Welcome to Your Ultimate Guide to Google Apps Script

Ever find yourself lost in a sea of repetitive tasks within Google Sheets, Docs, or Gmail? Have you ever thought, "There has to be a better way to do this"? There is, and you've found it.

Enter Google Apps Script, the powerful, cloud-based scripting platform built right into Google Workspace. Based on JavaScript, it's the key to unlocking a new level of productivity by allowing you to automate, integrate, and extend the Google applications you use every single day. From sending customized emails to generating reports and creating custom user interfaces, Apps Script transforms your static documents and spreadsheets into dynamic, intelligent applications.

Whether you're a complete beginner curious about what's possible, an intermediate user looking to solve a specific problem, or an educator crafting learning materials, this guide is designed for you. We've structured it around the 100 most common questions that developers encounter, creating a comprehensive resource for learners at all levels.

Inside, we will journey from the absolute fundamentals—like writing your first function—to advanced topics like connecting to external APIs and deploying your own web applications. Each question is paired with a clear, detailed explanation and a practical, ready-to-use code example. Think of it as a comprehensive FAQ and a hands-on cookbook rolled into one.

Dive in, find the answers you need, and start transforming the way you work with Google Workspace. Let's begin automating!

Getting Started & Fundamentals

This section covers the absolute basics, from what Apps Script is to fundamental coding concepts.

---

1. What is Google Apps Script?

Explanation: Google Apps Script is a **cloud-based scripting language** for light-weight application development in the Google Workspace platform. It's based on JavaScript, but it lives entirely in Google's cloud. You don't need to install anything. Its primary power is its ability to easily interact with and automate Google services like Sheets, Docs, Drive, Gmail, and Calendar.

Code Example: A "Hello, World!" in Apps Script is often done using the `Logger` to see output.

```
function helloWorld() {
  // The Logger is a built-in tool to print text, variables, or objects for debugging.
  // To view the logs, run the function and go to View > Logs.
  Logger.log('Hello, World!');
}
```

---

2. How do I create a Google Apps Script?

Explanation: You can create a script in two main ways:

1. **Standalone Script:** Go to `script.google.com` and create a new project. This is useful for scripts that work across multiple services or don't belong to a single file.
2. **Bound Script:** Open a Google Sheet, Doc, or Form, and go to **Extensions > Apps Script**. This creates a script that is "bound" to that specific file, making it easier to interact with it directly.

---

3. What's the difference between a standalone and a bound script?

Explanation:

- **Bound Scripts** are linked directly to a Google Sheet, Doc, Form, or Slide file. They have special privileges to act on that file without needing its ID. For example, `SpreadsheetApp.getActiveSpreadsheet()` only works in a script bound to a spreadsheet.
- **Standalone Scripts** are not attached to any specific file. They are general-purpose and can be used to create web apps or interact with various Google services, but you must explicitly reference files by their ID or URL.

---

4. How do I run a function in the Apps Script editor?

Explanation: In the script editor, you'll see a dropdown menu of all the functions in your script. Select the function you want to run from the list and click the ▸ **Run** button. The first time you run a script that accesses your data (like reading a Sheet), Google will prompt you to grant it the necessary permissions.

---

5. What are `onOpen()` and `onEdit()`?

Explanation: These are **simple triggers**. They are special, reserved function names that automatically run when a specific event occurs.

- `onOpen(e)`: Runs automatically whenever a user with edit access opens the spreadsheet, document, or form. It's most commonly used to add a **custom menu** to the UI.
- `onEdit(e)`: Runs automatically whenever a user changes the value of any cell in a spreadsheet.

Code Example:

```
/**
 * Runs when the spreadsheet is opened.
 * This is a simple trigger.
 */
function onOpen() {
  // Add a custom menu to the spreadsheet UI.
  SpreadsheetApp.getUi()
```

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

/**
 * A function that will be called from the custom menu.
 */
function showAlert() {
  SpreadsheetApp.getUi().alert('You clicked the custom menu item!');
}
```

---

6. What is the difference between `let`, `const`, and `var`?

Explanation: These are all used to declare variables, but they have different "scopes" and rules.

- `var`: The oldest way. It's function-scoped, meaning it's accessible anywhere within the function it's defined in. It can be re-declared and updated. (Generally, avoid using `var`).
- `let`: The modern standard. It's block-scoped, meaning it's only accessible within the curly braces (`{}`) it's defined in (like inside a `for` loop or `if` statement). It can be updated but not re-declared in the same scope.
- `const`: Used for constants. It's also block-scoped, but its value **cannot be changed** after it's assigned. This is great for values you know shouldn't be altered, like an API key or a fixed spreadsheet name.

Code Example:

```
function variableExamples() {
  const SPREADSHEET_ID = 'your_spreadsheet_id_here'; // Cannot be changed
  let loopCounter = 0; // Can be changed

  for (let i = 0; i < 5; i++) {
    loopCounter = i;
    Logger.log(loopCounter); // Works fine
  }

  // Logger.log(i); // This would cause an error because 'i' only exists inside the loop.
}
```

---

7. How do I comment my code?

Explanation: Comments are crucial for explaining what your code does. Apps Script ignores them.

- `//`: Use this for single-line comments.
- `/* ... */`: Use this for multi-line comments.
- `/** ... */`: This is a special multi-line comment style called **JSDoc**. It's the best practice because it enables autocomplete and provides descriptions of your functions and parameters in the editor.

Code Example:

```
/**
 * Gets the name of a sheet by its index (position).
 * @param {number} index The position of the sheet (0 for the first sheet).
 * @returns {string} The name of the sheet.
 */
function getSheetNameByIndex(index) {
  // This is a single-line comment explaining the next line.
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheets()[index];
  return sheet.getName();

  /*
    This is a multi-line comment.
    It can span several lines and is useful for longer explanations.
  */
}
```

---

8. What are the main Google Workspace Services in Apps Script?

Explanation: Apps Script provides built-in services to control Google products. You access them through global objects. The most common ones are:

- `SpreadsheetApp`: For Google Sheets.
- `DocumentApp`: For Google Docs.
- `DriveApp`: For Google Drive.
- `GmailApp`: For sending emails from your Gmail account.
- `CalendarApp`: For managing Google Calendar events.
- `FormApp`: For creating and managing Google Forms.
- `SlidesApp`: For Google Slides.

---

9. How do I get a specific sheet in a spreadsheet?

Explanation: You have several ways to grab a sheet object:

- `getActiveSheet()`: Gets the sheet the user is currently viewing.
- `getSheetByName(name)`: Gets a sheet by its exact name (the name you see in the tab at the bottom).
- `getSheets()`: Gets an array of all sheets in the spreadsheet, which you can then access by index (e.g., `getSheets()[0]` for the very first sheet).

Code Example:

```
function getSpecificSheet() {
  const ss = SpreadsheetApp.getActiveSpreadsheet();

  // Method 1: Get the sheet the user is currently looking at.
  const activeSheet = ss.getActiveSheet();
  Logger.log(`Active sheet is: ${activeSheet.getName()}`);

  // Method 2: Get a sheet by its name.
  const dataSheet = ss.getSheetByName('Sheet1');
  if (dataSheet) {
    Logger.log('Found the sheet named "Sheet1"!');
  }

  // Method 3: Get the very first sheet by its position.
  const firstSheet = ss.getSheets()[0];
  Logger.log(`The first sheet is: ${firstSheet.getName()}`);
}
```

---

10. What is the `Logger` and how do I use it?

Explanation: The `Logger` is your best friend for debugging. It's a simple tool that lets you print out the values of variables, messages, or objects to see what your script is doing at a certain point. After running a function that uses `Logger.log()`, you can view the output by going to **View > Logs** (or `Ctrl+Enter` / `Cmd+Enter`).

---

11. What is an "Execution Log"?

Explanation: The Execution Log is a record of all the times your functions have been run. You can find it under the "Executions" tab (a clock icon) on the left side of the editor. It shows you which function ran, when it started, how long it took, and whether it completed successfully or failed. If it failed, you can see the exact error message, which is critical for troubleshooting.

---

12. How do I handle errors?

Explanation: Use a `try...catch` block. You "try" to run code that might fail (like fetching a URL or accessing a sheet that might not exist). If an error occurs, the code inside the `catch` block is executed, preventing your script from crashing. This allows you to log the error gracefully or show a message to the user.

Code Example:

```
function mightFail() {
  try {
    // Try to get a sheet that doesn't exist. This will cause an error.
    const sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName('NonExistentSheet');
    sheet.getRange('A1').setValue('This will not run.');
  } catch (error) {
    // If an error happens in the 'try' block, this code runs.
    Logger.log(`An error occurred: ${error.message}`);
    // You could also show an alert to the user.
    SpreadsheetApp.getUi().alert('Oops! Could not find the required sheet.');
  }
}
```

---

13. How do I declare and use an array?

Explanation: An array is a list of values stored in a single variable. In Apps Script (and JavaScript), you declare an array using square brackets `[]`. Arrays are "zero-indexed," meaning the first item is at index `0`.

Code Example:

```
function arrayExample() {
  // An array of strings
  const teamMembers = ['Naza', 'Maureen', 'Stephanie', 'Lars'];

  // Accessing an item by its index
  Logger.log(teamMembers[0]); // Outputs: Naza

  // Finding the length of the array
  Logger.log(`There are ${teamMembers.length} team members.`);

  // Looping through an array
  for (let i = 0; i < teamMembers.length; i++) {
    Logger.log(`Welcome, ${teamMembers[i]}!`);
  }

  // A more modern way to loop
  teamMembers.forEach(member => {
    Logger.log(`Hello again, ${member}!`);
  });
}
```

---

14. How do I use a `for` loop?

Explanation: A `for` loop is used to repeat a block of code a specific number of times. It's perfect for iterating through arrays or rows in a spreadsheet. The basic structure has three parts: an initializer (`let i = 0`), a condition (`i < 10`), and an incrementer (`i++`).

Code Example:

```
function loopExample() {
  // This loop will run 5 times, with 'i' going from 0 to 4.
  for (let i = 0; i < 5; i++) {
    Logger.log(`This is loop number: ${i}`);
  }
}
```

---

15. What is an `if` statement?

Explanation: An `if` statement allows your script to make decisions. It runs a block of code **only if** a certain condition is true. You can add `else if` for more conditions or an `else` to run code if none of the conditions are met.

Code Example:

```
function decisionMaking() {
  const score = 85;

  if (score > 90) {
    Logger.log('Grade: A');
  } else if (score > 80) {
    Logger.log('Grade: B'); // This block will run
  } else {
    Logger.log('Grade: C or lower');
  }
}
```

Google Sheets

This is the most common use case for Apps Script. Learn how to read, write, and manipulate data in Google Sheets.

---

16. How do I read a value from a single cell?

Explanation: First, get the sheet you want to work with. Then, get the specific `Range` (cell) using a notation like `"A1"`. Finally, use the `getValue()` method to retrieve the data from that cell.

Code Example:

```
function readSingleCell() {
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName('Sheet1');
  // Get the range for cell A1
  const range = sheet.getRange('A1');
  // Get the value from that cell
  const value = range.getValue();
  Logger.log(`The value in A1 is: ${value}`);
}
```

---

17. How do I write a value to a single cell?

Explanation: Similar to reading, you first get the `Range` object for the cell you want to write to. Then, you use the `setValue()` method to place new data into it. Any existing data in that cell will be overwritten.

Code Example:

```
function writeSingleCell() {
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName('Sheet1');
  // Get the range for cell B1
  const range = sheet.getRange('B1');
  // Set the value of that cell
  range.setValue('Hello from Apps Script!');
}
```

---

18. What's the difference between `getValue()` and `getValues()`?

Explanation: This is a crucial and performance-related concept.

- `getValue()`: Reads the value from a **single cell** (a 1x1 range). It returns a single value (e.g., a string, number, or date).
- `getValues()`: Reads the values from a **range of multiple cells**. It always returns a **2D array** (an array of arrays), where each inner array represents a row. Even if you get a single column, it will be structured as `[[row1Value], [row2Value], ...]`.

---

19. How do I read an entire column of data?

Explanation: It's much more efficient to read all the data at once with `getValues()` rather than one cell at a time in a loop. To get a whole column, you can define the range using A1 notation (e.g., `"A2:A"` to get all of column A from row 2 down).

Code Example:

```
function readEntireColumn() {
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName('Sheet1');
  const lastRow = sheet.getLastRow(); // Find the last row with content

  // Get all values from column A, from row 2 to the last row
  // Using .getValues() is much faster than looping .getValue()
  const range = sheet.getRange(`A2:A${lastRow}`);
  const values = range.getValues(); // This is a 2D array: [['A2'], ['A3'], ...]

  // Loop through the 2D array
  for (let i = 0; i < values.length; i++) {
    // Access the value inside the inner array at index 0
    let cellValue = values[i][0];
    Logger.log(cellValue);
  }
}
```

---

20. How do I write an array of data to a sheet?

Explanation: Use `setValues()`, which is the plural counterpart to `setValue()`. It takes a 2D array and writes it to a specified range. The dimensions of the 2D array **must match** the dimensions of the range you select. For example, to write to 3 rows and 2 columns, your array must have 3 inner arrays, each with 2 elements.

Code Example:

```
function writeArrayToSheet() {
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName('Sheet2');
  const data = [
    ['Name', 'Email'],
    ['Lars', 'lars@example.com'],
    ['Naza', 'naza@example.com'],
    ['Maureen', 'maureen@example.com']
  ];

  // Get a range that matches the size of our data (4 rows, 2 columns)
  // Starting at row 1, column 1, for 4 rows, and 2 columns.
  const range = sheet.getRange(1, 1, data.length, data[0].length);

  // Set the values all at once
  range.setValues(data);
}
```

---

21. What is `getRange(row, column, numRows, numColumns)`?

Explanation: This is the most flexible way to select a range. Instead of A1 notation, you use numbers (integers).

- `row`: The starting row number (1 is the first row).
- `column`: The starting column number (1 is column A).
- `numRows` (optional): How many rows to select.
- `numColumns` (optional): How many columns to select.

Code Example:

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

 // Get a single cell (C5)
 // Row 5, Column 3
 const cellC5 = sheet.getRange(5, 3);
 cellC5.setValue('This is C5');
```

```
  // Get a 10-row by 4-column block starting at A1
  // Row 1, Column 1, for 10 rows, for 4 columns
  const dataBlock = sheet.getRange(1, 1, 10, 4);
  dataBlock.setBackground('#f3f3f3'); // Set background color for the block
}
```

---

22. How do I find the last row or last column with data?

Explanation: This is essential for working with dynamic data where the number of rows changes.

- `getLastRow()`: Returns the row number of the last row that contains any data.
- `getLastColumn()`: Returns the column number of the last column that contains any data.

Code Example:

```
function findLasts() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const lastRow = sheet.getLastRow();
  const lastCol = sheet.getLastColumn();

  Logger.log(`The last row with data is: ${lastRow}`);
  Logger.log(`The last column with data is: ${lastCol}`);

  // Now you can get all the data dynamically
  const allData = sheet.getRange(1, 1, lastRow, lastCol).getValues();
  Logger.log(allData);
}
```

---

23. How do I append a new row of data?

Explanation: Use the `appendRow()` method on a sheet object. This is a special helper function that automatically finds the first empty row and adds your data there. It takes a 1D array where each element corresponds to a column.

Code Example:

```
function addNewRow() {
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName('Logs');
  const timestamp = new Date();
  const user = Session.getActiveUser().getEmail();
  const message = 'New entry added.';

  // appendRow expects a 1D array of values
  sheet.appendRow([timestamp, user, message]);
}
```

---

24. How do I clear content from a range?

Explanation: You have several options for clearing, available on the `Range` object:

- `clearContent()`: Deletes only the values, but keeps formatting (bold, colors, etc.).
- `clearFormat()`: Deletes only the formatting, but keeps the values.
- `clear()`: Deletes everything—values, formatting, notes, etc.

Code Example:

```
function clearData() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const range = sheet.getRange('A1:C10');

  // To delete only the text/values
  range.clearContent();

  // To reset all formatting
  // range.clearFormat();

  // To wipe everything in the range
  // range.clear();
}
```

---

25. How do I delete a row or a column?

Explanation: You can delete entire rows or columns using the `deleteRow()` or `deleteColumn()` methods on the `Sheet` object. You provide the position of the row/column you want to remove.

Code Example:

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

  // Delete the 5th row
  sheet.deleteRow(5);

  // Delete the 3rd column (Column C)
  sheet.deleteColumn(3);
}
```

---

26. How do I create a new sheet?

Explanation: Use the `insertSheet()` method on the `Spreadsheet` object. You can optionally give it a name.

Code Example:

```
function createNewSheet() {
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const newSheetName = 'Archive ' + new Date().toLocaleDateString();

  // Check if a sheet with that name already exists
  if (!ss.getSheetByName(newSheetName)) {
    ss.insertSheet(newSheetName);
    SpreadsheetApp.getUi().alert(`Sheet "${newSheetName}" created!`);
  } else {
    SpreadsheetApp.getUi().alert(`Sheet "${newSheetName}" already exists.`);
  }
}
```

---

27. How do I get the URL of the active spreadsheet?

Explanation: The `Spreadsheet` object has a `getUrl()` method that returns its public URL.

Code Example:

```
function getSpreadsheetUrl() {
  const url = SpreadsheetApp.getActiveSpreadsheet().getUrl();
  Logger.log(url);
  SpreadsheetApp.getUi().alert(`This spreadsheet's URL is: ${url}`);
}
```

---

28. How do I get the ID of the active spreadsheet?

Explanation: The `Spreadsheet` object has a `getId()` method. The ID is the long string of random characters in the spreadsheet's URL, and it's often needed to work with other services like DriveApp.

Code Example:

```
function getSpreadsheetId() {
  const id = SpreadsheetApp.getActiveSpreadsheet().getId();
  Logger.log(`The ID is: ${id}`);
}
```

---

29. How do I change a cell's background color?

Explanation: Use `setBackground()` on a `Range` object. You can use color names (like `"red"`) or hex codes (like `"#ff0000"`).

Code Example:

```
function formatCells() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const range = sheet.getRange('A1:A10');

  // Set the background color for all cells in the range
  range.setBackground('#cfe2f3'); // A light blue color

  // You can also change font color, weight, etc.
  range.setFontColor('black');
  range.setFontWeight('bold');
}
```

---

30. How do I use the `onEdit(e)` event object?

Explanation: The `e` in `onEdit(e)` is the **event object**. It contains information about the edit that just happened. This is incredibly powerful. Some useful properties are:

- `e.range`: The `Range` object that was edited.
- `e.value`: The new value that was entered into the cell.
- `e.oldValue`: The value that was in the cell before the edit.
- `e.source`: The `Spreadsheet` object where the edit occurred.
- `e.user`: The email of the user who made the edit.

Code Example: Add a timestamp in the next column whenever a cell in column A is edited.

```javascript
/**
 * Runs automatically when a user edits a cell.
 * @param {Object} e The event object.
 */
function onEdit(e) {
  const range = e.range;
  const sheet = range.getSheet();

  // Check if the edit was in column A (column 1) and in the 'Tasks' sheet.
  if (range.getColumn() == 1 && sheet.getName() == 'Tasks') {
    // Get the cell next to it in column B and set the current date.
    const adjacentCell = sheet.getRange(range.getRow(), 2);
    adjacentCell.setValue(new Date()).setNumberFormat('yyyy-mm-dd h:mm:ss');
  }
}
```

---

31. How do I create a custom function?

Explanation: A custom function is a function you write in Apps Script that you can use in a Google Sheet just like a built-in function (e.g., `=SUM()`). It must return a value.

Code Example: Create a function `=DOUBLE(A1)` that multiplies a cell's value by 2.

```javascript
/**
 * Doubles the input value.
 * @param {number} input The value to double.
 * @returns The doubled value.
 * @customfunction
 */
function DOUBLE(input) {
  if (typeof input !== 'number') {
    return 'Input must be a number.';
  }
  return input * 2;
}
```

After saving, you can go into any cell and type `=DOUBLE(5)` or `=DOUBLE(A1)`.

---

32. How do I copy a sheet to another spreadsheet?

Explanation: Use the `copyTo()` method on a `Sheet` object. This method takes the destination `Spreadsheet` object as its argument. You first need to open the destination spreadsheet by its ID.

Code Example:

```
function copySheetToAnotherSpreadsheet() {
  const sourceSpreadsheet = SpreadsheetApp.getActiveSpreadsheet();
  const sheetToCopy = sourceSpreadsheet.getSheetByName('Monthly Report');

  const destinationSpreadsheetId = 'PASTE_THE_ID_OF_THE_OTHER_SPREADSHEET_HERE';
  const destinationSpreadsheet = SpreadsheetApp.openById(destinationSpreadsheetId);

  // Copy the sheet and give it a new name in the destination
  const newSheet = sheetToCopy.copyTo(destinationSpreadsheet);
  newSheet.setName('Copied Report - ' + new Date().getMonth());

  SpreadsheetApp.getUi().alert('Sheet copied successfully!');
}
```

33. How do I hide or unhide a sheet?

Explanation: Use the `hideSheet()` and `unhideSheet()` methods on a `Sheet` object.

Code Example:

```
function toggleSheetVisibility() {
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName('Confidential
Data');

  if (sheet.isSheetHidden()) {
    sheet.showSheet(); // 'showSheet' is the method to unhide
    Logger.log('Sheet is now visible.');
  } else {
    sheet.hideSheet();
    Logger.log('Sheet is now hidden.');
  }
}
```

34. How can I protect a range or sheet?

Explanation: You can apply protections to prevent users from editing certain parts of your sheet. Use the `protect()` method on a `Range` or `Sheet` object. You can configure who is allowed to edit the protected area.

Code Example:

```
function protectHeaderRow() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const range = sheet.getRange('A1:Z1'); // Protect the entire first row

  // Protect the range
  const protection = range.protect().setDescription('Header Row Protection');
  // Ensure the current user is an editor and remove all other editors.
  const me = Session.getEffectiveUser();
  protection.addEditor(me);
  protection.removeEditors(protection.getEditors());

  // Optionally add other specific editors
  // protection.addEditor('other.user@example.com');

  // You can also warn users when they try to edit
  protection.setWarningOnly(true);
}
```

---

35. How do I find a value in a sheet (like Ctrl+F)?

Explanation: While there isn't a direct "find" method, the standard way is to get all the data into a 2D array and then loop through it to find the value you're looking for. This is extremely fast.

Code Example:

```
function findValueInSheet(valueToFind) {
  const sheet = SpreadsheetApp.getActiveSheet();
  const data = sheet.getDataRange().getValues(); // Get all data

  for (let row = 0; row < data.length; row++) {
    for (let col = 0; col < data[row].length; col++) {
      if (data[row][col] == valueToFind) {
        // Found it! Return the cell notation.
        // Add 1 because arrays are 0-indexed but sheets are 1-indexed.
        const cellNotation = sheet.getRange(row + 1, col + 1).getA1Notation();
        Logger.log(`Found "${valueToFind}" at cell ${cellNotation}`);
        return cellNotation;
      }
    }
  }
  Logger.log(`Value "${valueToFind}" not found.`);
  return null;
}
```

---

36. How do I sort data in a sheet?

Explanation: You can sort a `Range` using the `sort()` method. You provide an object specifying which column to sort by and whether it should be ascending.

Code Example:

```
function sortData() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const range = sheet.getRange("A2:C100"); // The range to sort, excluding headers

  // Sort by column B (the 2nd column), in ascending order.
  range.sort({column: 2, ascending: true});
}
```

---

37. How do I set a formula in a cell?

Explanation: Instead of `setValue()`, use `setFormula()` on a `Range` object. The formula should be a string, just like you would type it into the cell.

Code Example:

```
function applyFormulas() {
  const sheet = SpreadsheetApp.getActiveSheet();
  // Set a simple SUM formula in cell D2
  sheet.getRange('D2').setFormula('=SUM(B2:C2)');

  // Set a formula with text
  sheet.getRange('E2').setFormula('=IF(D2>100, "Goal Met", "Under Goal")');
}
```

---

38. How do I get the displayed value of a cell (e.g., for dates or currency)?

Explanation: `getValue()` returns the raw underlying data (a date object, a number). If you want the formatted string that you actually *see* in the sheet (e.g., "$5.00" or "9/15/2025"), use `getDisplayValue()`. The plural version, `getDisplayValues()`, also exists for ranges.

Code Example:

```
function getDisplayAndRawValues() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const range = sheet.getRange('A1'); // Assume A1 has a date like 9/15/2025

  const rawValue = range.getValue(); // This will be a JavaScript Date object
  const displayValue = range.getDisplayValue(); // This will be the string "9/15/2025"

  Logger.log(`Raw Value: ${rawValue}`);
  Logger.log(`Display Value: ${displayValue}`);
}
```

---

39. How do I use `SpreadsheetApp.flush()`?

Explanation: Apps Script sometimes batches operations (like `setValue()`) to be more efficient. `SpreadsheetApp.flush()` forces all pending changes to be applied to the spreadsheet immediately. This is useful when your script depends on a calculation from a formula you just set, or before you show a UI element that depends on the new data being present.

Code Example:

```
function forceUpdate() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const cell = sheet.getRange('A1');

  cell.setFormula('=NOW()');

  // If we read the value right away, it might still be the old one.
  // flush() ensures the formula calculates before the next line runs.
  SpreadsheetApp.flush();

  const calculatedValue = cell.getDisplayValue();
  SpreadsheetApp.getUi().alert(`The calculated time is: ${calculatedValue}`);
}
```

---

40. How do I filter data in a sheet?

Explanation: You can create, remove, and manage spreadsheet filters programmatically. Get the `Filter` object from a range and then set its criteria.

Code Example:

```
function applyAFilter() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const range = sheet.getDataRange();

  // Remove any existing filter to start fresh
  if (sheet.getFilter()) {
    sheet.getFilter().remove();
  }
  // Create a new filter on the data range
  const filter = range.createFilter();

  // Create a filter criteria to only show rows where column 3 (C) is not blank
  const criteria = SpreadsheetApp.newFilterCriteria()
    .whenCellNotEmpty()
    .build();

  filter.setColumnFilterCriteria(3, criteria);
}
```

Google Docs, Drive, & Files

Automate document creation, file management, and folder organization in Google Drive.

---

41. How do I create a new Google Doc?

Explanation: Use `DocumentApp.create()` and provide a name for the new document. This creates a new file in the root of your Google Drive.

Code Example:

```
function createNewGoogleDoc() {
  const docName = 'My New Report - ' + new Date().toLocaleDateString();
  const doc = DocumentApp.create(docName);

  // Get the URL and ID of the newly created document
  const url = doc.getUrl();
  const id = doc.getId();
  Logger.log(`Created document "${docName}" at ${url}`);
  SpreadsheetApp.getUi().alert(`New Doc created! URL: ${url}`);
}
```

---

42. How do I open a Doc by its ID?

Explanation: Use `DocumentApp.openById()` with the document's ID string.

Code Example:

```
function openSpecificDoc() {
  const docId = 'PASTE_YOUR_DOCUMENT_ID_HERE';
  const doc = DocumentApp.openById(docId);
  Logger.log(`Opened document: ${doc.getName()}`);
}
```

---

43. How do I get the text from a Google Doc?

Explanation: First, you get the `body` of the document. Then you can use `getText()` to get all the text as a single string, or `getParagraphs()` to get an array of all paragraph elements.

Code Example:

```
function readDocText() {
  const doc = DocumentApp.getActiveDocument(); // For a bound script
  const body = doc.getBody();
  const fullText = body.getText();

  Logger.log(fullText);
}
```

44. How do I add a paragraph to a Google Doc?

Explanation: Use the `appendParagraph()` method on the document's `body` object. You can also add titles, lists, and tables.

Code Example:

```
function addContentToDoc() {
  const doc = DocumentApp.getActiveDocument();
  const body = doc.getBody();

  body.appendParagraph('This is the main title')
      .setHeading(DocumentApp.ParagraphHeading.TITLE);

  body.appendParagraph('This is a new paragraph added by a script.');
  body.appendParagraph('Here is another one.');

  // Add a bulleted list item
  body.appendListItem('First list item.');
  body.appendListItem('Second list item.');
}
```

45. How do I find and replace text in a Google Doc?

Explanation: Use the `replaceText()` method on the document's `body`. It supports simple text replacement and even regular expressions. The search pattern is case-sensitive.

Code Example:

```
function findAndReplace() {
  const doc = DocumentApp.getActiveDocument();
  const body = doc.getBody();
  // Simple replacement
  body.replaceText('{{CLIENT_NAME}}', 'Lars S.');

  // Using regular expressions to ignore case
  // The 'g' flag means 'global' - replace all instances
  body.replaceText('(?i)september', 'October'); // (?i) is regex for case-insensitive
}
```

---

46. How do I create a folder in Google Drive?

Explanation: Use `DriveApp.createFolder()` and provide a name.

Code Example:

```
function createDriveFolder() {
  const folderName = 'Project Invoices';
  const folder = DriveApp.createFolder(folderName);
  Logger.log(`Created folder named "${folder.getName()}" with ID: ${folder.getId()}`);
}
```

---

47. How do I get a folder by its name or ID?

Explanation:

- `getFolderById(id)`: The most reliable way. IDs are unique.
- `getFoldersByName(name)`: This returns a `FolderIterator`, which is like an array, because you can have multiple folders with the same name. You usually need to check if it `hasNext()` and then grab the first one with `next()`.

Code Example:

```
function getDriveFolder() {
  // By ID (best method)
  const folderId = 'PASTE_YOUR_FOLDER_ID_HERE';
  const folderById = DriveApp.getFolderById(folderId);
  Logger.log(`Folder found by ID: ${folderById.getName()}`);

  // By Name
  const folderName = 'Project Invoices';
  const folders = DriveApp.getFoldersByName(folderName);
  if (folders.hasNext()) {
    const folderByName = folders.next();
    Logger.log(`Folder found by name: ${folderByName.getName()}`);
  } else {
    Logger.log(`No folder named "${folderName}" found.`);
  }
}
```

---

48. How do I list all files in a folder?

Explanation: Use `getFiles()` on a `Folder` object. This returns a `FileIterator`. You use a `while` loop with `hasNext()` to go through all the files.

Code Example:

```
function listFilesInFolder() {
  const folder = DriveApp.getFolderById('PASTE_YOUR_FOLDER_ID_HERE');
  const files = folder.getFiles();

  while (files.hasNext()) {
    const file = files.next();
    Logger.log(`File Name: ${file.getName()}, URL: ${file.getUrl()}`);
  }
}
```

---

49. How do I create a file from text content?

Explanation: Use `createFile(name, content)` on a `Folder` object (or on `DriveApp` to create it in the root). The content can be plain text, HTML, CSV, etc.

Code Example:

```
function createTextFile() {
  const folder = DriveApp.getFolderById('PASTE_YOUR_FOLDER_ID_HERE');
  const fileName = 'log.txt';
  const content = `Server check completed at ${new Date()}`;

  folder.createFile(fileName, content);
  Logger.log(`File "${fileName}" created.`);
}
```

---

50. How do I move a file to another folder?

Explanation: Use the `moveTo()` method on a `File` object. It takes the destination `Folder` object as its argument.

Code Example:

```
function moveFile() {
  const file = DriveApp.getFileById('PASTE_THE_FILE_ID_HERE');
  const destinationFolder = 
DriveApp.getFolderById('PASTE_THE_DESTINATION_FOLDER_ID_HERE');

  file.moveTo(destinationFolder);
  Logger.log(`Moved "${file.getName()}" to "${destinationFolder.getName()}".`);
}
```

Gmail & Calendar

Send emails, manage drafts, and create calendar events programmatically.

---

51. How do I send a simple email?

Explanation: Use `GmailApp.sendEmail()`. At a minimum, you need to provide the recipient, subject, and body.

Code Example:

```
function sendSimpleEmail() {
  const recipient = 'user@example.com';
  const subject = 'Hello from Google Apps Script';
  const body = 'This email was sent automatically using a script!';

  GmailApp.sendEmail(recipient, subject, body);
  Logger.log('Email sent.');
}
```

---

52. How do I send an email with HTML content?

Explanation: In `sendEmail()`, you can provide an options object as the fourth parameter. In this object, you can specify an `htmlBody` property.

Code Example:

```
function sendHtmlEmail() {
  const recipient = 'user@example.com';
  const subject = 'Your Weekly Report';
  const htmlBody = `
    <html>
      <body>
        <h1>Report Summary</h1>
        <p>Here is the data you requested.</p>
        <p style="color: blue;">The process completed successfully.</p>
      </body>
    </html>
  `;

  GmailApp.sendEmail(recipient, subject, '', { htmlBody: htmlBody });
  Logger.log('HTML email sent.');
}
```

---

53. How do I send an email with an attachment?

Explanation: Add an `attachments` property to the options object. The value should be an array of file blobs. You can get a file blob from Google Drive using `getBlob()`.

Code Example:

```
function sendEmailWithAttachment() {
  const recipient = 'user@example.com';
  const subject = 'Document Attached';
  const body = 'Please see the attached file.';
  const file = DriveApp.getFileById('PASTE_FILE_ID_HERE');

  GmailApp.sendEmail(recipient, subject, body, {
    attachments: [file.getBlob()]
  });
  Logger.log('Email with attachment sent.');
}
```

---

54. What is my remaining daily email quota?

Explanation: Google limits how many emails you can send per day (100 for consumer accounts, 1,500 for Workspace). You can check your remaining quota with `getRemainingDailyQuota()`.

Code Example:

```
function checkEmailQuota() {
  const quota = MailApp.getRemainingDailyQuota(); // MailApp or GmailApp work
  Logger.log(`You can send ${quota} more emails today.`);
}
```

---

55. How do I get my own email address?

Explanation: Use `Session.getActiveUser().getEmail()`. This gets the email of the user running the script. `getEffectiveUser()` gets the email of the user who owns the script (they might be different if a script is deployed as a web app).

Code Example:

```
function whoAmI() {
  const activeUser = Session.getActiveUser().getEmail();
  const effectiveUser = Session.getEffectiveUser().getEmail();

  Logger.log(`Script is running as: ${activeUser}`);
  Logger.log(`Script is owned by: ${effectiveUser}`);
}
```

---

56. How do I create a Google Calendar event?

Explanation: Use `CalendarApp.createEvent()`. You need to provide a title, a start time, and an end time (as JavaScript `Date` objects).

Code Example:

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

  const title = 'Team Meeting';
  const startTime = new Date('2025-10-28T10:00:00-04:00'); // Use ISO 8601 format
  const endTime = new Date('2025-10-28T11:00:00-04:00');

  calendar.createEvent(title, startTime, endTime);
  Logger.log('Event created.');
}
```

---

57. How do I create a calendar event with more details (guests, description)?

Explanation: Use the more detailed version of `createEvent()` which takes an options object. You can specify guests, location, description, and more.

Code Example:

```
function createDetailedEvent() {
  const calendar = CalendarApp.getDefaultCalendar();

  const title = 'Project Kickoff';
  const startTime = new Date(); // now
  const endTime = new Date(startTime.getTime() + (60 * 60 * 1000)); // 1 hour from now

  const options = {
    description: 'This is the kickoff meeting for the new website project.',
    location: 'Virtual / Google Meet',
    guests: 'naza@example.com,maureen@example.com',
    sendInvites: true // This will email the guests
  };

  calendar.createEvent(title, startTime, endTime, options);
  Logger.log('Detailed event created and invites sent.');
}
```

---

58. How do I get events from a calendar?

Explanation: Use `getEvents(startTime, endTime)` on a `Calendar` object to get all events within a specific date range. This returns an array of `CalendarEvent` objects.

Code Example:

```
function listTodaysEvents() {
  const calendar = CalendarApp.getDefaultCalendar();
  const now = new Date();
  const startOfDay = new Date(now.setHours(0, 0, 0, 0));
  const endOfDay = new Date(now.setHours(23, 59, 59, 999));

  const events = calendar.getEvents(startOfDay, endOfDay);

  if (events.length === 0) {
    Logger.log('No events scheduled for today.');
    return;
  }

  Logger.log('Today\'s Events:');
  events.forEach(event => {
    Logger.log(`- ${event.getTitle()} at ${event.getStartTime().toLocaleTimeString()}`);
  });
}
```

---

59. How do I get a calendar by its name?

Explanation: Use `CalendarApp.getCalendarsByName(name)`. Like with Drive folders, this returns an array since names may not be unique.

Code Example:

```
function getWorkCalendar() {
  const calendarName = 'Work';
  const calendars = CalendarApp.getCalendarsByName(calendarName);

  if (calendars.length > 0) {
    const workCalendar = calendars[0];
    Logger.log(`Found calendar: ${workCalendar.getName()}`);
    return workCalendar;
  }
  return null;
}
```

---

60. How do I delete a calendar event?

Explanation: Call the `deleteEvent()` method on the `CalendarEvent` object itself.

Code Example:

```
function cancelMeeting() {
  const calendar = CalendarApp.getDefaultCalendar();
  const now = new Date();
  const oneHourFromNow = new Date(now.getTime() + (60 * 60 * 1000));

  // Find events in the next hour with "Meeting" in the title
  const events = calendar.getEvents(now, oneHourFromNow);
  events.forEach(event => {
    if (event.getTitle().includes('Meeting')) {
      Logger.log(`Deleting event: ${event.getTitle()}`);
      event.deleteEvent();
    }
  });
}
```

Triggers and Automation

Make your scripts run automatically based on time, edits, or other events.

---

61. What is a trigger?

Explanation: A trigger is what causes a script function to run automatically without you having to click the "Run" button. There are several types:

- **Simple Triggers:** Special function names like `onOpen()` and `onEdit()` that run on specific, simple events. They have some limitations.
- **Installable Triggers:** Triggers you create manually (through the UI or with code). They can do more than simple triggers (like send emails) and can run on different events (time-driven, on form submit, etc.).

---

62. How do I set up a time-driven trigger?

Explanation: Go to the script editor, click the "Triggers" icon (a clock) on the left sidebar. Click "Add Trigger", choose the function to run, select "Time-driven" as the event source, and then configure how often you want it to run (e.g., every hour, every day at 9 am).

---

63. How do I create a trigger programmatically?

Explanation: You can use the `ScriptApp` service to build and delete triggers with code. This is useful for automating the setup of your scripts.

Code Example:

```
function createTimeDrivenTrigger() {
  // First, delete any existing triggers for this function to avoid duplicates.
  deleteTriggersForFunction('dailyReport');

  // Create a trigger to run the 'dailyReport' function every day at 8am.
  ScriptApp.newTrigger('dailyReport')
    .timeBased()
    .atHour(8)
    .everyDays(1)
    .create();

  Logger.log('Trigger for dailyReport created.');
}

function deleteTriggersForFunction(functionName) {
  const allTriggers = ScriptApp.getProjectTriggers();
  allTriggers.forEach(trigger => {
    if (trigger.getHandlerFunction() === functionName) {
      ScriptApp.deleteTrigger(trigger);
    }
  });
}
```

---

64. What is an "On Form Submit" trigger?

Explanation: This is a powerful installable trigger for scripts bound to a Google Form or a Sheet linked to a Form. It runs a function every time a user submits a response to the form.

The event object e contains the form response data.

Code Example: Log the new form response to the console.

```
JavaScript
/**
 * Runs when a Google Form is submitted.
 * @param {Object} e The event object containing the form response.
 */
function onFormSubmit(e) {
  // e.values contains the submitted answers in an array
  const responses = e.values;
  const timestamp = responses[0]; // The first column is always the timestamp
  const userEmail = responses[1];
  const userFeedback = responses[2];

  Logger.log(`New feedback from ${userEmail}: "${userFeedback}"`);

  // You could then send an email notification, create a calendar event, etc.
}
```

---

65. What is an "On Change" trigger for a spreadsheet?

Explanation: This is an installable trigger that's more powerful than `onEdit()`. It runs when the *structure* of a spreadsheet changes, such as when a new sheet is added, a row is inserted or deleted, or a sheet is renamed.

Code Example:

```
function onSpreadsheetChange(e) {
  // e.changeType will be "EDIT", "INSERT_ROW", "REMOVE_COLUMN", "OTHER", etc.
  Logger.log(`A change of type "${e.changeType}" occurred.`);
  if (e.changeType === 'INSERT_ROW') {
    SpreadsheetApp.getActiveSpreadsheet().toast('A new row was inserted!');
  }
}
```

---

66. How do I list all triggers for a project?

Explanation: Use `ScriptApp.getProjectTriggers()` to get an array of all triggers associated with the current script.

Code Example:

```
function listAllTriggers() {
  const triggers = ScriptApp.getProjectTriggers();
  if (triggers.length === 0) {
    Logger.log('This project has no triggers.');
    return;
  }

  triggers.forEach(trigger => {
    Logger.log(`
      Function: ${trigger.getHandlerFunction()}
      Type: ${trigger.getEventType()}
      Source: ${trigger.getTriggerSource()}
    `);
  });
}
```

---

67. How do I delete a trigger programmatically?

Explanation: Loop through the project triggers, find the one you want to delete (e.g., by its handler function name), and then use `ScriptApp.deleteTrigger()`.

Code Example: (See `deleteTriggersForFunction` in question #63)

---

68. What are the limitations of simple triggers (`onOpen`, `onEdit`)?

Explanation: Simple triggers run with limited authorization. This means they **cannot** perform certain actions, such as:

- Sending emails (`GmailApp`)
- Accessing other files they aren't bound to (`DriveApp.getFileById`)
- Creating calendar events (`CalendarApp`)
- Anything that requires user data permissions beyond the current file. If you need to do any of these things, you must use an **installable trigger**.

---

69. Why did my time-driven trigger fail?

Explanation: A trigger can fail for many reasons. Check the "Executions" log for the error message. Common causes include:

- **Permissions Error:** The script's permissions were revoked or changed. Re-authorizing the script usually fixes this.
- **Service Outage:** A Google service (like Sheets) was temporarily down.
- **Code Error:** A change in your code introduced a bug.
- **Exceeding Quotas:** Your script ran too long (max 6 minutes for consumers) or used too many resources.

---

70. How can I temporarily disable all triggers?

Explanation: In the Triggers UI, each trigger has an on/off toggle. Programmatically, the easiest way is to rename the function the trigger calls. For example, rename `dailyReport()` to `dailyReport_disabled()`. The trigger will fail to find the function and won't run.

User Interface (UI) & Dialogs

Interact with the user through menus, dialogs, and sidebars.

---

71. How do I create a custom menu?

Explanation: Use `SpreadsheetApp.getUi().createMenu()` inside an `onOpen()` function. This ensures the menu appears every time the user opens the file. You use `addItem()` to add menu items, specifying the text to display and the name of the function to run when it's clicked.

Code Example:

```
function onOpen() {
  SpreadsheetApp.getUi()
      .createMenu('Automation Tools')
      .addItem('Sort Data', 'sortDataFunction')
      .addSeparator()
      .addSubMenu(SpreadsheetApp.getUi().createMenu('Reports')
          .addItem('Generate Weekly Report', 'generateWeeklyReport'))
      .addToUi();
}

// These are the functions that will be called by the menu items.
function sortDataFunction() { /* ... code to sort ... */ }
function generateWeeklyReport() { /* ... code to generate a report ... */ }
```

---

72. What is `SpreadsheetApp.getUi()`?

Explanation: `getUi()` returns the **UI environment** for the active spreadsheet, document, or form. This `Ui` object is your gateway to creating menus, alerts, prompts, and sidebars. You cannot use it in a standalone script that isn't opened within a document context.

---

73. How do I show a simple alert message?

Explanation: Use `ui.alert()`. It shows a simple dialog box with an "OK" button. It's great for confirming that an action has been completed.

Code Example:

```
function showConfirmation() {
  const ui = SpreadsheetApp.getUi();
  ui.alert('Success!', 'The report has been generated successfully.', ui.ButtonSet.OK);
}
```

---

74. How do I ask the user for text input?

Explanation: Use `ui.prompt()`. This shows a dialog with a text input field. You can check which button the user clicked (`OK` or `Cancel`) and get the text they entered.

Code Example:

```
function askForName() {
  const ui = SpreadsheetApp.getUi();
  const response = ui.prompt('Enter Your Name', 'Please enter your first name:',
ui.ButtonSet.OK_CANCEL);

  // Check if the user clicked OK
  if (response.getSelectedButton() == ui.Button.OK) {
    const name = response.getResponseText();
    SpreadsheetApp.getActiveSheet().getRange('A1').setValue(`Welcome, ${name}!`);
  } else {
    ui.alert('You cancelled the prompt.');
  }
}
```

---

75. How do I show a custom sidebar?

Explanation: A sidebar is an HTML panel that opens on the right side of the editor. You create an HTML file in your Apps Script project and then use `HtmlService.createHtmlOutputFromFile()` to display it.

Code Example:

Code.gs

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

MySidebar.html

```HTML
<!DOCTYPE html>
<html>
  <head>
    <base target="_top">
  </head>
  <body>
    <h1>Sidebar Content</h1>
    <p>This is a custom sidebar.</p>
    <input type="button" value="Close" onclick="google.script.host.close()" />
  </body>
</html>
```

76. How do I create a custom dialog (modal)?

Explanation: Similar to a sidebar, but it's a modal dialog that appears in the center of the screen. The user must interact with it before they can continue using the spreadsheet. Use `showModalDialog()`.

Code Example:

Code.gs

```
JavaScript
function showModalDialog() {
  const html = HtmlService.createHtmlOutputFromFile('MyDialog')
      .setWidth(300)
      .setHeight(200);
  SpreadsheetApp.getUi().showModalDialog(html, 'My Custom Dialog');
}
```

MyDialog.html (similar HTML structure to the sidebar)

---

77. How do I call a server-side Apps Script function from client-side HTML?

Explanation: This is key for interactive UI. You use `google.script.run`. This is an asynchronous API. You can also add `withSuccessHandler()` and `withFailureHandler()` to process the return value or handle errors.

Code Example:

Code.gs

```
JavaScript
function getSheetData() {
  return SpreadsheetApp.getActiveSheet().getDataRange().getValues();
}
```

MySidebar.html

```
HTML
<!DOCTYPE html>
<html>
  <head>
    <base target="_top">
  </head>
  <body>
    <button onclick="fetchData()">Get Data</button>
    <div id="output"></div>
    <script>
      function fetchData() {
        document.getElementById('output').innerText = 'Loading...';
        google.script.run
          .withSuccessHandler(onDataReceived)
          .withFailureHandler(onFailure)
          .getSheetData();
      }

      function onDataReceived(data) {
        // data is the 2D array returned from getSheetData()
        document.getElementById('output').innerText = JSON.stringify(data);
      }

      function onFailure(error) {
        document.getElementById('output').innerText = 'Error: ' + error.message;
      }
    </script>
  </body>
</html>
```

78. What is `HtmlService`?

Explanation: `HtmlService` is the Apps Script service that lets you serve web pages. You can use it to build user interfaces (sidebars, dialogs) or even full standalone web apps. It has security features, like sandboxing, to protect user data.

---

79. What does `<base target="_top">` do in the HTML?

Explanation: This tag is required in the `<head>` of your HTML file. It tells any links in your HTML to open in the main browser window, rather than inside the small iframe of the sidebar or dialog, preventing your users from getting trapped.

---

80. How do I show a "toast" message?

Explanation: A toast is a small, black, non-disruptive message that appears at the bottom-right corner of the screen and disappears after a few seconds. It's great for quick, subtle notifications. Use `spreadsheet.toast()`.

Code Example:

```
function showToastNotification() {
  SpreadsheetApp.getActiveSpreadsheet().toast('Your data has been processed.', 'Status Update', 5);
  // The '5' is the timeout in seconds.
}
```

APIs, External Services & Advanced Topics

Connect to other services, deploy web apps, and manage your projects.

---

81. How do I fetch data from an external API?

Explanation: Use `UrlFetchApp.fetch()`. This allows your script to make HTTP requests (GET, POST, etc.) to any public API on the internet. It returns an `HTTPResponse` object, and you use `getContentText()` to get the response body.

Code Example: Fetch a random joke from an API.

JavaScript

```
function getApiData() {
  const url = 'https://official-joke-api.appspot.com/random_joke';
  const response = UrlFetchApp.fetch(url);
  const json = response.getContentText();
  const data = JSON.parse(json); // Parse the JSON string into an object

  Logger.log(`Setup: ${data.setup}`);
  Logger.log(`Punchline: ${data.punchline}`);
}
```

---

82. How do I make a POST request with a payload?

Explanation: Add an `options` object as a second parameter to `UrlFetchApp.fetch()`. In the options, you can specify the `method` (e.g., 'post'), `headers`, and the `payload` (the data you are sending).

Code Example:

```
function postDataToApi() {
  const url = 'https://httpbin.org/post'; // A service that echoes your request
  const data = {
    user: 'Lars',
    projectId: 12345,
    status: 'complete'
  };
  const options = {
    method: 'post',
    contentType: 'application/json',
    // Convert the JavaScript object to a JSON string.
    payload: JSON.stringify(data)
  };

  const response = UrlFetchApp.fetch(url, options);
  Logger.log(response.getContentText());
}
```

---

83. How do I parse JSON data?

Explanation: When you get data from an API, it's usually a JSON *string*. To use it as a JavaScript object, you need to parse it using `JSON.parse()`.

Code Example: (See question #81)

---

84. How do I convert an object or array to a JSON string?

Explanation: Use `JSON.stringify()`. This is necessary when you need to send data in the `payload` of a POST request.

Code Example: (See question #82)

---

85. How do I deploy a script as a web app?

Explanation: A web app is a script that has a public URL and can be visited by anyone (if you allow it).

1. Your script must have a `doGet(e)` or `doPost(e)` function. This function must return an `HtmlOutput` or `ContentOutput` object.

2. In the editor, click **Deploy > New deployment**.
3. Select "Web app" as the type.
4. Configure the access settings (e.g., who can run it, who can visit the URL).
5. You'll get a URL that you can share.

Code Example:

```
/**
 * This function runs when someone visits the web app URL.
 * @param {Object} e The event parameter.
 */
function doGet(e) {
  // Return simple text
  // return ContentService.createTextOutput('Hello, world!');

  // Or return HTML
  return HtmlService.createHtmlOutput('<h1>Welcome to my Web App!</h1>');
}
```

---

86. What's the difference between `doGet(e)` and `doPost(e)`?

Explanation: These are special functions for web apps, similar to `onOpen()`.

- `doGet(e)`: Runs when someone visits the URL directly in their browser (a GET request). The event parameter `e` contains any URL query parameters.
- `doPost(e)`: Runs when data is sent to the URL from a form or another service (a POST request). The event parameter `e` contains the data that was sent.

---

87. How do I manage script versions and deployments?

Explanation: When you deploy a web app or an add-on, you are deploying a specific, saved *version* of your code. If you make changes to your code, they will **not** be live on the deployed URL until you create a **new version** and update the deployment.

1. **Save a new version:** File > Manage versions > Save new version.

2. **Update deployment:** Deploy > Manage deployments, select your deployment, click the pencil (Edit) icon, and choose the new version.

---

88. How do I use script properties?

Explanation: `PropertiesService` is a way to store simple key-value string data. It's like a small database for your script. This is the perfect place to store settings, API keys, or information that needs to persist between script executions. There are three types:

- `getUserProperties()`: Stores data for the current user only.
- `getScriptProperties()`: Stores data for all users of the script.
- `getDocumentProperties()`: For add-ons, stores data for the current document only.

Code Example:

```
function storeAndRetrieveApiKey() {
  const scriptProperties = PropertiesService.getScriptProperties();

  // Store a property
  scriptProperties.setProperty('API_KEY', 'abcdef123456');
  Logger.log('API Key saved.');

  // Retrieve a property
  const apiKey = scriptProperties.getProperty('API_KEY');
  Logger.log(`Retrieved API Key: ${apiKey}`);

  // Delete a property
  // scriptProperties.deleteProperty('API_KEY');
}
```

---

89. How do I connect to a JDBC database?

Explanation: `Jdbc` is a service that lets you connect to external databases like MySQL, Microsoft SQL Server, and Oracle. You use `Jdbc.getConnection(url, user, password)` to establish a connection.

Code Example: (Conceptual - requires a real database)

JavaScript

```
function queryMySqlDatabase() {
  const dbUrl = 'jdbc:mysql://<your-db-host>:3306/<your-db-name>';
  const user = '<your-username>';
  const password = '<your-password>';

  try {
    const conn = Jdbc.getConnection(dbUrl, user, password);
    const stmt = conn.createStatement();
    const results = stmt.executeQuery('SELECT name, email FROM users');

    while (results.next()) {
      Logger.log(`Name: ${results.getString('name')}, Email: ${results.getString('email')}`);
    }

    results.close();
    stmt.close();
  } catch (e) {
    Logger.log('Error connecting to database: ' + e.message);
  } finally {
    if (conn) {
      conn.close();
    }
  }
}
```

90. What is an Apps Script library?

Explanation: A library is a way to reuse code across multiple Apps Script projects. You can take one script project, save a version of it, and then include it in another project. This is great for utility functions that you use all the time.

1. In the "reusable" project, go to **Project Settings** (gear icon) and copy the **Script ID**.
2. In the "main" project, go to the **Libraries** section (+) and paste the Script ID.
3. Choose a version and an identifier (e.g., `MyUtils`). You can now call functions from that library like `MyUtils.myFunction()`.

---

91. What is the execution time limit for a script?

Explanation: The maximum execution time depends on your account type:

- **Consumer Account (gmail.com):** 6 minutes per execution.
- **Google Workspace Account:** 30 minutes per execution. If your script hits this limit, it will be terminated. You need to optimize your code or break down long tasks into smaller chunks that can be run by triggers.

---

92. Why is my script running so slow?

Explanation: The number one cause of slow scripts is making too many calls to Google services inside a loop. For example, reading 100 cells one by one (`getValue()`) is 100 service calls. Reading them all at once (`getValues()`) is just one call. **Best Practice:** Read all your data into an array, process the array in your script, and then write all your results back to the sheet at once.

---

93. How can I debug my script?

Explanation:

1. **Logger.log():** The simplest way. Print variable values at different points to see what's happening.
2. **Debugger:** In the editor, you can set "breakpoints" by clicking on the line number. When you click the **"Debug"** button (instead of "Run"), the script will pause at the breakpoint, and you can inspect the values of all variables at that moment.

94. How do I use `Utilities.formatDate()`?

Explanation: This is a very useful utility for formatting JavaScript `Date` objects into strings in a specific format and timezone.

Code Example:

```
function formatMyDate() {
  const now = new Date();
  const timeZone = Session.getScriptTimeZone(); // e.g. "America/New_York"

  // Format as 'Year-Month-Day'
  const formattedDate = Utilities.formatDate(now, timeZone, 'yyyy-MM-dd');
  Logger.log(formattedDate);

  // Format as 'Hour:Minute:Second AM/PM'
  const formattedTime = Utilities.formatDate(now, timeZone, 'hh:mm:ss a');
  Logger.log(formattedTime);
}
```

---

95. What are advanced Google services?

Explanation: Some Google APIs (like the Google Analytics API or the YouTube API) are not enabled by default. You have to enable them in the script editor under the "Services" section (+). Once enabled, they appear in autocomplete just like standard services (`SpreadsheetApp`, etc.).

---

96. What is `clasp`?

Explanation: `clasp` is the **C**ommand **L**ine **A**pps **S**cript **P**rojects tool. It's an open-source tool from Google that lets you develop your Apps Script projects on your local computer using your own code editor (like VS Code). It allows you to use modern development tools like Git for version control.

---

97. How do I create an add-on?

Explanation: An add-on is a script that is published to the Google Workspace Marketplace so that other people can install and use it. It's a much more involved process than a simple script. It requires creating a Google Cloud Platform project, configuring an OAuth consent screen, and following specific publishing guidelines. It's the final step to distributing your script widely.

---

98. How do I work with files larger than 10MB?

Explanation: Certain Apps Script service methods have size limits (e.g., `getBlob()` might time out on very large files). For large files, especially when dealing with APIs, you may need to use resumable uploads or other strategies that are often handled by enabling Advanced Google Services, which use the underlying REST APIs directly.

---

99. What are script manifests?

Explanation: Every Apps Script project has a manifest file named `appsscript.json`. It's hidden by default, but you can view it by going to **Project Settings** and checking the box "Show 'appsscript.json' manifest file in editor". This file contains project metadata, like library dependencies, API scopes, and add-on configurations. You usually don't need to edit it manually unless you're doing advanced development.

---

100. Where can I find the official documentation?

Explanation: The official Google Apps Script documentation is the ultimate source of truth. It contains reference guides for every service and method, tutorials, and examples. You can find it at: https://developers.google.com/apps-script.