



2. Getting Started	3
Accessing Google Apps Script	3
Your First Script	3
Understanding the Interface	4
3. Understanding the Basics	4
Syntax and Structure	4
Variables and Data Types	4
Functions and Control Structures	5
4. Working with Google Services	5
Google Sheets	5
Reading Data	5
Writing Data	6
Example: Sum of a Column	6
Google Docs	6
Inserting Text	6
Replacing Text	7
Gmail	7
Sending an Email	7
Reading Emails	7
5. Advanced Concepts	8
Triggers and Events	8
Time-driven Trigger	8
Event-driven Trigger	8
Publishing Scripts	9
Web App	9
Libraries and APIs	9
Using External APIs	9
Including Libraries	9
6. Best Practices	10
Debugging and Error Handling	10
Optimization	10
7. Quizzes and Exercises	10
Quiz Questions	10
Coding Exercises	11
Exercise 1: Create a Custom Function in Google Sheets	11
Exercise 2: Automate Email Reports	12
Additional Exercises	13
8. Conclusion	13

Welcome to your comprehensive guide on Google Apps Script! Whether you're a beginner or looking to deepen your understanding, this guide will provide you with everything you need to get started, along with examples, quizzes, and coding exercises to reinforce your learning.

## 1. Introduction to Google Apps Script

### What is Google Apps Script?

Google Apps Script is a cloud-based scripting language for light-weight application development in the G Suite platform. It is based on JavaScript and allows you to create scripts and applications that interact with Google Workspace applications like Google Sheets, Docs, Gmail, and more.

### Benefits and Uses

- Automation: Automate repetitive tasks across Google services.
- Customization: Create custom functions and menus.
- Integration: Integrate Google services with third-party APIs.
- Rapid Development: Build applications quickly with minimal setup.

### Prerequisites

- Basic understanding of JavaScript.
- A Google account to access Google Workspace applications.

## 2. Getting Started

### Accessing Google Apps Script

You can access Google Apps Script through several ways:

1. From Google Sheets, Docs, or Forms:
  - Open a document.
  - Navigate to Extensions > Apps Script.
2. Directly via the Apps Script Editor:
  - Go to [script.google.com](https://script.google.com).
  - Click on New Project.

### Your First Script

Let's create a simple script that shows an alert in a Google Docs document.

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

1. Open a Google Doc.
2. Go to Extensions > Apps Script.

Replace the default code with:

```
function showAlert() {  
  
DocumentApp.getUi().alert('Hello, World!');  
  
}
```

3. Save the script.
4. Go back to the document and refresh.
5. Go to Extensions > Apps Script > showAlert.

## Understanding the Interface

- Script Editor: Where you write your code.
- Project Settings: Configure your script's properties.
- Execution Log: View logs and errors.
- Libraries: Add external libraries.

## 3. Understanding the Basics

### Syntax and Structure

Apps Script uses modern JavaScript (ES6 and later). Here's a basic structure:

```
function functionName(parameters) {  
  
// code to execute  
  
}
```

### Variables and Data Types

- Variables: let, const, var
- Data Types: String, Number, Boolean, Array, Object

Example:

```
let greeting = 'Hello, World!';
```

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

```
const pi = 3.1415;

var isActive = true;
```

## Functions and Control Structures

- Functions: Blocks of code designed to perform a particular task.
- Control Structures: if, for, while, switch

Example:

```
function checkNumber(num) {

  if (num > 0) {

    return 'Positive';

  } else if (num < 0) {

    return 'Negative';

  } else {

    return 'Zero';

  }

}
```

## 4. Working with Google Services

Google Apps Script provides built-in services to interact with Google Workspace applications.

### Google Sheets

#### Reading Data

```
function readSheetData() {

  let sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();

  let data = sheet.getDataRange().getValues();

}
```

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

```
Logger.log(data);  
}
```

## Writing Data

```
function writeSheetData() {  
  let sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();  
  sheet.getRange('A1').setValue('Hello, Sheets!');  
}
```

## Example: Sum of a Column

```
function sumColumnA() {  
  let sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();  
  let values = sheet.getRange('A:A').getValues();  
  let sum = 0;  
  for (let i = 0; i < values.length; i++) {  
    if (typeof values[i][0] === 'number') {  
      sum += values[i][0];  
    }  
  }  
  Logger.log('Total Sum: ' + sum);  
}
```

## Google Docs

### Inserting Text

```
function insertText() {
```

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

```
let doc = DocumentApp.getActiveDocument();
let body = doc.getBody();
body.appendParagraph('This is a new paragraph. ');
}
```

## Replacing Text

```
function replaceText() {
let doc = DocumentApp.getActiveDocument();
let body = doc.getBody();
body.replaceText('old text', 'new text');
}
```

## Gmail

### Sending an Email

```
function sendEmail() {
let recipient = 'example@example.com';
let subject = 'Test Email';
let body = 'This is a test email sent from Apps Script.';
MailApp.sendEmail(recipient, subject, body);
}
```

### Reading Emails

```
function readEmails() {
let threads = GmailApp.getInboxThreads(0, 10);
for (let i = 0; i < threads.length; i++) {
```

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

```
let messages = threads[i].getMessages();
for (let j = 0; j < messages.length; j++) {
  Logger.log(messages[j].getSubject());
}
}
}
```

## 5. Advanced Concepts

### Triggers and Events

Triggers allow your script to execute automatically.

#### Time-driven Trigger

```
function createTimeDrivenTrigger() {
  ScriptApp.newTrigger('myScheduledFunction')
    .timeBased()
    .everyHours(1)
    .create();
}

function myScheduledFunction() {
  // Code to execute every hour
}
```

#### Event-driven Trigger

```
function onOpen(e) {
  SpreadsheetApp.getUi().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!');
}
```

## Publishing Scripts

You can publish your script as a web app or an add-on.

### Web App

```
function doGet(e) {

return HtmlService.createHtmlOutput('<h1>Hello, World!</h1>');

}
```

- Deploy via Publish > Deploy as web app.

## Libraries and APIs

### Using External APIs

```
function callExternalAPI() {

let response = UrlFetchApp.fetch('https://api.example.com/data');

let data = JSON.parse(response.getContentText());

Logger.log(data);

}
```

### Including Libraries

- Go to Resources > Libraries.
- Enter the Script ID of the library.

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

## 6. Best Practices

### Debugging and Error Handling

- Use `Logger.log()` to output values.
- Use `try...catch` blocks to handle errors.

```
function safeFunction() {  
  
  try {  
  
    // Code that may throw an error  
  
  } catch (error) {  
  
    Logger.log('Error: ' + error.message);  
  
  }  
  
}
```

### Optimization

- Minimize API calls.
- Cache results when possible.
- Use batch operations.

## 7. Quizzes and Exercises

### Quiz Questions

1. What function is used to send an email in Apps Script?

- A. `GmailApp.sendEmail()`
- B. `MailApp.sendEmail()`
- C. `EmailApp.send()`
- D. `MessageApp.sendEmail()`

<details><summary>Answer</summary>

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

Answer: B. MailApp.sendEmail()

</details>

2. How do you create a time-driven trigger that runs every day?

A. ScriptApp.newTrigger('functionName').timeBased().everyDays(1).create();

B. ScriptApp.createTrigger('functionName').daily().create();

C. ScriptApp.newTrigger('functionName').everyDay().create();

D. ScriptApp.timeTrigger('functionName').everyDays(1).create();

<details><summary>Answer</summary>

Answer: A. ScriptApp.newTrigger('functionName').timeBased().everyDays(1).create();

</details>

## Coding Exercises

### Exercise 1: Create a Custom Function in Google Sheets

Objective: Write a custom function that calculates the factorial of a number.

Instructions:

1. Open a new Google Sheets document.
2. Go to Extensions > Apps Script.
3. Write a function factorial(n) that returns the factorial of n.
4. Use the function in the sheet as =factorial(A1) where A1 contains a number.

Solution:

```
function factorial(n) {  
  
  if (n <= 1) return 1;  
  
  else return n * factorial(n - 1);  
  
}
```

Explanation:

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

- The function uses recursion to calculate the factorial.
- It can be called directly from the sheet.

## Exercise 2: Automate Email Reports

Objective: Send an email every Monday morning with the total number of rows in a Google Sheet.

Instructions:

1. Write a function `sendWeeklyReport()` that counts the rows.
2. Use `MailApp.sendEmail()` to send the email.
3. Set up a time-driven trigger to run every Monday at 8 AM.

Solution:

```
function sendWeeklyReport() {  
  
  let sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();  
  
  let numRows = sheet.getLastRow();  
  
  let recipient = 'example@example.com';  
  
  let subject = 'Weekly Report';  
  
  let body = 'The total number of rows is ' + numRows;  
  
  MailApp.sendEmail(recipient, subject, body);  
  
}  
  
function createWeeklyTrigger() {  
  
  ScriptApp.newTrigger('sendWeeklyReport')  
  
    .timeBased()  
  
    .onWeekDay(ScriptApp.WeekDay.MONDAY)  
  
    .atHour(8)  
  
    .create();  
  
}
```

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

```
}
```

Explanation:

- `sendWeeklyReport()` counts the rows and sends an email.
- `createWeeklyTrigger()` sets up the trigger.

## Additional Exercises

1. **Modify Text in Google Docs:** Write a script that finds all instances of a specific word in a Google Doc and highlights them.
2. **Batch Update in Google Sheets:** Update a range of cells with values from an array in one operation.
3. **Custom Menu:** Add a custom menu in Google Sheets that, when clicked, formats the selected cells to a specific style.

## 8. Conclusion

You've now learned the fundamentals of Google Apps Script, from basic syntax to interacting with Google Workspace applications. With practice, you'll be able to automate tasks, integrate services, and build powerful applications.

Next Steps:

- Explore the Apps Script documentation for more detailed information.
- Experiment with building custom add-ons.
- Join the Google Apps Script community for support and collaboration.