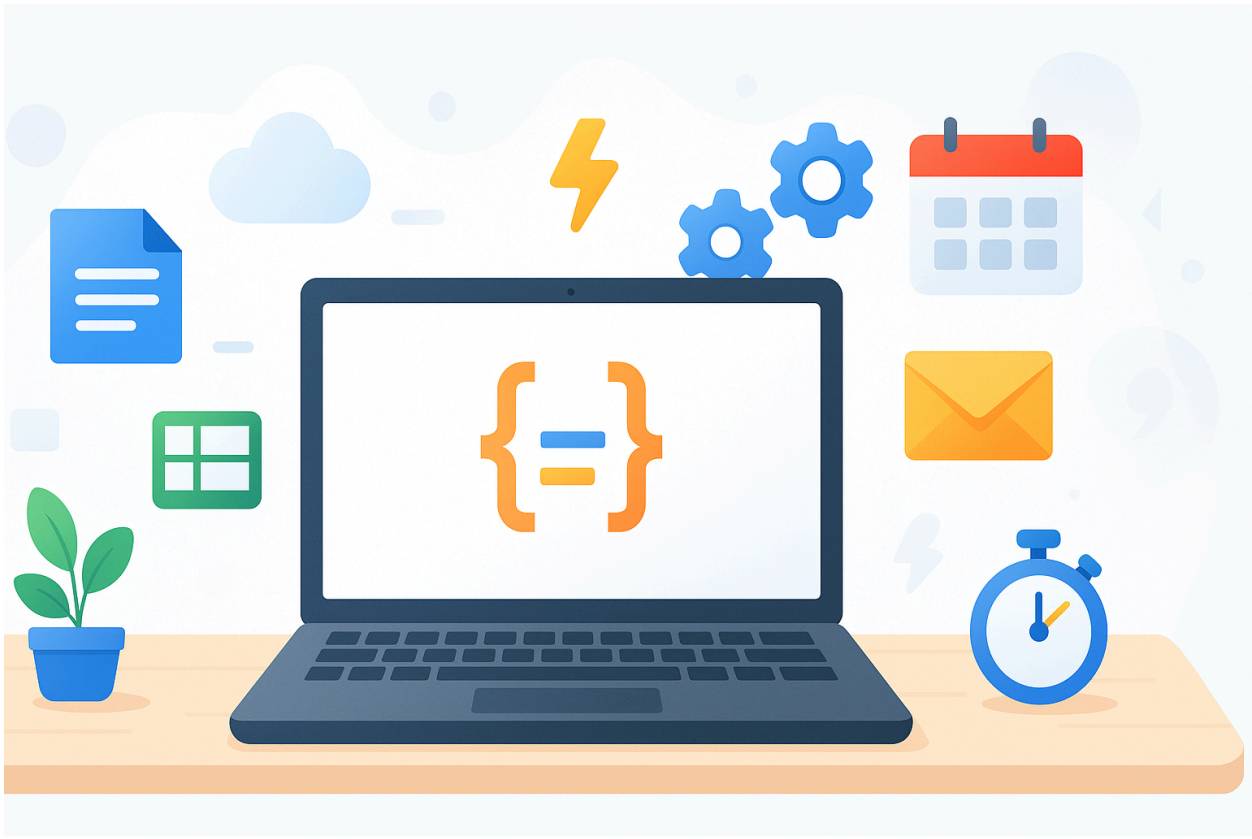

Comprehensive 28-day guide to learning Google Apps Script, designed for complete beginners. Each day's lesson should take less than an hour to complete.

How to Use This Guide

Welcome to your 28-day journey into Google Apps Script! This guide is designed to take you from a complete beginner to someone who can confidently automate tasks and build small applications within the Google Workspace ecosystem.

Your Daily Plan (Under 1 Hour):

- **Read the Explanation:** Start by reading the core concepts for the day. We'll keep it simple and to the point.
- **Analyze the Code:** Review the provided code example. Each line is explained so you know exactly what's happening.
- **Run the Example:** Type the code into the Apps Script editor yourself. Running the code and seeing the result is a crucial learning step.
- **Complete the Exercise:** Apply what you've learned with a small, practical exercise.
- **Take the Quiz:** Test your understanding with a few quick questions. The answers are explained to reinforce the concepts.
- **Use the AI Prompts:** This is your secret weapon. Copy and paste the prompts into an AI like Gemini or ChatGPT to explore topics more

deeply, ask for alternative examples, or clarify anything you're unsure about. It's like having a personal tutor.

To get started, open a new Google Sheet by typing **sheets.new** into your browser's address bar, then go to **Extensions > Apps Script** to open the editor where you'll be working. Let's begin!

Week 1: The Absolute Basics

Day 1: Your First Script - "Hello, World!"

Learning Objective: Understand what Google Apps Script is, how to access the script editor, and how to write and run a basic function that logs a message.

Content Explanation:
Google Apps Script is a cloud-based scripting language based on JavaScript. It allows you to automate tasks across Google products like Sheets, Docs, and Gmail. You don't need to install any software; all your code is written and runs on Google's servers. You'll write your code in the Apps Script editor, which you can open from any Google Workspace file (like a Sheet) by going to Extensions > Apps Script.

Code Example:
This function will print a message to the Execution Log, a panel that helps you see what your script is doing.
JavaScript

```
function sayHello() {
  // Logger.log() is used to print information to the log for debugging.
  Logger.log("Hello, World!");
}
```

function sayHello() { ... }: This declares a block of code called a **function** named sayHello.

Logger.log(...): This is a built-in command that writes text into the execution log. It's your most important tool for checking your work.

To Run This Code:

Click the **Save project** icon (looks like a floppy disk).

Select sayHello from the function dropdown menu above the code.

Click **Run**.

Authorize the script: The first time you run a project, Google will ask for permission. Click "Review permissions," choose your account, click "Advanced," then "Go to (unsafe)," and finally "Allow."

After it runs, check the **Execution log** at the bottom of the screen. You should see "Hello, World!".

Exercise:
Create a new function named logMyFavoriteFood. Inside this function, use Logger.log() to print the name of your favorite food. Run the function and check the log.

Quiz:

What is the primary purpose of Logger.log()?

a) To save the file.

b) To display pop-up messages to the user.

c) To print information in the execution log for the developer.

Where do you go in Google Sheets to open the Apps Script editor?

a) File > New Script

b) Extensions > Apps Script

c) Tools > Macros

Answers & Explanations:

(c) Logger.log() is a developer tool for debugging and seeing the internal state of a script.

(b) The script editor is always found under the Extensions menu.

AI Prompts for Deeper Learning:

"What are three common real-world tasks that can be automated with Google Apps Script?"

"Explain the difference between the Apps Script Logger and the JavaScript console.log()."

Day 2: Variables and Data Types

Learning Objective: Understand how to store and use information using variables and learn about the most common data types.

Content Explanation:
A variable is like a labeled box where you can store information. You give it a name and put data inside it. In modern JavaScript (and thus Apps Script), we create variables using let or const. Use let for variables whose value might change, and const for variables that will never change (constants). Common data types include:

String: Text, enclosed in quotes (e.g., "Hello").

Number: Any number, with or without decimals (e.g., 10, 3.14).

Boolean: A true or false value.

Code Example:

JavaScript

```
function learnVariables() {
  let recipientName = "Casey";
  const emailSubject = "Meeting Reminder";
  let meetingHour = 14; // Using 24-hour time
  let isUrgent = true;

  Logger.log(recipientName); // Logs: Casey
  Logger.log(emailSubject); // Logs: Meeting Reminder
  Logger.log(meetingHour);   // Logs: 14.0
  Logger.log(isUrgent);      // Logs: true
}
```

let recipientName = "Casey";: We declare a variable recipientName and store the string "Casey" in it.

const emailSubject = ...: We declare a constant emailSubject. We can't change its value later.

let meetingHour = 14;: We store the number 14. Notice numbers don't use quotes.

let isUrgent = true;: We store the boolean value true.

Exercise:
Create a function called myProfile. Inside it, create three variables:

myName (a string with your name).

myAge (a number with your age).

likesPizza (a boolean, true or false).
Use Logger.log() to print each of these variables.

Quiz:

Which line of code correctly declares a variable that can be changed later?

a) constant myName = "Alex";

b) let myName = "Alex";

c) variable myName = "Alex";

What data type is the value false?

a) String

b) Number

c) Boolean

Answers & Explanations:

(b) let is used for variables that you intend to reassign later.

(c) true and false are the two possible boolean values.

AI Prompts for Deeper Learning:

"Explain the difference between let, const, and var in JavaScript. Which one should I use in Google Apps Script and why?"

"Show me an example of how to combine a string and a number variable into a single message using Google Apps Script."

Day 3: Introduction to Google Sheets - Getting a Value

Learning Objective: Learn how to connect your script to the spreadsheet it's attached to and read a value from a specific cell.

Content Explanation:
This is where Apps Script gets powerful! You can control Google Sheets with code. The SpreadsheetApp service is your gateway to interacting with the spreadsheet. To get data, you follow a path: SpreadsheetApp -> Active Spreadsheet -> A Specific Sheet -> A Specific Cell (Range) -> The Value.

Code Example:
Setup: In your Google Sheet, type "First Name" in cell A1.
JavaScript

```
function readCellA1() {
  // 1. Get the currently active spreadsheet.
  const ss = SpreadsheetApp.getActiveSpreadsheet();

  // 2. Get the sheet named "Sheet1".
  const sheet = ss.getSheetByName("Sheet1");

  // 3. Get the cell range "A1".
```

```
  const cell = sheet.getRange("A1");

  // 4. Get the actual value from that cell.
  const value = cell.getValue();

  // 5. Log the value to see it.
  Logger.log(value); // Should log: First Name
}
```

SpreadsheetApp.getActiveSpreadsheet(): This is the starting point. It gets the entire spreadsheet file your script is bound to.

.getSheetByName("Sheet1"): From the spreadsheet, we grab a specific sheet by its name (the name in the tab at the bottom).

.getRange("A1"): On that sheet, we specify the cell we want. "A1" is standard spreadsheet notation.

.getValue(): Finally, we extract the data that's inside that cell.

Exercise:

In your spreadsheet, type your city in cell **B2**.

Create a new function readMyCity.

Inside the function, write the code to read the value from cell **B2** of "Sheet1".

Use Logger.log() to print the value. Run it and check the log.

Quiz:

What service is the main entry point for controlling Google Sheets?

a) SheetService

b) GoogleSheet

c) SpreadsheetApp

Which method is used to get the data *out of* a cell range object?

a) .readCell()

b) .getValue()

c) .extractData()

Answers & Explanations:

(c) SpreadsheetApp is the global object you always start with for sheet manipulation.

(b) After you get a range with .getRange(), you must call .getValue() to read its contents.

AI Prompts for Deeper Learning:

"In Google Apps Script, what's the difference between getActiveSpreadsheet() and openById()?"

"How would I read the value from cell C5 instead of A1 in the example script? Show me the modified line of code."

Day 4: Writing a Value to a Google Sheet

Learning Objective: Learn how to use Apps Script to write or change a value in a specific cell in your spreadsheet.

Content Explanation:
Writing to a cell is very similar to reading. You follow the same path to get the cell you want (SpreadsheetApp -> Sheet -> Range), but instead of using .getValue(), you use the .setValue() method.

Code Example:
This script will write the text "Status: Complete" into cell B1.
JavaScript

```
function writeToCellB1() {
  // 1. Get the spreadsheet and the specific sheet.
  const sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Sheet1");

  // 2. Get the target cell range "B1".
  const cell = sheet.getRange("B1");

  // 3. Set the value of that cell.
  cell.setValue("Status: Complete");

  Logger.log("Value has been written to B1!");
}
```

cell.setValue("Status: Complete"): This is the key command. It takes one argument—the data you want to place in the cell—and writes it, overwriting

anything that was there before.

Exercise:

Create a function called addTimestamp.

Inside the function, get the current date and time by creating a new Date object: const now = new Date();.

Write the code to put this now variable into cell **C1** of your sheet.

Run the script and check your spreadsheet to see the timestamp appear.

Quiz:

What method do you use to write data *into* a cell?

a) .putValue()

b) .writeValue()

c) .setValue()

What will happen if you use .setValue() on a cell that already contains data?

a) It will add the new data next to the old data.

b) It will return an error.

c) It will overwrite the existing data.

Answers & Explanations:

(c) The method to remember is .setValue().

(c) .setValue() is a destructive action; it replaces the cell's current content.

AI Prompts for Deeper Learning:

"How can I write the same value to multiple cells at once in Google Apps Script, for example, to the range A1:A5?"

"Show me how to clear the content of a cell, like B1, using Google Apps Script."

Day 5: Simple Math and String Concatenation

Learning Objective: Perform basic arithmetic operations (add, subtract) and combine strings (concatenation) using variables.

Content Explanation:
You can do math in Apps Script just like on a calculator. You can also

combine strings using the + operator. This is called concatenation. It's useful for building dynamic messages.

Code Example:

JavaScript

```
function doSomeMath() {
  // --- Math Operations ---
  let quantity = 5;
  let price = 10;
  let totalCost = quantity * price; // Multiplication
  Logger.log(totalCost); // Logs: 50.0

  let balance = 100;
  let withdrawal = 20;
  let newBalance = balance - withdrawal; // Subtraction
  Logger.log(newBalance); // Logs: 80.0

  // --- String Concatenation ---
  let firstName = "Alex";
  let lastName = "Chen";
  // We add a space " " in the middle to separate the names.
  let fullName = firstName + " " + lastName;
  Logger.log(fullName); // Logs: Alex Chen

  // --- Combining Strings and Numbers ---
  let item = "apples";
  let itemCount = 12;
  let message = "You have " + itemCount + " " + item + ".";
  Logger.log(message); // Logs: You have 12 apples.
}
```

Exercise:

Create a function called calculateBill.

Inside, create two variables: billAmount (set to 50) and tipPercentage (set to 0.15).

Create a third variable, tipAmount, by multiplying billAmount by tipPercentage.

Create a fourth variable, totalBill, by adding billAmount and tipAmount.

Log a final message like "Total to pay: $" followed by the totalBill value.

Quiz:

What is the result of "Hello " + "World"?

a) HelloWorld

b) Hello World

c) An error

Which symbol is used for multiplication?

a) x

b) *

c) &

Answers & Explanations:

(b) The + operator joins the strings exactly as they are. The space inside the first string is preserved.

(b) The asterisk * is the standard programming symbol for multiplication.

AI Prompts for Deeper Learning:

"Explain what template literals are in JavaScript and show me how to rewrite the 'message' variable from today's lesson using them. Why are they often better for combining strings and variables?"

"What is the modulo operator (%) in programming? Give me a simple Google Apps Script example."

Day 6: Simple if Statements

Learning Objective: Understand how to make decisions in your code using a simple if statement to run code only when a certain condition is true.

Content Explanation:
An if statement checks if a condition is true. If it is, the code inside its curly braces {} will run. If the condition is false, the code block is skipped entirely. This allows your script to make decisions. We use comparison operators like > (greater than), < (less than), and === (strictly equal to).

Code Example:

This script checks a value in cell A1 and writes a message in B1 only if the value is "Complete".
Setup: In your sheet, type Complete into cell A1.
JavaScript

```
function checkStatus() {
  const sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Sheet1");
  const statusCell = sheet.getRange("A1");
  const statusValue = statusCell.getValue();

  // Check if the value in A1 is exactly equal to the string "Complete".
  if (statusValue === "Complete") {
    // This code only runs if the condition is true.
    sheet.getRange("B1").setValue("Task has been verified.");
    Logger.log("Status was complete. Message written.");
  }

  Logger.log("Script finished.");
}
```

if (statusValue === "Complete") { ... }: The condition is inside the parentheses (). We use the triple equals === to check for exact equality (value and type). If statusValue holds the string "Complete", the code inside the {} runs. If you change A1 to "In Progress" and run it again, the message in B1 will not appear.

Exercise:

In cell **A2**, enter a number (e.g., 15).

Create a function checkScore.

The function should read the number from **A2**.

Use an if statement to check if the score is greater than 10.

If it is, write "Pass" into cell **B2**.

Test your script with a number greater than 10 and one less than 10.

Quiz:

Which operator is used to check if two values are strictly equal?

a) =

b) ==

c) ===

In an if (condition) { ... }, when does the code inside the {} run?

a) Always.

b) Only when the condition is true.

c) Only when the condition is false.

Answers & Explanations:

(c) === is the strict equality operator. A single = is for assignment (placing a value in a variable).

(b) The if block is conditional and executes only when its condition evaluates to true.

AI Prompts for Deeper Learning:

"Explain the difference between == and === in JavaScript with examples. Why is it almost always better to use ===?"

"Show me how to use an if...else statement in Google Apps Script to write 'Pass' if a score is over 10, and 'Fail' otherwise."

Day 7: Writing Your First Useful Function

Learning Objective: Combine reading, decision-making, and writing to create a practical function that adds a timestamp when a task is marked as "Done".

Content Explanation:
Let's put everything from Week 1 together! We will create a function that you could actually use. The goal is to monitor a "Status" column. When a cell in that column is changed to "Done", our script will automatically put the current date and time in the cell next to it.

Code Example:
This function will be simple for now. We will manually check cell A1 and update B1. (In a later lesson, we'll learn how to make this run automatically!)
Setup: In cell A1, type Done.
JavaScript

```
function addTimestampIfDone() {
  // 1. Define our target cells.
  const statusCell = "A1";
  const timestampCell = "B1";

  // 2. Get the sheet.
  const sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Sheet1");

  // 3. Read the value from the status cell.
  const status = sheet.getRange(statusCell).getValue();

  // 4. Use an 'if' statement to make a decision.
  if (status === "Done") {
    // 5. If the status is "Done", write the current date to the timestamp cell.
    const now = new Date(); // Creates a date object for the current
moment.
    sheet.getRange(timestampCell).setValue(now);
    Logger.log("Task marked as Done. Timestamp added.");
  } else {
    // This 'else' block runs if the condition is false.
    Logger.log("Status is not 'Done'. No action taken.");
  }
}
```

Exercise:
Adapt the addTimestampIfDone function.

Create a new function called logCompletion.

It should check cell **A2** for the text "COMPLETE" (all caps).

If it finds it, it should write "Logged by script on [current date]" into cell **B2**. (Hint: You'll need to use string concatenation: "Logged by script on " + new Date()).

Quiz:

What does new Date() create?

a) A string with today's date, like "2025-09-10".

b) A special object representing the current date and time.

c) The number of seconds since 1970.

In the main example, if cell A1 contains "In Progress", what will be logged?

a) "Task marked as Done. Timestamp added."

b) "Status is not 'Done'. No action taken."

c) Nothing will be logged.

Answers & Explanations:

(b) new Date() creates a Date object, which Google Sheets knows how to format and display correctly as a date/time.

(b) Since "In Progress" is not equal to "Done", the if condition is false, and the code inside the else block is executed.

AI Prompts for Deeper Learning:

"How can I format the date from new Date() in Google Apps Script to look like 'YYYY-MM-DD'? Show me the code using Utilities.formatDate()."

"What is a simple trigger in Google Apps Script? How could I set one up to run my addTimestampIfDone function automatically whenever I edit the spreadsheet?"

Week 2: Interacting with Data

Day 8: Arrays - Storing Lists of Data

Learning Objective: Understand what an array is and how to create and access items in a list.

Content Explanation:
An array is a special variable that can hold more than one value at a time. Think of it as a numbered list. Each item in the list is called an element. The position of an element is its index, and indexing starts at 0.

Code Example:
JavaScript
```
function workWithArrays() {
  // An array of strings
  const taskList = ["Review report", "Call client", "Submit invoice"];

  // Accessing elements by index (remember, we start at 0!)
  const firstTask = taskList[0]; // "Review report"
  const secondTask = taskList[1]; // "Call client"

  Logger.log("The first task is: " + firstTask);
  Logger.log("The second task is: " + secondTask);

  // Finding the number of items in an array
  const numberOfTasks = taskList.length;
  Logger.log("You have " + numberOfTasks + " tasks."); // Logs: You have 3 tasks.

  // Adding an item to the end of an array
  taskList.push("Plan next week");
  Logger.log(taskList); // Logs: [Review report, Call client, Submit invoice, Plan next week]
}
```

Exercise:

Create a function myGroceries.

Inside, create an array named groceryList containing three food items you

want to buy.

Log the **first** item in the list.

Log the **total number** of items on your list.

Quiz:

What is the index of the *third* element in an array?

a) 1

b) 2

c) 3

Which property tells you how many elements are in an array named myArray?

a) myArray.count

b) myArray.size

c) myArray.length

Answers & Explanations:

(b) Array indexing is 0-based, so the first is at index 0, the second at 1, and the third at 2.

(c) The .length property is used to get the number of elements.

AI Prompts for Deeper Learning:

"Show me how to remove the last item from an array in Google Apps Script."

"How can I check if a specific item, like 'Call client', exists within my taskList array? Show me the code using the .includes() method."

Day 9: For Loops - Repeating Actions

Learning Objective: Learn how to use a for loop to repeat a block of code a specific number of times.

Content Explanation:
A for loop is a fundamental way to automate repetitive tasks. It lets you run the same code over and over again, often with a counter that changes each time. It has three parts:

Initialization: let i = 0; - Create a counter variable.

Condition: i < 5; - The loop continues as long as this is true.

Increment: i++ - Increase the counter by 1 after each cycle.

Code Example:

JavaScript

```
function simpleForLoop() {
  // This loop will run 5 times (for i = 0, 1, 2, 3, 4)
  for (let i = 0; i < 5; i++) {
    Logger.log("The current loop number is: " + i);
  }
}

// Looping through an array
function loopThroughTasks() {
  const taskList = ["Review report", "Call client", "Submit invoice"];

  // Loop from index 0 up to (but not including) the array's length
  for (let i = 0; i < taskList.length; i++) {
    const currentTask = taskList[i];
    Logger.log("Task " + (i + 1) + ": " + currentTask);
  }
}
```

Exercise:

Create a function countToTen.

Write a for loop that logs the numbers from 1 to 10. (Hint: Your condition might be i <= 10 or i < 11, and you might start with let i = 1).

Quiz:

In for (let i = 0; i < 3; i++), what are the values of i that the loop will run for?

a) 0, 1, 2, 3

b) 1, 2, 3

c) 0, 1, 2

What does i++ do?

a) Checks if i is positive.

b) Adds 1 to the value of i.

c) Doubles the value of i.

Answers & Explanations:

(c) The loop runs while i is *less than* 3. It stops when i becomes 3.

(b) i++ is shorthand for i = i + 1.

AI Prompts for Deeper Learning:

"What is a 'for...of' loop in JavaScript? Show me how to rewrite the loopThroughTasks function using it. Why is it sometimes simpler?"

"How can I use a for loop to write the numbers 1 through 5 into cells A1 through A5 in a Google Sheet?"

Day 10: Reading and Writing Multiple Cells

Learning Objective: Use your knowledge of arrays and loops to read a column of data from a sheet, process it, and write it back.

Content Explanation:
Apps Script can handle entire ranges of cells at once, which is much faster than reading/writing one cell at a time. When you use .getValues() (plural) on a range, you get back a 2D array (an array of arrays), where each inner array represents a row.

Code Example:
Setup: In your sheet, enter the following in column A:

A1: item_01

A2: item_02

A3: item_03

JavaScript

```
function processItemList() {
  const sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Sheet1");

  // Get the entire range of data from A1 to A3.
  const dataRange = sheet.getRange("A1:A3");
  const dataValues = dataRange.getValues(); // Returns a 2D array:
[[item_01], [item_02], [item_03]]

  const processedData = []; // Create a new empty array for our results.

  // Loop through each row of the 2D array.
```

```
  for (let i = 0; i < dataValues.length; i++) {
    const originalValue = dataValues[i][0]; // Get the item from the first
column of the current row.
    const newValue = originalValue.toUpperCase() + " - PROCESSED";
    processedData.push([newValue]); // Add the new value as an array (a
row) to our results.
  }

  // Write the new data to column B.
  sheet.getRange("B1:B3").setValues(processedData);
}
```

Exercise:

In cells A1:A3, enter three different numbers.

Create a function doubleTheNumbers.

Read the values from A1:A3.

Loop through them, multiply each number by 2, and store the results in a new array.

Write this new array to cells B1:B3 using .setValues().

Quiz:

.getValues() (plural) on a range returns what?

a) A simple array of values.

b) A 2D array (an array of rows).

c) A string with all values joined together.

Why is dataValues[i][0] used to get the value in the example?

a) dataValues[i] gets the row (an array), and [0] gets the first element in that row array.

b) It's a typo and should be dataValues[i].

c) It accesses the first sheet and first cell.

Answers & Explanations:

(b) Always remember getValues() returns an array of arrays, even for a single column.

(a) This is how you access elements in a 2D array: the first index for the outer array (row) and the second for the inner array (column).

AI Prompts for Deeper Learning:

"In Google Apps Script, what is the difference in performance between using setValue() in a loop versus using setValues() once? Why is setValues() preferred?"

"My data is in A1:C3. How would the getValues() call change, and how would I access the value from column C in the first row?"

Day 11: Introduction to Custom Functions

Learning Objective: Create a custom function in Apps Script that you can use directly in your Google Sheet like a built-in function (e.g., =SUM()).

Content Explanation:
A custom function is a special Apps Script function that can be called from a cell in your Google Sheet. It's a powerful way to add custom logic to your spreadsheets.
Rules for Custom Functions:

They must return a value.

They can only take numbers, strings, or booleans as input. They cannot, for example, access SpreadsheetApp.

The function name cannot have an underscore _ at the end.

Code Example:
This function takes a number and doubles it.
JavaScript

```
/**
 * Doubles the input value.
 *
 * @param {number} input The number to double.
 * @return The doubled number.
 * @customfunction
 */
function DOUBLE(input) {
  if (typeof input !== 'number') {
    return "Please enter a number.";
  }
  return input * 2;
}
```

/** ... */: This is a special comment block called **JSDoc**. It tells Google Sheets that this is a custom function and provides help text.

@customfunction: This tag is essential. It officially registers the function.

function DOUBLE(input): The function takes one argument, input. This will be the value from the cell or range you provide in the sheet.

return input * 2;: Instead of logging, custom functions must return the result so it can be displayed in the cell.

How to Use:

Save the script.

Go to any cell in your sheet and type =DOUBLE(5). The cell should display 10.

You can also reference another cell, like =DOUBLE(A1).

Exercise:

Create a custom function called GETINITIALS.

It should take one argument, a full name (e.g., "John Smith").

It should return the person's initials (e.g., "JS").

Hint: You can use fullName.split(" ") to turn the string into an array of words, then take the first letter of the first and second words.

Quiz:

What must a custom function do to display its result in a cell?

a) Use Logger.log().

b) Use sheet.setValue().

c) Use the return keyword.

Which of the following can a custom function NOT do?

a) Perform math calculations.

b) Access SpreadsheetApp to get another cell's value.

c) Combine two strings.

Answers & Explanations:

(c) The return statement passes the final value back to the cell that

called the function.

(b) Custom functions have security restrictions and cannot access most Google Workspace services.

AI Prompts for Deeper Learning:

"Write a Google Apps Script custom function that takes a US dollar amount and a tax rate (as a decimal) and returns the total amount including tax."

"Why can't a custom function in Google Apps Script modify a different cell? Explain the security reasons."

Day 12: Sending a Simple Email

Learning Objective: Use the MailApp service to send an email from your script.

Content Explanation:
Apps Script makes it incredibly easy to send emails. The MailApp service allows you to send emails from your Google account. This is perfect for notifications, reports, and alerts. When you run this for the first time, you will need to re-authorize your script, giving it permission to send email on your behalf.

Code Example:
JavaScript

```
function sendSimpleEmail() {
  // Get your own email address to send the email to yourself.
  const recipient = Session.getActiveUser().getEmail();
  const subject = "Test Email from Google Apps Script";
  const body = "This email was sent automatically from my script!";

  // The sendEmail method takes recipient, subject, and body as arguments.
  MailApp.sendEmail(recipient, subject, body);

  Logger.log("Email sent!");
}
```

Session.getActiveUser().getEmail(): A safe way to get the email address of the person running the script.

MailApp.sendEmail(...): This is the core command. It takes the three main

components of an email as arguments and sends it.

Exercise:

Create a function sendDataEmail.

It should read a value from cell **A1**.

It should send an email to yourself where the **subject line** is "Data from Sheet" and the **body** of the email is the value you read from cell A1.

Quiz:

What service is used for sending emails?

a) EmailApp

b) GmailApp

c) MailApp

What information is required by the basic MailApp.sendEmail() method?

a) Recipient, Subject, and Body.

b) Recipient and Subject only.

c) Just the recipient's email address.

Answers & Explanations:

(c) MailApp is the simple, direct service for sending mail. (GmailApp is more advanced and gives you more control, which we'll see later).

(a) You must provide these three core pieces of information to send a standard email.

AI Prompts for Deeper Learning:

"What is the difference between MailApp and GmailApp in Google Apps Script? Give an example of a task that only GmailApp can do."

"How can I modify my sendSimpleEmail function to send the email to multiple recipients at once?"

Day 13: Creating UI - Simple Alerts and Prompts

Learning Objective: Learn how to create simple user interface elements like alerts and prompts to interact with the user.

Content Explanation:
Sometimes you need to show a message or ask the user for input. The

SpreadsheetApp.getUi() service gives you access to the user interface of the spreadsheet, allowing you to create simple pop-up dialogs.

Code Example:

JavaScript

```
function showDialogs() {
  // Get the UI environment of the spreadsheet.
  const ui = SpreadsheetApp.getUi();

  // 1. A simple alert dialog.
  ui.alert("This is a simple notification!");

  // 2. An alert with a title and buttons.
  const response = ui.alert("Confirmation Needed", "Do you want to
proceed?", ui.ButtonSet.YES_NO);

  // Check which button was clicked.
  if (response == ui.Button.YES) {
    Logger.log("The user clicked Yes.");
  } else {
    Logger.log("The user clicked No or closed the dialog.");
  }

  // 3. A prompt to get input from the user.
  const promptResponse = ui.prompt("Enter Your Name", "Please enter your
first name:", ui.ButtonSet.OK_CANCEL);

  // Check if the user clicked OK and entered text.
  if (promptResponse.getSelectedButton() == ui.Button.OK) {
    const userName = promptResponse.getResponseText();
    Logger.log("User's name is: " + userName);
  }
}
```

Exercise:

Create a function confirmAndWrite.

Show the user a prompt asking "What value should be written to A1?".

If the user clicks "OK", take the text they entered and write it into cell **A1**.

If they click "Cancel", log a message saying "Action was cancelled."

Quiz:

Which method creates a pop-up that just shows a message and an "OK" button?

a) .prompt()

b) .alert()

c) .show()

How do you get the text a user typed into a prompt dialog?

a) promptResponse.getText()

b) promptResponse.getResponseText()

c) promptResponse.getValue()

Answers & Explanations:

(b) .alert() is for displaying information. .prompt() is for gathering information.

(b) The prompt method returns an object, and you use the .getResponseText() method on that object to see what the user typed.

AI Prompts for Deeper Learning:

"How can I add a custom menu to my Google Sheet that will run my showDialogs script when I click it?"

"Show me an example of how to create a custom HTML sidebar in a Google Sheet using Apps Script."

Day 14: Review and Combine

Learning Objective: Combine concepts from the last two weeks to build a multi-step, useful script.

Content Explanation:
Let's build a small "report sender" tool. It will read a range of data from our sheet, ask the user for an email address, and then email that data to them. This combines SpreadsheetApp, getUi(), loops, and MailApp.

Code Example:
Setup: In cells A1:A3, list three tasks (e.g., "Task 1", "Task 2", "Task 3").
JavaScript

```
function sendTaskReport() {
  const ui = SpreadsheetApp.getUi();

  // 1. Get the recipient's email address.
  const promptResponse = ui.prompt("Enter recipient's email address:");
  if (promptResponse.getSelectedButton() != ui.Button.OK ||
promptResponse.getResponseText() == "") {
    ui.alert("Operation cancelled.");
    return; // Stop the function if the user cancels or enters nothing.
  }
  const recipient = promptResponse.getResponseText();

  // 2. Read the task data from the sheet.
  const sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Sheet1");
  const taskValues = sheet.getRange("A1:A3").getValues();

  // 3. Format the data into an email body.
  let emailBody = "Here is your task report:\n\n";
  for (let i = 0; i < taskValues.length; i++) {
    // The \n character creates a new line in the email.
    emailBody += "- " + taskValues[i][0] + "\n";
  }

  // 4. Send the email.
  const subject = "Your Daily Task Report";
  MailApp.sendEmail(recipient, subject, emailBody);

  ui.alert("Report sent successfully to " + recipient);
}
```

Exercise:
Create a "Data Logger" script.

Ask the user for a "Log Entry" using a prompt.

Find the *first empty row* in column A of your sheet. (Hint: A common way is to get all values, loop until you find an empty cell "", and use that row number).

Write the user's log entry into that empty row in column A.

Show an alert confirming that the entry was logged.

Quiz:

What does the return; command do inside the if statement in the example?

a) It returns a value to the user.

b) It stops the function from executing any further.

c) It restarts the function.

What is the purpose of the "\n" character in the emailBody string?

a) To add a tab space.

b) To add a new line.

c) To add a hyphen.

Answers & Explanations:

(b) An empty return; is a useful way to exit a function early if a condition isn't met.

(b) \n is the "newline character," used to format text in emails, logs, and other text-based content.

AI Prompts for Deeper Learning:

"Rewrite the for loop in the sendTaskReport function using the Array.map() and Array.join() methods to make the code more concise."

"How could I modify the 'Data Logger' exercise to also add a timestamp in column B next to the log entry?"

Week 3: Advanced Workspace Services

Day 15: Introduction to Triggers

Learning Objective: Learn how to make your functions run automatically in response to events, such as opening a spreadsheet or editing a cell.

Content Explanation:
A trigger connects a script function to a specific event. This is the key to true automation. You no longer need to click "Run" yourself. There are two types:

Simple Triggers: Special function names like onOpen(e) and onEdit(e) that run automatically when the spreadsheet is opened or a cell is edited, respectively. They have some limitations.

Installable Triggers: Triggers you set up manually through the editor. They are more powerful and flexible.

Code Example: Simple onOpen Trigger
This script adds a custom menu to the spreadsheet UI every time it's opened.
JavaScript

```
/**
 * Runs when the spreadsheet is opened.
 * @param {Object} e The event parameter.
 */
function onOpen(e) {
  SpreadsheetApp.getUi()
      .createMenu('My Custom Menu')
      .addItem('Show Alert', 'showAlertFunction')
      .addToUi();
}

function showAlertFunction() {
  SpreadsheetApp.getUi().alert('You clicked the custom menu item!');
}
```

function onOpen(e): This specific function name is a simple trigger. Apps Script runs it automatically.

.createMenu(...): Creates a new top-level menu.

.addItem(...): Adds an item to the menu. The first argument is the visible text, and the second is the name of the function to run when clicked (as a string).

.addToUi(): Finalizes the menu and adds it to the spreadsheet.
To Test: Save the script and reload your spreadsheet. The new menu should appear.

Exercise:
Create a simple onEdit(e) trigger.

Write a function named onEdit(e).

Inside, use Logger.log() to print "A cell was edited!".

Save the script. Go to your sheet and type anything into any cell.

Go back to the editor and check the "Executions" list on the left to see if your function ran and what it logged.

Quiz:

What is the correct name for a simple trigger function that runs when a user changes a cell's value?

a) whenEdited(e)

b) onEdit(e)

c) cellChange(e)

In the onOpen(e) example, what does 'showAlertFunction' represent?

a) The text that appears in the menu.

b) The name of the function to execute, written as a string.

c) A variable containing the function.

Answers & Explanations:

(b) onEdit(e) is the reserved name for the simple edit trigger.

(b) When adding menu items, you must provide the name of the function to be called as a text string.

AI Prompts for Deeper Learning:

"What is the event object e that gets passed to onOpen(e) and onEdit(e)? Show me how to use e in an onEdit trigger to find out the address and new value of the edited cell."

"How do I set up a time-driven (installable) trigger to run a function every morning at 9 AM? Walk me through the steps in the Apps Script editor."

Day 16: Working with Google Docs

Learning Objective: Learn how to create and manipulate Google Docs using the DocumentApp service.

Content Explanation:
Just like SpreadsheetApp controls Sheets, DocumentApp is your tool for Google Docs. You can create new documents, open existing ones, and programmatically add content like paragraphs, lists, and tables. This is great for generating reports or standard letters.

Code Example:
This script creates a new Google Doc and adds a title and a paragraph to it.
JavaScript

```
function createGoogleDoc() {
  // 1. Create a new document with a title.
  const doc = DocumentApp.create("My First Scripted Document");

  // 2. Get the body of the document to add content to it.
  const body = doc.getBody();

  // 3. Append a paragraph.
  body.appendParagraph("This is the first paragraph, added by Google Apps Script.");

  // 4. Add a heading.
  body.appendParagraph("A Section Heading").setHeading(DocumentApp.ParagraphHeading.HEADING1);

  // 5. Add a list item.
  body.appendListItem("First list item.");
  body.appendListItem("Second list item.");

  // 6. Log the URL of the new document.
  Logger.log("Document created! URL: " + doc.getUrl());
}
```

DocumentApp.create(...): Creates a brand new Google Doc file in your Google Drive root folder.

doc.getBody(): Gets the main body section of the document where you can add text.

body.appendParagraph(...): Adds a new paragraph with the specified text.

setHeading(...): Styles a paragraph as a heading.

Exercise:

Create a function generateMeetingNotes.

The script should create a new Google Doc titled "Meeting Notes - [Today's Date]". (Hint: Use new Date() and format it).

Add a Heading 1 that says "Attendees".

Add a bulleted list with the names of three fictional attendees.

Quiz:

What is the main service for interacting with Google Docs?

a) DocApp

b) WriterApp

c) DocumentApp

Which method is used to add a new text paragraph to a document's body?

a) .addParagraph()

b) .insertText()

c) .appendParagraph()

Answers & Explanations:

(c) The service is named DocumentApp.

(c) .appendParagraph() is the standard method for adding a new paragraph to the end of the document body.

AI Prompts for Deeper Learning:

"How can I open an *existing* Google Doc by its ID using Google Apps Script and add a new paragraph to the end of it?"

"Show me the code to find and replace all instances of the text '{{client_name}}' with 'Acme Corp' in a Google Doc template."

Day 17: Working with Google Drive

Learning Objective: Use DriveApp to interact with files and folders in your Google Drive, such as finding files or creating folders.

Content Explanation:
DriveApp is the service that lets you manage your Google Drive. It acts like a file manager, allowing you to create, find, move, and delete files and folders. It's the glue that can connect your different Apps Script projects.

Code Example:
This script lists the names of the first 5 files in your Google Drive's root folder.
JavaScript

```
function listDriveFiles() {
  // 1. Get all files in the root folder of your Drive.
  const files = DriveApp.getFiles();

  let count = 0;
  // 2. The 'files' object is an iterator, so we use a 'while' loop.
  while (files.hasNext() && count < 5) {
    const file = files.next();
    Logger.log("File Name: " + file.getName() + ", URL: " + file.getUrl());
    count++;
  }
}

function createFolder() {
  // Create a new folder named "Script-Generated Reports".
  const folder = DriveApp.createFolder("Script-Generated Reports");
  Logger.log("Folder created: " + folder.getName());
}
```

DriveApp.getFiles(): Returns an "iterator," which is an object you can loop through to get each file.

files.hasNext(): Checks if there are more files to process in the iterator.

files.next(): Gets the next file from the iterator.

file.getName(): Returns the name of the file.

DriveApp.createFolder(...): Creates a new folder.

Exercise:

Create a function createFileInFolder.

First, get a folder by its name. You can use DriveApp.getFoldersByName("Script-Generated Reports").next();. (This assumes the folder from the example exists).

Then, inside that folder object, use the .createFile() method to create a new, empty text file named log.txt with some initial content.

Quiz:

What is the main service for interacting with Google Drive?

a) FolderApp

b) DriveApp

c) FilesApp

The DriveApp.getFiles() method returns a collection of files. How do you access the individual files from it?

a) By using an index like files[0].

b) By using a while (files.hasNext()) loop and the files.next() method.

c) By using a for loop from 0 to files.length.

Answers & Explanations:

(b) The service is DriveApp.

(b) Many DriveApp methods return iterators, not arrays, so the hasNext()/next() pattern is required.

AI Prompts for Deeper Learning:

"How can I find a specific file in my Google Drive by its name, for example, 'My Budget 2025.xlsx'?"

"Show me the Google Apps Script code to move an existing file into the 'Script-Generated Reports' folder we created."

Day 18: Advanced Email - GmailApp

Learning Objective: Explore the more powerful GmailApp service to read your inbox, find specific emails, and send more complex emails.

Content Explanation:
While MailApp is great for simple sending, GmailApp gives you full control over your Gmail account. You can search for threads, get message details, add labels, and more. It's the difference between a mail-sending tool and a full email automation service.

Code Example:
This script finds the first 5 unread emails in your inbox and logs their subjects.
JavaScript

```
function getUnreadEmails() {
  // 1. Search for unread emails in the inbox.
  const threads = GmailApp.search("is:unread in:inbox");

  // 2. Loop through the first 5 threads found.
  for (let i = 0; i < threads.length && i < 5; i++) {
    const thread = threads[i];
    const firstMessage = thread.getMessages()[0]; // Get the first message in the thread.

    Logger.log("Subject: " + firstMessage.getSubject());
    Logger.log("From: " + firstMessage.getFrom());
  }
}

function sendHtmlEmail() {
  const recipient = Session.getActiveUser().getEmail();
  const subject = "HTML Email Test";
  // The 'htmlBody' option allows you to use HTML tags.
  const body = "This is an email with <b>bold text</b> and an <i>italicized word</i>.";

  GmailApp.sendEmail(recipient, subject, "", { htmlBody: body });
}
```

GmailApp.search(...): Uses standard Gmail search syntax to find threads.

thread.getMessages(): Gets all messages within a single email thread.

{ htmlBody: body }: This is an "options" object. It allows sendEmail to

accept advanced parameters like an HTML body.

Exercise:

Create a function findAndStarEmails.

Use GmailApp.search() to find any emails sent from "notifications@google.com".

Loop through the results and use the message.star() method to star each message.

Quiz:

Which GmailApp method lets you find emails using syntax like "from:example@email.com"?

a) .find()

b) .get()

c) .search()

In the sendHtmlEmail example, why is the third argument (the plain text body) an empty string ""?

a) It's a mistake; it should contain the body text.

b) It's required, but we are providing the body in the htmlBody option instead.

c) It tells Gmail to not send the email.

Answers & Explanations:

(c) The .search() method directly uses the powerful search query language built into Gmail.

(b) When you use the advanced options object, you still need to provide a value for the plain text body argument, even if it's empty.

AI Prompts for Deeper Learning:

"How can I use Google Apps Script to get the attachments from an email and save them to a specific folder in Google Drive?"

"Show me how to create a draft email in Gmail using GmailApp but not send it."

Day 19: Working with Google Calendar

Learning Objective: Use CalendarApp to read events from your calendar and create new events.

Content Explanation:
CalendarApp lets you automate your schedule. You can create events for meetings, set reminders, invite guests, and check your availability, all from a script. This is fantastic for building scheduling tools or automatically logging events from other sources.

Code Example:
JavaScript
```
function listNextTenEvents() {
  // 1. Get the default calendar for the user.
  const calendar = CalendarApp.getDefaultCalendar();

  // 2. Define the time range to search.
  const now = new Date();
  const oneWeekFromNow = new Date(now.getTime() + (7 * 24 * 60 * 60 *
1000));

  // 3. Get the events in that range.
  const events = calendar.getEvents(now, oneWeekFromNow);

  // 4. Log the title and start time of the first 10 events found.
  for (let i = 0; i < events.length && i < 10; i++) {
    Logger.log("Event: " + events[i].getTitle() + " at " +
events[i].getStartTime());
  }
}

function createCalendarEvent() {
  const calendar = CalendarApp.getDefaultCalendar();

  const title = "Project Sync-Up";
  const startTime = new Date("September 15, 2025 10:00:00 EST");
  const endTime = new Date("September 15, 2025 10:30:00 EST");

  const newEvent = calendar.createEvent(title, startTime, endTime, {
    location: 'Virtual Meeting',
    guests: 'friend@example.com'
```

```
  });

  Logger.log("Event created with ID: " + newEvent.getId());
}
```

Exercise:

Create a function scheduleLunch.

It should create a 1-hour event on your default calendar for tomorrow at noon.

The title should be "Lunch Break".

Quiz:

What method is used to get the user's primary calendar?

a) CalendarApp.getPrimaryCalendar()

b) CalendarApp.getMainCalendar()

c) CalendarApp.getDefaultCalendar()

What are the three mandatory arguments for createEvent()?

a) Title, Start Time, and End Time.

b) Title, Location, and Guests.

c) Title, Date, and Duration.

Answers & Explanations:

(c) The method to remember is getDefaultCalendar().

(a) You must provide a title and a start and end Date object to create a basic event. Other options like location and guests are optional.

AI Prompts for Deeper Learning:

"How can I find a specific calendar by its name (e.g., 'Team Holidays') instead of using the default one?"

"Write a Google Apps Script function that checks if I have any events scheduled tomorrow between 2 PM and 4 PM."

Day 20: Working with Google Forms

Learning Objective: Use FormApp to create a Google Form

programmatically.

Content Explanation:
The FormApp service allows you to create and modify Google Forms with code. You can build entire surveys, quizzes, or feedback forms, add different types of questions, and set options like making questions required.

Code Example:
This script creates a simple contact information form.
JavaScript

```
function createContactForm() {
  // 1. Create a new form with a title and description.
  const form = FormApp.create('New Contact Form')
      .setTitle('Contact Information')
      .setDescription('Please fill out your details.');

  // 2. Add a text item for the name.
  form.addTextItem()
      .setTitle('What is your full name?')
      .setRequired(true);

  // 3. Add a multiple-choice question for contact preference.
  form.addMultipleChoiceItem()
      .setTitle('Preferred method of contact?')
      .setChoiceValues(['Email', 'Phone', 'Text']);

  // 4. Add a paragraph text item for comments.
  form.addParagraphTextItem()
      .setTitle('Any additional comments?');

  Logger.log('Form created! Published URL: ' + form.getPublishedUrl());
  Logger.log('Editor URL: ' + form.getEditUrl());
}
```

Exercise:

Create a function createSurvey.

It should create a new form titled "Customer Satisfaction Survey".

Add a "checkbox" question (.addCheckboxItem()) asking "Which products have you used?". Set the choices to "Product A", "Product B", and "Product

C".

Add a "scale" question (.addScaleItem()) asking to "Rate your overall satisfaction" from 1 to 5.

Quiz:

What is the service used for creating Google Forms?

a) SurveyApp

b) FormApp

c) QuestionApp

In the example, what does .setRequired(true) do?

a) It makes the entire form required.

b) It makes the "What is your full name?" question mandatory.

c) It requires the user to log in.

Answers & Explanations:

(b) The service is FormApp.

(b) Methods like .setRequired() apply to the specific item they are called on, in this case, the text item for the name.

AI Prompts for Deeper Learning:

"How can I programmatically get all the responses that have been submitted to an existing Google Form and log them in a Google Sheet?"

"Write a script that creates a Google Form and sets it up as a quiz, including setting a correct answer and point value for a multiple-choice question."

Day 21: Storing Simple Data with Properties Service

Learning Objective: Learn how to store and retrieve small pieces of information (like settings or a user's name) without using a spreadsheet, using PropertiesService.

Content Explanation:
Sometimes you need to save a small piece of data, like a user's preference or the ID of a file you just created. Using a whole spreadsheet for this is overkill. PropertiesService is a built-in key-value store, perfect for saving simple script properties.

Code Example:

```
JavaScript
function storeAndRetrieveProperties() {
  // Get the store for this specific script.
  const scriptProperties = PropertiesService.getScriptProperties();

  // 1. Storing a property.
  scriptProperties.setProperty('API_KEY', 'xyz123abc');
  scriptProperties.setProperty('LAST_RUN_USER',
Session.getActiveUser().getEmail());

  Logger.log("Properties have been set.");

  // 2. Retrieving a property.
  const savedApiKey = scriptProperties.getProperty('API_KEY');
  Logger.log("The saved API key is: " + savedApiKey);

  // 3. Deleting a property.
  // scriptProperties.deleteProperty('API_KEY');
  // Logger.log("Property deleted.");
}
```

PropertiesService.getScriptProperties(): Gets a property store that is accessible by anyone who can edit the script project.

.setProperty('KEY', 'VALUE'): Saves a value associated with a key. Both must be strings.

.getProperty('KEY'): Retrieves the value for the given key.

Exercise:

Create a function updateRunCount.

The first time it runs, it should get a property called runCount. It will be null (doesn't exist). So, it should set the runCount property to '1'.

The next time it runs, it should get the property, convert it to a number, add 1, and save it back as a string.

Log the current run count each time the function executes.

Quiz:

PropertiesService stores data in what format?

a) Rows and columns.

b) Key-value pairs.

c) Arrays of objects.

What data types must the key and value be when using .setProperty()?

a) They can be any data type (number, boolean, etc.).

b) The key must be a string, but the value can be anything.

c) Both the key and the value must be strings.

Answers & Explanations:

(b) It works like a dictionary or a simple database where each piece of data (value) is looked up by its unique name (key).

(c) This is a limitation you must remember. If you have a number, you need to convert it to a string before saving and parse it back to a number after retrieving it.

AI Prompts for Deeper Learning:

"What is the difference between PropertiesService.getScriptProperties(), getUserProperties(), and getDocumentProperties()? Give a scenario where you would use each one."

"Show me how to store a JavaScript object in PropertiesService by using JSON.stringify() before saving and JSON.parse() after retrieving."

Week 4: Building Small Applications

Day 22: Error Handling with try...catch

Learning Objective: Learn how to handle potential errors gracefully in your script using a try...catch block so that it doesn't crash unexpectedly.

Content Explanation:
Sometimes, things go wrong. A file might not exist, a service might be down, or a user might enter bad data. A try...catch block lets you "try" to run some code that might fail. If it does, the script doesn't stop; instead, the catch block runs, where you can log the error or notify the user.

Code Example:
JavaScript

```
function robustFunction() {
  try {
    // --- TRY BLOCK ---
    // Let's try to access a sheet that doesn't exist. This will cause an error.
    const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("NonExistentSheet");
    const data = sheet.getRange("A1").getValue(); // This line will never be reached.
    Logger.log("Success! Data is: " + data);

  } catch (error) {
    // --- CATCH BLOCK ---
    // If an error happens anywhere in the 'try' block, this code runs.
    Logger.log("An error occurred!");
    Logger.log("Error details: " + error.message);
    Logger.log("Stack trace: " + error.stack);

    // You could also send an email or show an alert to the user here.
    SpreadsheetApp.getUi().alert("Sorry, the script could not find the required sheet.");
  }

  Logger.log("Script execution finished."); // This line will always run.
}
```

try { ... }: The code that might fail goes here.

catch (error) { ... }: This block only executes if an error occurs in the try block. The error variable contains details about what went wrong.

Exercise:

Write a function divideNumbers.

It should prompt the user for two numbers.

Inside a try block, convert the user's text input to numbers and divide the first by the second. Log the result.

The catch block should handle cases where the input isn't a number or the user tries to divide by zero. It should log a friendly error message.

Quiz:

If the code inside a try block runs successfully without any errors, what happens to the catch block?

a) It runs after the try block.

b) It is skipped.

c) It causes the script to stop.

What is the purpose of the error variable in catch (error)?

a) It's a string that always says "Error".

b) It's an object containing information about the specific error that occurred.

c) It's a boolean that is true if an error happened.

Answers & Explanations:

(b) The catch block is exclusively for handling errors; if there are none, it's ignored.

(b) The error object is crucial for debugging, as its .message and .stack properties can tell you exactly where and why the code failed.

AI Prompts for Deeper Learning:

"What is a finally block in a try...catch...finally statement? Provide a Google Apps Script example where it would be useful."

"How can I 'throw' my own custom error in Google Apps Script? For

example, if a user enters a negative number where it should be positive."

Day 23: Reading Data from an API - UrlFetchApp (Part 1)

Learning Objective: Understand what an API is and use UrlFetchApp to retrieve simple data from a public API.

Content Explanation:
An API (Application Programming Interface) is a way for different computer programs to talk to each other. Many websites provide APIs that let you request data in a structured format called JSON. Google Apps Script can act as a client, requesting data from these APIs using the UrlFetchApp service. This opens up a world of possibilities, allowing you to pull in live data like stock prices, weather, or project management tasks.

Code Example:
We will use a free, simple API called JSONPlaceholder, which provides fake data for testing. This script will fetch a single "to-do" item.
JavaScript

```
function fetchApiData() {
  try {
    // 1. The URL of the API endpoint.
    const apiUrl = "https://jsonplaceholder.typicode.com/todos/1";

    // 2. Fetch the data from the URL.
    const response = UrlFetchApp.fetch(apiUrl);

    // 3. Get the text content of the response.
    const responseText = response.getContentText();

    // 4. Parse the JSON string into a JavaScript object.
    const data = JSON.parse(responseText);

    // 5. Log specific properties of the object.
    Logger.log("Fetched To-Do Item:");
    Logger.log("User ID: " + data.userId);
    Logger.log("Title: " + data.title);
    Logger.log("Completed: " + data.completed);

  } catch (e) {
    Logger.log("Failed to fetch data: " + e.message);
```

```
  }
}
```

UrlFetchApp.fetch(apiUrl): Makes an HTTP GET request to the specified URL and returns an HTTPResponse object.

response.getContentText(): Extracts the raw text content (the JSON string) from the response.

JSON.parse(responseText): This is a crucial step. It converts the JSON text into a usable JavaScript object, so you can access its properties with dot notation (like data.title).

Exercise:

Use the JSONPlaceholder API to fetch data about a specific user. The URL is https://jsonplaceholder.typicode.com/users/1.

Create a function fetchUserData.

Fetch the data from that URL.

Parse the JSON response.

Log the user's **name**, **email**, and **city** (Hint: the city is inside the address object).

Quiz:

What is the primary service for making requests to external websites/APIs?

a) HttpApp

b) ApiApp

c) UrlFetchApp

What does JSON.parse() do?

a) It creates a beautiful visualization of the data.

b) It converts a JSON text string into a JavaScript object.

c) It sends the JSON data back to the server.

Answers & Explanations:

(c) UrlFetchApp is the service for all external web requests.

(b) You almost always need to parse the JSON text you receive from an API before you can work with the data in your code.

AI Prompts for Deeper Learning:

"What is JSON? Explain its structure (keys, values, objects, arrays) in simple terms."

"Using the API at https://api.quotable.io/random, write a Google Apps Script function that fetches a random quote and writes it into cell A1 of a Google Sheet."

Day 24: Writing API Data to a Sheet - UrlFetchApp (Part 2)

Learning Objective: Fetch a list of items from an API and write the structured data into a Google Sheet using a loop.

Content Explanation:
A common use case for APIs is to pull in a list of data (e.g., a list of products, users, or posts) and display it in a Google Sheet. This involves fetching the data, looping through the resulting array of objects, and building a 2D array that can be written to the sheet using .setValues().

Code Example:
This script fetches 10 "to-do" items from the JSONPlaceholder API and writes their title and completion status to a sheet.
JavaScript

```
function writeApiDataToSheet() {
  // 1. Fetch the list of to-do items.
  const response =
UrlFetchApp.fetch("https://jsonplaceholder.typicode.com/todos?_limit=10");
  const data = JSON.parse(response.getContentText()); // This is now an
array of objects.

  // 2. Prepare a 2D array for the sheet. Start with headers.
  const sheetData = [["Title", "Completed Status"]];

  // 3. Loop through the array of to-do objects.
  for (let i = 0; i < data.length; i++) {
    const todo = data[i];
    // Create a row for each item and push it to our sheetData array.
    sheetData.push([todo.title, todo.completed]);
  }

  // 4. Write the data to the spreadsheet.
```

```
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  // Clear old data first.
  sheet.clear();
  // Write the new data. The range will be automatically sized.
  sheet.getRange(1, 1, sheetData.length,
sheetData[0].length).setValues(sheetData);
}
```

const data = JSON.parse(...): data is now an array, where each element is an object like { "userId": 1, "id": 1, "title": "...", "completed": false }.

sheetData.push([todo.title, todo.completed]): We extract the specific pieces of information we want and format them as a row (an inner array) to be added to our main sheetData array.

Exercise:

Create a function writeUserListToSheet.

Fetch the list of 10 users from https://jsonplaceholder.typicode.com/users.

Write the **Name**, **Email**, and **Website** of each user into columns A, B, and C of your sheet. Include headers.

Quiz:

In the example, why do we create an empty array sheetData and use .push() in a loop?

a) To format the raw API data into the 2D array structure that .setValues() requires.

b) To slow down the script so the API doesn't block us.

c) To count how many items are in the API response.

What does sheet.getRange(1, 1, sheetData.length, sheetData[0].length) do?

a) It gets a single cell at row 1, column 1.

b) It creates a new sheet.

c) It dynamically selects a range starting at A1 that is exactly the size of our sheetData array (rows x columns).

Answers & Explanations:

(a) This is a standard pattern: fetch data, transform it into the required 2D array format, and then write it to the sheet in a single,

efficient operation.

(c) This four-argument version of .getRange() (startRow, startCol, numRows, numCols) is perfect for writing data when you don't know the exact size beforehand.

AI Prompts for Deeper Learning:

"What are API keys and how are they used for authentication? Show me an example of how to include an API key in the 'headers' option of a UrlFetchApp.fetch() call."

"Explain what an HTTP POST request is. Write a Google Apps Script example using UrlFetchApp to send (POST) some JSON data to https://jsonplaceholder.typicode.com/posts."

Day 25: Building a Custom Sidebar

Learning Objective: Create a custom sidebar with HTML content that can interact with your Apps Script functions.

Content Explanation:
For a richer user experience, you can create a custom sidebar using standard HTML, CSS, and JavaScript. This allows you to build much more complex interfaces than simple alerts and prompts. The HTML file lives inside your Apps Script project.

Code Example:
Step 1: Create the Apps Script code (Code.gs)
JavaScript

```
function showSidebar() {
  const html = HtmlService.createHtmlOutputFromFile('Sidebar')
      .setTitle('My Custom Sidebar');
  SpreadsheetApp.getUi().showSidebar(html);
}

function doSomething(name) {
  Logger.log(name + " clicked the button!");
  SpreadsheetApp.getActive().toast(name + " says hello!");
}
```

Step 2: Create the HTML file (Sidebar.html)

In the editor, click + > HTML and name the file Sidebar.

Put this code in the file:

```
HTML
<!DOCTYPE html>
<html>
  <head>
    <base target="_top">
  </head>
  <body>
    <h3>Sidebar Content</h3>
    <p>Enter your name and click the button.</p>
    <input type="text" id="nameInput" placeholder="Your name">
    <button onclick="callAppsScript()">Click Me</button>

    <script>
      function callAppsScript() {
        var name = document.getElementById('nameInput').value;

google.script.run.withSuccessHandler(onSuccess).doSomething(name)
;
      }

      function onSuccess() {
        console.log("Function call was successful.");
        // You could close the sidebar or clear the input here.
      }
    </script>
  </body>
</html>
```

HtmlService.createHtmlOutputFromFile('Sidebar'): This loads the HTML file.

google.script.run.doSomething(name): This is the magic! It's client-side JavaScript (in the HTML) that calls your server-side Apps Script function (doSomething). The argument name is passed from the browser to your script.

withSuccessHandler(onSuccess): An optional handler that runs a client-side function if the Apps Script call completes successfully.

Exercise:

Add another button to the Sidebar.html file with the text "Write to A1".

Create a new Apps Script function called writeDataToA1(someText).

Make the new button call this function, passing the text from the nameInput field.

The writeDataToA1 function should write the received text into cell A1 of the active sheet.

Quiz:

In the HTML file, what is the purpose of google.script.run?

a) To run the HTML file.

b) To create a bridge that allows client-side JavaScript to call server-side Apps Script functions.

c) To check for script errors.

How do you pass a value from an input field in the sidebar to your Apps Script function?

a) The script automatically knows the values of all input fields.

b) You can't pass values; you can only trigger functions.

c) You pass it as an argument in the google.script.run call, like doSomething(name).

Answers & Explanations:

(b) This is the core object for communication from the client (sidebar) to the server (your .gs code).

(c) Any arguments you provide in the google.script.run call will be passed directly to your server-side Apps Script function.

AI Prompts for Deeper Learning:

"How can an Apps Script function return data *back* to the sidebar's JavaScript? Show an example where a .gs function returns a string, and the withSuccessHandler displays it in the sidebar."

"How can I add some basic CSS styling to my Sidebar.html file to make the button look better?"

Day 26: Project 1 - Email Merger

Learning Objective: Build a complete, practical tool that reads data from a sheet and sends personalized emails.

Content Explanation:

This project combines many skills: reading sheet data, loops, string manipulation, and sending emails. We will create a tool that takes a list of names and email addresses from a sheet, along with a message template, and sends a personalized email to each person.

Code Example:
Setup: In your sheet:

Column A (Header Name): Alice, Bob

Column B (Header Email): Your email address for both rows (so you can test).

Cell **E1**: Hi {{name}}, this is a test email. (This is our template).

JavaScript

```
function runEmailMerger() {
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();

  // 1. Get data from the sheet. Assumes headers are in row 1.
  // A2:B is A-Number notation for "start at A2, go to the end of column B".
  const dataRange = sheet.getRange("A2:B" + sheet.getLastRow());
  const contactData = dataRange.getValues();

  // 2. Get the email template.
  const template = sheet.getRange("E1").getValue();
  const subject = "A Personalized Message for You";

  // 3. Loop through each contact.
  for (const row of contactData) {
    const name = row[0];
    const email = row[1];

    // Skip rows that don't have an email address.
    if (!email) {
      continue;
    }

    // 4. Personalize the message.
    const message = template.replace("{{name}}", name);

    // 5. Send the email.
    try {
      GmailApp.sendEmail(email, subject, message);
      // Optional: Mark as sent in the sheet.
      // sheet.getRange(contactData.indexOf(row) + 2, 3).setValue("Sent");
```

```
    } catch (e) {
      Logger.log("Failed to send to " + email + ". Error: " + e.message);
    }
  }
  SpreadsheetApp.getActive().toast("Email merge complete!");
}
```

Exercise:

Add a third column to your sheet for Product (e.g., "Magic Widget", "Super Gizmo").

Update the template in E1 to Hi {{name}}, thank you for your interest in the {{product}}.

Modify the script to read the third column and replace both {{name}} and {{product}} placeholders in the email body.

Quiz:

What does the String.replace("{{name}}", name) method do?

a) It renames the name variable to {{name}}.

b) It finds the first occurrence of the text {{name}} in the template string and replaces it with the value of the name variable.

c) It checks if the template contains the name.

What is the purpose of the continue; keyword in the loop?

a) It stops the entire script.

b) It tells the loop to immediately stop the current iteration and move to the next row.

c) It pauses the script for one second.

Answers & Explanations:

(b) This is a fundamental technique for creating dynamic text from templates.

(b) continue is a useful control flow statement for skipping records that don't meet a certain criteria (like having a valid email).

AI Prompts for Deeper Learning:

"How could I add an onOpen trigger to create a custom menu item that runs this runEmailMerger function?"

"The current script might exceed Google's daily email quota if the list is long. How could I modify it to only process 20 rows at a time and use PropertiesService to remember where it left off for the next run?"

Day 27: Project 2 - Form to Calendar

Learning Objective: Build an automation that takes submissions from a Google Form and automatically creates an event on a Google Calendar.

Content Explanation:
This project connects three services: Forms, Sheets (as the response destination), and Calendar. We'll create a form for people to request an appointment. When they submit it, an installable trigger will run a function that reads their response from the linked spreadsheet and puts the event on our calendar.

Setup:

Create a Form: Create a Google Form with three questions:

Event Title (Short answer)

Event Date (Date question)

Your Email (Short answer)

Link to Sheet: In the "Responses" tab of the form, click the spreadsheet icon to create a new spreadsheet to store responses.

Open Script Editor: From that new spreadsheet, go to Extensions > Apps Script.

Code Example:
JavaScript

```
function onFormSubmit(e) {
  // NOTE: This function MUST be installed as a trigger. It will not run automatically.
  try {
    // 1. Get the data from the event object 'e'.
    const response = e.namedValues;
    const title = response['Event Title'][0];
    const dateString = response['Event Date'][0];
    const guestEmail = response['Your Email'][0];

    // 2. Create a full Date object for the event. Let's make it a 1-hour event
```

```
at 9 AM.
    // The form only gives us month/day/year, so we need to parse it and add
the time.
    const eventDate = new Date(dateString);
    const startTime = new Date(eventDate.getFullYear(),
eventDate.getMonth(), eventDate.getDate(), 9, 0, 0);
    const endTime = new Date(eventDate.getFullYear(),
eventDate.getMonth(), eventDate.getDate(), 10, 0, 0);

    // 3. Get the calendar and create the event.
    const calendar = CalendarApp.getDefaultCalendar();
    calendar.createEvent(title, startTime, endTime, {
      guests: guestEmail,
      sendInvites: true // This will email an invitation to the guest.
    });

    Logger.log("Event created successfully for " + title);

  } catch (err) {
    Logger.log("Failed to create event. Error: " + err.message);
  }
}
```

How to Install the Trigger:

In the Apps Script editor, click the "Triggers" icon (looks like a clock) on the left.

Click "+ Add Trigger".

Choose the onFormSubmit function to run.

Select event source: From spreadsheet.

Select event type: On form submit.

Click "Save". You will need to authorize the script again.

Exercise:

Modify the Form and the script to include an Event Description field (a "Paragraph" question in the form).

When the event is created, pass this description into the options object of

the createEvent method.

Quiz:

Why do we need to set up an installable trigger for this script?

a) Simple triggers like onEdit don't work for form submissions.

b) The onFormSubmit function needs special permissions that only installable triggers can provide.

c) Both of the above are correct.

What is e.namedValues in the onFormSubmit(e) function?

a) An array of the submitted values in order.

b) An object where the keys are the question titles from your form.

c) The email address of the person who submitted the form.

Answers & Explanations:

(c) Responding to a form submission is a protected action that requires explicit, installable trigger setup and authorization.

(b) The e.namedValues object is extremely convenient because it lets you access response data by the name of the question, making your code much easier to read than if you used array indices.

AI Prompts for Deeper Learning:

"How could I check the calendar for conflicts before creating the event? Show me the code to see if the startTime to endTime slot is already busy."

"Modify the script so that after the event is created, it writes 'Event Created' into a new column in the response spreadsheet for that row."

Day 28: What's Next and How to Keep Learning

Learning Objective: Review what you've learned and understand the best resources for continuing your Apps Script journey.

Content Explanation:
Congratulations! You've completed the 28-day guide and have gone from a complete beginner to building genuinely useful automations. You now have a solid foundation in the most important Google Workspace services.

What You Have Learned:

The Basics: Variables, loops, if statements, functions.

Core Services: Controlling Sheets, Docs, Drive, Gmail, Calendar, and Forms.

Automation: Using simple and installable triggers to run code automatically.

User Interface: Creating menus, alerts, and custom HTML sidebars.

External Data: Fetching and using data from external APIs.

Best Practices: Handling errors and storing properties.

How to Keep Learning:

Build Your Own Projects: The best way to learn is by doing. Think of a repetitive task you do at work or in your personal life. Can you automate it? Start small and build on it.

Read the Official Documentation: Google's official documentation is your best friend. It's comprehensive and has examples for every method. When you wonder, "Can I do X?", the documentation is the first place to look.

Explore Online Communities: Websites like Stack Overflow (tag: google-apps-script) and the Google Workspace Developer Community are full of experts and people asking and answering questions.

Follow Experts and Blogs: Many developers share tutorials and advanced techniques. Search for "Google Apps Script tutorials" to find blogs and YouTube channels. A great resource is "Totally Unscripted."

Use AI as a Learning Partner: You've seen the prompts throughout this guide. Continue using AI! When you have an idea, ask it: "Write a Google Apps Script function that does [your idea]." Analyze the code it gives you, understand it, and then adapt it to your needs.

Final Project Idea:
Think of a project that combines at least three different services. For example:

An "Invoice Generator": Reads client data from a Sheet, uses a Doc as a template to create a PDF invoice, saves the PDF to a specific client folder in Drive, and then emails the invoice to the client using Gmail.

Parting Words:
You have a powerful new skill. Automation is not just about saving time; it's about eliminating errors, creating new capabilities, and freeing up your mind to focus on more important, creative work. Keep experimenting, keep

building, and have fun!