getBody();

  for (var i = 0; i < formResponses.length; i++) {
    body.appendParagraph('Question ' + (i + 1) + ': ' + formResponses[i]);
  }
}
```

Save the script.

Click on the form icon in the script editor to configure the trigger for form submissions.

Set up the trigger to run the createDocument function when the form is submitted.

When someone submits the form, a new Google Docs document will be created with their responses.

Exercise 10: Create a Google Calendar Event from Google Sheets Data

Description: Write a Google Apps Script that reads data from a Google Sheets spreadsheet and creates Google Calendar events based on that data.

Steps:

Open a Google Sheets document with data about events (e.g., event name, date, time).

Click on "Extensions" in the menu bar, then select "Apps Script."

Replace any existing code with the following script:

```
function createCalendarEvents() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  var data = sheet.getDataRange().getValues();

  var calendar = CalendarApp.getDefaultCalendar(); //
You can also specify a specific calendar
```

```
  for (var i = 1; i < data.length; i++) {
    var event = {
      title: data[i][0],
      description: data[i][1],
      location: data[i][2],
      startTime: new Date(data[i][3]),
      endTime: new Date(data[i][4])
    };

    calendar.createEvent(event.title, event.startTime,
event.endTime, event);
  }
}
```

Save the script.

In the script editor, create a time-driven trigger to run the createCalendarEvents function at your desired frequency.

When the script runs, it will create calendar events based on the data in the spreadsheet.

These exercises provide a practical introduction to Google Apps Script and demonstrate how to automate tasks, integrate Google services, and extend the functionality of Google Workspace applications. As you work through these exercises, you'll gain valuable experience and become more proficient in using Google Apps Script for various purposes. Happy scripting!

So, are you ready to streamline your work, boost your productivity, and explore the endless possibilities of automation? Let's embark on this exciting journey together and unlock the true potential of Google Apps Script!