

Google Apps Script for Google Sheets: Comprehensive Guide



Google Apps Script for Google Sheets: Comprehensive Guide	1
What is Google Apps Script?	2
How to Use Google Apps Script in Sheets	2
Key Concepts in Google Sheets Apps Script	2
Basic Examples	2
Example 1: Write Data to a Sheet	2
Example 2: Read Data from a Sheet	3
Advanced Examples	3
Example 3: Add a New Sheet	3
Example 4: Create a Custom Menu	3
Working with Data	4
Example 5: Append Data to a Sheet	4
Example 6: Loop Through Rows	4
Exercises	4
Exercise 1: Write and Read Data	4
Solution:	5

Learn more HTML, CSS, JavaScript Web Development at <https://basescripts.com/> Laurence Svekis

Exercise 2: Create a Summary	5
Solution:	5
Exercise 3: Add a Timestamp	5
Solution:	5
Multiple-Choice Questions	6
Question 1:	6
Question 2:	6
Question 3:	6
Advanced Example: Create an Expense Tracker	6
Best Practices	7

Google Apps Script is a powerful tool for automating and extending the functionality of Google Sheets. This guide introduces key features, code examples, and exercises for working with Google Sheets using Apps Script.

What is Google Apps Script?

Google Apps Script is a cloud-based JavaScript platform for automating and enhancing Google Workspace applications, including Google Sheets. It allows you to perform tasks such as:

- Automating repetitive operations.
- Creating custom menus and functions.
- Integrating Google Sheets with external services.

How to Use Google Apps Script in Sheets

1. Open a Google Sheet.
2. Click **Extensions > Apps Script**.
3. Write your script in the Apps Script editor and save it.

Key Concepts in Google Sheets Apps Script

1. **SpreadsheetApp**: Accesses Google Sheets.
2. **Sheet**: Represents a single sheet within a spreadsheet.
3. **Range**: Represents a range of cells.
4. **Values**: Data within cells.

Basic Examples

Example 1: Write Data to a Sheet

```
function writeData() {
```

Learn more HTML, CSS, JavaScript Web Development at <https://basescripts.com/> Laurence Svekis

```
const sheet =  
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();  
sheet.getRange("A1").setValue("Hello, Apps Script!");  
}
```

Explanation:

- `SpreadsheetApp.getActiveSpreadsheet()`: Gets the active spreadsheet.
- `getActiveSheet()`: Selects the currently active sheet.
- `getRange("A1").setValue(...)`: Sets a value in cell A1.

Example 2: Read Data from a Sheet

```
function readData() {  
  const sheet =  
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();  
  const value = sheet.getRange("A1").getValue();  
  Logger.log(value); // Logs the value in A1  
}
```

Explanation:

- `getValue()`: Retrieves the value from the specified range.
- `Logger.log(...)`: Logs data for debugging.

Advanced Examples

Example 3: Add a New Sheet

```
function addNewSheet() {  
  const spreadsheet = SpreadsheetApp.getActiveSpreadsheet();  
  spreadsheet.insertSheet("New Sheet");  
}
```

Explanation:

- `insertSheet("New Sheet")`: Adds a new sheet named "New Sheet".

Example 4: Create a Custom Menu

```
function onOpen() {  
  const ui = SpreadsheetApp.getUi();  
  ui.createMenu("Custom Menu")  
}
```

Learn more HTML, CSS, JavaScript Web Development at <https://basescripts.com/> Laurence Svekis

```

        .addItem("Show Alert", "showAlert")
        .addToUi();
    }
function showAlert() {
    SpreadsheetApp.getUi().alert("Hello from Apps Script!");
}

```

Explanation:

- `createMenu("Custom Menu")`: Adds a custom menu to the toolbar.
- `addItem("Show Alert", "showAlert")`: Links a menu item to the `showAlert` function.

Working with Data

Example 5: Append Data to a Sheet

```

function appendData() {
    const sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
    sheet.appendRow(["Name", "Age", "Email"]);
}

```

Explanation:

- `appendRow([...])`: Appends an array as a new row at the bottom of the sheet.

Example 6: Loop Through Rows

```

function loopThroughRows() {
    const sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
    const data = sheet.getDataRange().getValues();
    for (let i = 0; i < data.length; i++) {
        Logger.log(`Row ${i + 1}: ${data[i].join(", ")}`);
    }
}

```

Explanation:

- `getDataRange().getValues()`: Retrieves all values in the sheet as a 2D array.

Exercises

Exercise 1: Write and Read Data

Write a script to:

1. Write "Hello, World!" in cell B2.
2. Read the value from B2 and log it.

Solution:

```
function writeAndRead() {
  const sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  sheet.getRange("B2").setValue("Hello, World!");
  const value = sheet.getRange("B2").getValue();
  Logger.log(value);
}
```

Exercise 2: Create a Summary

Write a script that calculates the sum of numbers in column A and displays it in cell B1.

Solution:

```
function calculateSum() {
  const sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  const data = sheet.getRange("A1:A").getValues().flat();
  const sum = data.reduce((total, num) => total + (parseFloat(num) ||
0), 0);
  sheet.getRange("B1").setValue(`Sum: ${sum}`);
}
```

Exercise 3: Add a Timestamp

Write a script to add the current date and time in column B whenever column A is updated.

Solution:

```
function onEdit(e) {
  const range = e.range;
  if (range.getColumn() === 1) {
    const sheet = range.getSheet();
```

Learn more HTML, CSS, JavaScript Web Development at <https://basescripts.com/> Laurence Svekis

```
    const row = range.getRow();
    sheet.getRange(row, 2).setValue(new Date());
  }
}
```

Multiple-Choice Questions

Question 1:

Which method is used to access the active spreadsheet?

1. `getSpreadsheet()`
2. `SpreadsheetApp.getActiveSpreadsheet()`
3. `SpreadsheetApp.open()`
4. `getActiveSheet()`

Answer: 2. `SpreadsheetApp.getActiveSpreadsheet()`

Question 2:

How do you append a row of data to a sheet?

1. `sheet.addRow()`
2. `sheet.appendData([...])`
3. `sheet.appendRow([...])`
4. `sheet.insertRow([...])`

Answer: 3. `sheet.appendRow([...])`

Question 3:

What does `getDataRange()` do?

1. Retrieves a specific cell range.
2. Retrieves all cells with data in the sheet.
3. Retrieves the last row with data.
4. Retrieves only numeric data.

Answer: 2. Retrieves all cells with data in the sheet.

Advanced Example: Create an Expense Tracker

```
function addExpense(date, description, amount) {
```

Learn more HTML, CSS, JavaScript Web Development at <https://basescripts.com/> Laurence Svekis

```
    const sheet =  
    SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();  
    sheet.appendRow([date, description, amount]);  
  }  
  function createExpenseTracker() {  
    const sheet =  
    SpreadsheetApp.getActiveSpreadsheet().insertSheet("Expenses");  
    sheet.appendRow(["Date", "Description", "Amount"]);  
    sheet.getRange("A1:C1").setFontWeight("bold");  
  }  
}
```

Explanation:

- `addExpense(date, description, amount)`: Appends a new expense entry.
- `createExpenseTracker()`: Sets up an "Expenses" sheet with formatted headers.

Best Practices

1. **Test your scripts:** Use the Logger or debug tools.
2. **Use comments:** Document complex scripts.
3. **Modularize your code:** Break larger scripts into functions.