🚀 Boost Your Google Workspace Skills: Exciting Google Apps Script Exercises! 📊👩‍💻

Exercise 1: Create a Custom Menu in Google Sheets

Objective: To create a custom menu in Google Sheets that triggers a simple script.

Code Sample:

```
function onOpen() {
  var ui = SpreadsheetApp.getUi();
  ui.createMenu('Custom Menu')
```

```
      .addItem('Show Alert', 'showAlert')
      .addToUi();
}

function showAlert() {
  SpreadsheetApp.getUi().alert('Hello, Google
Sheets!');
}
```

Explanation:

This script creates a custom menu named "Custom Menu" in the Google Sheets UI. When "Show Alert" is clicked, it triggers the showAlert function that displays a simple alert box with a message.

Exercise 2: Read Data from a Range in Google Sheets

Objective: Read data from a specified range in a sheet and log it.

Code Sample:

```
function readRangeData() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  var range = sheet.getRange("A1:B2");
  var values = range.getValues();
```

```
  for (var i = 0; i < values.length; i++) {
    var row = values[i];
    Logger.log(row[0] + ", " + row[1]);
  }
}
```

Explanation:

This script reads data from the range A1:B2 of the active sheet. getValues returns a two-dimensional array of values, which is then iterated over to log each cell's data.

Exercise 3: Send an Email using GmailApp

Objective: Use Google Apps Script to send an email via Gmail.

Code Sample:

```
function sendEmail() {
  var recipient = "example@example.com";
  var subject = "Test Email from Google Apps Script";
  var body = "This is a test email sent from Google
Apps Script.";

  GmailApp.sendEmail(recipient, subject, body);
}
```

Explanation:

This script sends an email using the Gmail service. Replace recipient with the desired email address. The GmailApp.sendEmail method is used to send the email.

Exercise 4: Create a Google Doc

Objective: Programmatically create a Google Doc with some content.

Code Sample:

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

  body.appendParagraph('This is a new Google Document created by Google Apps Script.');

  doc.saveAndClose();
}
```

Explanation:

This script creates a new Google Document titled "New Document" and adds a paragraph of text to it. The DocumentApp.create method is used to create the document.

Exercise 5: Modify Cell Values in Google Sheets

Objective: Write a script to modify cell values in a Google Sheet.

Code Sample:

```
function updateCellValues() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  sheet.getRange("A1").setValue("Hello");
  sheet.getRange("B1").setValue("World");
}
```

Explanation:

This script updates the values of cells A1 and B1 of the active sheet to "Hello" and "World", respectively. The setValue method is used to set the value of a cell.

Exercise 6: Create an Automated Reminder in Google Calendar

Objective: Write a script to create a calendar event and set an email reminder.

Code Sample:

```
function createCalendarEvent() {
  var calendar = CalendarApp.getDefaultCalendar();
  var startTime = new Date('March 15, 2024 10:00:00');
  var endTime = new Date('March 15, 2024 11:00:00');
```

```
  var event = calendar.createEvent('Meeting with Team',
startTime, endTime);

  event.addEmailReminder(30); // 30 minutes before
}
```

Explanation:

This script creates an event in the user's default Google Calendar. It sets up a meeting titled 'Meeting with Team' at a specified date and time. An email reminder is added 30 minutes before the event starts.

Exercise 7: Access and Modify Google Drive Files

Objective: List all files in the Google Drive root folder and log their names.

Code Sample:

```
function listDriveFiles() {
  var files = DriveApp.getRootFolder().getFiles();
  while (files.hasNext()) {
    var file = files.next();
    Logger.log(file.getName());
  }
}
```

Explanation:

This script accesses the root folder of Google Drive and iterates through all files in it. It logs the name of each file found, which can be viewed in the Google Apps Script's Logger.

Exercise 8: Fetch Data from an External API

Objective: Write a script to fetch data from an external API and log the response.

Code Sample:

```
function fetchDataFromAPI() {
  var response = UrlFetchApp.fetch("https://api.example.com/data");
  Logger.log(response.getContentText());
}
```

Explanation:

This script uses the UrlFetchApp service to make a GET request to an external API. It logs the response content, which can be JSON, XML, or plain text, depending on the API.

Exercise 9: Manipulate Spreadsheet Formatting

Objective: Change the background color of the first row in a Google Sheet.

Code Sample:

```
function formatFirstRow() {
```

```
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  var range = sheet.getRange("1:1"); // First row
  range.setBackground("yellow");
}
```

Explanation:

This script gets the active sheet and selects the first row. It then changes the background color of this row to yellow, demonstrating basic cell formatting capabilities.

Exercise 10: Sync Spreadsheet Data to Google Calendar

Objective: Create Google Calendar events based on data from a Google Sheets spreadsheet.

Code Sample:

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

  for (var i = 1; i < data.length; i++) {
    var row = data[i];
```

```
    var title = row[0]; // Event title in the first
column
    var startTime = new Date(row[1]); // Start time in
the second column
    var endTime = new Date(row[2]); // End time in the
third column
    CalendarApp.getDefaultCalendar().createEvent(title,
startTime, endTime);
  }
}
```

Explanation:

This script reads data from the active sheet, assuming that each row contains an event title, start time, and end time. It loops through each row and creates corresponding events in the user's default Google Calendar. This is a basic example of how to sync spreadsheet data with Google Calendar.