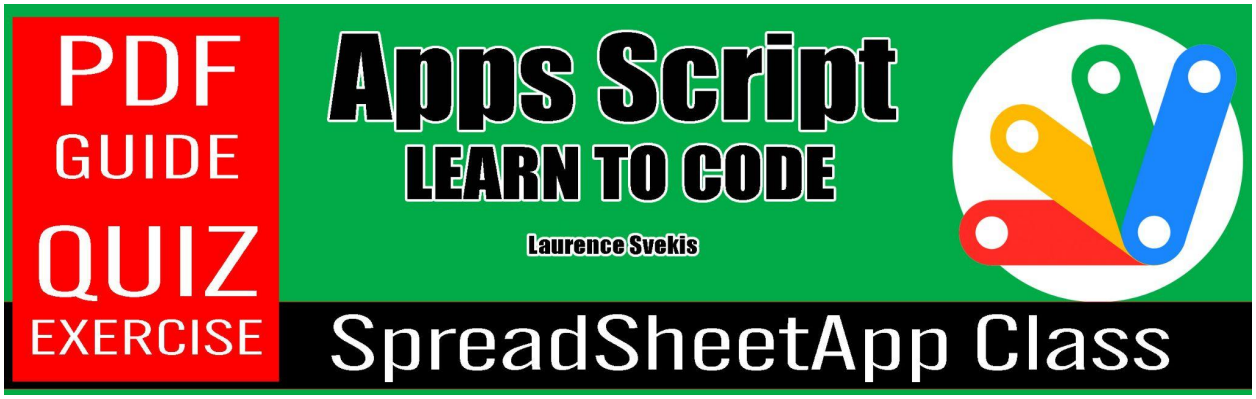
Google Apps Script SpreadSheetsApp class



Basic Example: **2**
Key Functionalities: **3**
Reading and Writing Data: 3
Formatting Cells: 3
Working with Formulas: 3
Manipulating Sheets: 3
Advanced Operations: 3
Google Apps Script - SpreadsheetApp Class **4**
1. Reading and Writing Data: 4
2. Formatting Cells: 4
3. Working with Formulas: 5
4. Manipulating Sheets: 5
5. Advanced Operations: 5
Coding Exercises **6**
Exercise 1: Read Data 6
Exercise 2: Write Data 7
Exercise 3: Format Cells 7
Exercise 4: Use Formulas 8
Exercise 5: Insert a Sheet 8

Exercise 6: Delete a Sheet 9
Exercise 7: Sort Data 10
Exercise 8: Create a Chart 10
Exercise 9: Advanced Formatting 11
Exercise 10: Use Triggers 12
Quiz Questions: **13**
Advanced Coding Exercise: Analyzing Sales Data **26**
Instructions for the Exercise: 31

Google Apps Script - sheetsapp Class

The spreadsheetsapp class is specifically designed to work with Google Sheets. It offers a range of methods that allow developers to perform operations such as reading and writing data, formatting cells, creating charts, and more.

Basic Example:

```
// Open the active spreadsheet
var spreadsheet =
SpreadsheetApp.getActiveSpreadsheet();

// Access the active sheet
var sheet = spreadsheet.getActiveSheet();

// Example: Set the value of cell A1
sheet.getRange('A1').setValue('Hello, Google Sheets!');
```

Key Functionalities:

Reading and Writing Data:

getRange(range): Retrieves a range of cells.

getValue(): Gets the value of a cell or a range of cells.

setValue(value): Sets the value of a cell or a range of cells.

Formatting Cells:

setBackgroundColor(color): Sets the background color of a cell or range of cells.

setFontColor(color): Sets the font color of a cell or range of cells.

setFontWeight(weight): Sets the font weight (bold) of a cell or range of cells.

Working with Formulas:

setFormula(formula): Sets a formula for a cell.

Manipulating Sheets:

insertSheet(sheetName): Inserts a new sheet into the spreadsheet.

deleteSheet(sheet): Deletes a specified sheet.

Advanced Operations:

sort(range, column, ascending): Sorts a range of cells based on a specified column.

createChart(type): Creates a new chart in the sheet.

Google Apps Script - SpreadsheetApp Class

The SpreadsheetApp class is the main class for interacting with Google Sheets in Google Apps Script. Here are some common code examples:

1. Reading and Writing Data:

```
// Open the active spreadsheet
var spreadsheet = SpreadsheetApp.getActiveSpreadsheet();

// Access the active sheet
var sheet = spreadsheet.getActiveSheet();

// Example: Get the value of cell A1
var cellValue = sheet.getRange('A1').getValue();

// Example: Set the value of cell B1
sheet.getRange('B1').setValue('New Value');
```

2. Formatting Cells:

```
// Example: Set the background color of cell A1 to yellow
sheet.getRange('A1').setBackground('yellow');
```

```
// Example: Set the font color of cell B1 to red
sheet.getRange('B1').setFontColor('red');

// Example: Make cell C1 bold
sheet.getRange('C1').setFontWeight('bold');
```

3. Working with Formulas:

```
// Example: Set a formula in cell A2 to sum values in A1 and B1
sheet.getRange('A2').setFormula('=SUM(A1:B1)');
```

4. Manipulating Sheets:

```
// Example: Insert a new sheet
spreadsheet.insertSheet('New Sheet');

// Example: Delete a sheet
var sheetToDelete = spreadsheet.getSheetByName('SheetToDelete');
spreadsheet.deleteSheet(sheetToDelete);
```

5. Advanced Operations:

```
// Example: Sort data in column A in ascending order
var rangeToSort = sheet.getRange('A2:A100');
rangeToSort.sort({column: 1, ascending: true});
```

// Example: Create a bar chart

var chart = sheet.newChart().asBarChart().addRange(rangeToSort).setPosition(1, 1, 0, 0).build();

sheet.insertChart(chart);

Note:

Ensure that you have the necessary permissions to access and modify the Google Sheet.

Coding Exercises

Exercise 1: Read Data

Task: Read data from a specific cell.

Steps:

1. Open a Google Sheet.
2. In the script editor, write a script to read the value from cell A1.
3. Log the value to the console.

Code:

```
function readData() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  var cellValue = sheet.getRange('A1').getValue();
```

```
  Logger.log('Value in A1: ' + cellValue);
}
```

Exercise 2: Write Data

Task: Write data to a specific cell.

Steps:

1. Open a Google Sheet.
2. In the script editor, write a script to set the value of cell B2 to "Hello, Google Apps Script!"

Code:

```
function writeData() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  sheet.getRange('B2').setValue('Hello, Google Apps
Script!');
}
```

Exercise 3: Format Cells

Task: Change the background color of a cell.

Steps:

1. Open a Google Sheet.
2. In the script editor, write a script to set the background color of cell C3 to blue.

Code:

```
function formatCell() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  sheet.getRange('C3').setBackground('blue');
}
```

Exercise 4: Use Formulas

Task: Set a formula in a cell.

Steps:

1. Open a Google Sheet.
2. In the script editor, write a script to set a formula in cell D4 to multiply values in A4 and B4.

Code:

```
function setFormula() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  sheet.getRange('D4').setFormula('=A4*B4');
}
```

Exercise 5: Insert a Sheet

Task: Insert a new sheet.

Steps:

1. Open a Google Sheet.
2. In the script editor, write a script to insert a new sheet named "NewSheet."

Code:

```
function insertSheet() {
  var spreadsheet =
SpreadsheetApp.getActiveSpreadsheet();
  spreadsheet.insertSheet('NewSheet');
}
```

Exercise 6: Delete a Sheet

Task: Delete a sheet.

Steps:

1. Open a Google Sheet.
2. In the script editor, write a script to delete a sheet named "SheetToDelete."

Code:

```
function deleteSheet() {
  var spreadsheet =
SpreadsheetApp.getActiveSpreadsheet();
  var sheetToDelete =
spreadsheet.getSheetByName('SheetToDelete');
  spreadsheet.deleteSheet(sheetToDelete);
}
```

Exercise 7: Sort Data

Task: Sort data in a column.

Steps:

1. Open a Google Sheet.
2. In the script editor, write a script to sort data in column E in ascending order.

Code:

```
function sortData() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  var rangeToSort = sheet.getRange('E:E');
  rangeToSort.sort({column: 1, ascending: true});
}
```

Exercise 8: Create a Chart

Task: Create a bar chart.

Steps:

1. Open a Google Sheet.
2. In the script editor, write a script to create a bar chart based on data in column F.

Code:

```
function createChart() {
```

```
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  var rangeForChart = sheet.getRange('F:F');
  var chart =
sheet.newChart().asBarChart().addRange(rangeForChart).s
etPosition(1, 1, 0, 0).build();
  sheet.insertChart(chart);
}
```

Exercise 9: Advanced Formatting

Task: Apply advanced formatting to cells.

Steps:

1. Open a Google Sheet.
2. In the script editor, write a script to set font color, background color, and bold style for cells in range G1:H5.

Code:

```
function advancedFormatting() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  var rangeToFormat = sheet.getRange('G1:H5');

rangeToFormat.setFontColor('green').setBackground('yell
ow').setFontWeight('bold');
```

```
}
```

Exercise 10: Use Triggers

Task: Set up a trigger to run a function periodically.

Steps:

1. Open a Google Sheet.
2. In the script editor, write a script to log a message to the console.
3. Set up a trigger to run this function every day.

Code:

```
function logMessage() {
  Logger.log('Triggered! Hello from Google Apps
Script.');
}

// Set up a trigger to run logMessage every day at 8 AM
function setUpTrigger() {

ScriptApp.newTrigger('logMessage').timeBased().everyDay
s(1).atHour(8).create();
}
```

Quiz Questions:

Question: What is the primary class for interacting with Google Sheets in Google Apps Script?

A) SheetApp

B) SpreadsheetApp

C) GoogleSheetApp

D) AppsScriptSheet

Answer: B) SpreadsheetApp

Question: How do you get the active sheet in a spreadsheet using SpreadsheetApp?

A) getActiveSheet()

B) activeSheet()

C) getCurrentSheet()

D) retrieveActiveSheet()

Answer: A) getActiveSheet()

Question: Which method is used to read the value of a specific cell in Google Sheets?

A) getCell()

B) readCell()

C) getValue()

D) retrieveValue()

Answer: C) getValue()

Question: How can you set the value of a cell in Google Sheets using Google Apps Script?

A) updateValue()

B) setValue()

C) setCell()

D) writeValue()

Answer: B) setValue()

Question: Which method is used to set the background color of a cell in Google Sheets?

A) setBackground()

B) setColor()

C) setCellColor()

D) fillColor()

Answer: A) setBackground()

Question: What function is used to set a formula in a cell in Google Sheets?

A) setFormula()

B) formulaCell()

C) writeFormula()

D) applyFormula()

Answer: A) setFormula()

Question: In Google Apps Script, how do you insert a new sheet into a spreadsheet?

A) createSheet()

B) addSheet()

C) insertSheet()

D) newSheet()

Answer: C) insertSheet()

Question: To delete a sheet in Google Sheets using Google Apps Script, you need to provide the sheet's _____.

A) Name

B) Index

C) ID

D) Both A and B

Answer: D) Both A and B

Question: How can you sort data in a specific column in ascending order using Google Apps Script?

A) sortAscending()

B) orderAsc()

C) sort({column: 1, ascending: true})

D) arrangeAscending()

Answer: C) sort({column: 1, ascending: true})

Question: Which method is used to create a bar chart in Google Sheets using Google Apps Script?

A) newChart().asBarChart()

B) createBarChart()

C) drawBarChart()

D) generateBarChart()

Answer: A) newChart().asBarChart()

Question: What is the purpose of the timeBased() method in setting up triggers in Google Apps Script?

A) It sets the timezone for the trigger.

B) It schedules the trigger to run at specific times.

C) It specifies the time interval between trigger runs.

D) It triggers an event based on the system clock.

Answer: B) It schedules the trigger to run at specific times.

Question: Which method is used to log information to the console in Google Apps Script?

A) writeLog()

B) console.log()

C) Logger.log()

D) logMessage()

Answer: C) Logger.log()

Question: In Google Apps Script, how do you set the font color of a cell?

A) setCellFontColor()

B) setFontColor()

C) cell.setFontColor()

D) changeFontColor()

Answer: B) setFontColor()

Question: What is the purpose of the everyDays(1) method when setting up a trigger for a daily event?

A) It specifies the number of days the trigger runs.

B) It sets the trigger to run every day.

C) It defines the time interval between trigger runs.

D) It triggers an event every 24 hours.

Answer: B) It sets the trigger to run every day.

Question: Which of the following is the correct way to apply both bold style and background color to a range of cells?

A) setBoldAndBackgroundColor()

B) setBold().setBackgroundColor()

C) setFontWeight('bold').setBackground('color')

D) formatBoldAndColor()

Answer: C) setFontWeight('bold').setBackground('color')

Question: What function is used to add a range of cells to a chart in Google Apps Script?

A) addDataRange()

B) setChartRange()

C) addRange()

D) includeData()

Answer: C) addRange()

Question: How do you specify the position of a chart when inserting it into a sheet using Google Apps Script?

A) setPosition(x, y)

B) setChartPosition(x, y)

C) chartPosition(x, y)

D) positionChart(x, y)

Answer: A) setPosition(x, y)

Question: In Google Apps Script, what does the build() method do when working with charts?

A) It finalizes and builds the chart.

B) It creates a new chart object.

C) It adds the chart to the spreadsheet.

D) It initializes the chart properties.

Answer: A) It finalizes and builds the chart.

Question: What is the purpose of the atHour(8) method when setting up a trigger?

A) It specifies the number of hours between trigger runs.

B) It sets the trigger to run at the specified hour.

C) It triggers an event every 8 hours.

D) It adjusts the timezone for the trigger.

Answer: B) It sets the trigger to run at the specified hour.

Question: Which method is used to set the formula in a range of cells in Google Sheets?

A) setFormulas()

B) applyFormula()

C) setRangeFormula()

D) range.setFormula()

Answer: D) range.setFormula()

Question: How do you retrieve the name of the active sheet in Google Apps Script?

A) getSheetName()

B) activeSheet.getName()

C) getCurrentSheetName()

D) sheetName()

Answer: B) activeSheet.getName()

Question: What is the purpose of the clear() method when applied to a range in Google Apps Script?

A) It removes formatting from the range.

B) It clears the content of the range.

C) It deletes the range.

D) It hides the range.

Answer: B) It clears the content of the range.

Question: How do you specify the font size of a cell in Google Apps Script?

A) setFontSize(size)

B) fontSize(size)

C) setFontHeight(size)

D) changeFontSize(size)

Answer: C) setFontHeight(size)

Question: Which of the following methods is used to copy data from one range to another in Google Sheets using Google Apps Script?

A) copyRangeTo()

B) range.copyTo(targetRange)

C) duplicateRange()

D) pasteRange()

Answer: B) range.copyTo(targetRange)

Question: What does the getValues() method return when applied to a range in Google Apps Script?

A) The formulas in the range.

B) The values of the cells in the range.

C) The formatting of the cells in the range.

D) The background color of the cells in the range.

Answer: B) The values of the cells in the range.

Question: How do you determine the number of rows in a range using Google Apps Script?

A) range.getRows()

B) rows(range)

C) rowCount(range)

D) getRowCount(range)

Answer: C) rowCount(range)

Question: What is the purpose of the getBackgrounds() method when applied to a range in Google Apps Script?

A) It retrieves the formulas of the cells in the range.

B) It returns the background colors of the cells in the range.

C) It provides the values of the cells in the range.

D) It gets the font colors of the cells in the range.

Answer: B) It returns the background colors of the cells in the range.

Question: How do you merge cells in a range in Google Apps Script?

A) range.merge()

B) mergeCells(range)

C) mergeRange()

D) mergeCellsIn(range)

Answer: A) range.merge()

Question: Which method is used to freeze rows in Google Sheets using Google Apps Script?

A) freezeRows()

B) setFrozenRows()

C) freezePane()

D) rows.freeze()

Answer: C) freezePane()

Question: How do you insert a new row above a specific row in Google Sheets using Google Apps Script?

A) insertRowAbove(row)

B) addRowAbove(row)

C) insertAbove(row)

D) insertRowsAbove(row)

Answer: A) insertRowAbove(row)

Question: What is the purpose of the getActiveRange() method in Google Apps Script?

A) It returns the currently selected range in the spreadsheet.

B) It provides information about the active sheet.

C) It gets the range of the entire sheet.

D) It retrieves the last modified range.

Answer: A) It returns the currently selected range in the spreadsheet.

Question: How do you set the text alignment of a range in Google Apps Script?

A) alignText()

B) setTextAlignment()

C) setAlignment()

D) textAlignment()

Answer: C) setAlignment()

Question: Which method is used to hide a specific column in Google Sheets using Google Apps Script?

A) hideColumn()

B) setColumnHidden()

C) column.hide()

D) hideSheetColumn()

Answer: B) setColumnHidden()

Question: How do you set the protection for a range in Google Sheets using Google Apps Script?

A) setProtection()

B) protectRange()

C) rangeProtection()

D) applyProtection()

Answer: B) protectRange()

Question: In Google Apps Script, what is the purpose of the getActiveSpreadsheet() method?

A) It opens a new spreadsheet.

B) It retrieves information about the active spreadsheet.

C) It creates a copy of the active spreadsheet.

D) It returns the currently selected spreadsheet.

Answer: B) It retrieves information about the active spreadsheet.

Question: How do you add a note to a specific cell in Google Sheets using Google Apps Script?

A) addNote()

B) setCellNote()

C) range.setNote()

D) noteRange()

Answer: C) range.setNote()

Question: What is the purpose of the getActiveCell() method in Google Apps Script?

A) It returns the currently selected cell in the spreadsheet.

B) It retrieves the active sheet in the spreadsheet.

C) It gets the range of the entire sheet.

D) It provides information about the active script.

Answer: A) It returns the currently selected cell in the spreadsheet.

Question: How do you create a hyperlink in a cell using Google Apps Script?

A) createLink()

B) setHyperlink()

C) hyperlinkCell()

D) addLink()

Answer: B) setHyperlink()

Question: In Google Apps Script, what method is used to resize a column in Google Sheets?

A) resizeColumn()

B) setColumnWidth()

C) columnWidth()

D) changeColumnSize()

Answer: B) setColumnWidth()

Advanced Coding Exercise: Analyzing Sales Data

Problem Statement:

You are given a Google Sheet that contains sales data for a company. The sheet has the following columns:

Column A: Product ID

Column B: Product Name

Column C: Units Sold

Column D: Unit Price

Table

Product ID	Product Name	Units Sold	Unit Price
1	Product A	100	20
2	Product B	150	15
3	Product C	80	30
4	Product D	120	25
5	Product E	200	18

You need to write a Google Apps Script that performs the following tasks:

1. Calculate the total revenue for each product (Units Sold * Unit Price).
2. Calculate the total revenue for the entire dataset.
3. Identify the product with the highest total revenue.
4. Write the calculated values (total revenue for each product, total revenue for the dataset, and the product with the highest revenue) to a new summary sheet.

Solution:

```
function analyzeSalesData() {
  // Get the active spreadsheet
  var spreadsheet =
SpreadsheetApp.getActiveSpreadsheet();

  // Access the sales data sheet
  var salesSheet =
spreadsheet.getSheetByName('SalesData');

  // Ensure the SalesData sheet exists
  if (!salesSheet) {
    Logger.log('SalesData sheet not found.');
    return;
  }
```

```
  // Create a new summary sheet
  var summarySheet =
spreadsheet.getSheetByName('Summary');

  // If Summary sheet doesn't exist, create one
  if (!summarySheet) {
    summarySheet = spreadsheet.insertSheet('Summary');
  }

  // Clear existing data in the summary sheet
  summarySheet.clear();

  // Get data range excluding header row
  var dataRange = salesSheet.getDataRange().offset(1,
0, salesSheet.getLastRow() - 1,
salesSheet.getLastColumn());

  // Get values from the data range
  var dataValues = dataRange.getValues();

  // Initialize variables for calculations
  var totalRevenue = 0;
  var productWithMaxRevenue = { name: '', revenue: 0 };
```

```
  var productRevenues = {};

  // Perform calculations
  dataValues.forEach(function (row) {
    var unitsSold = row[2];
    var unitPrice = row[3];

    // Calculate revenue for each product
    var productRevenue = unitsSold * unitPrice;
    totalRevenue += productRevenue;

    // Update product revenues
    var productId = row[0];
    var productName = row[1];
    productRevenues[productId] =
productRevenues[productId] ? productRevenues[productId]
+ productRevenue : productRevenue;

    // Update product with max revenue
    if (productRevenues[productId] >
productWithMaxRevenue.revenue) {
      productWithMaxRevenue.name = productName;
```

```
      productWithMaxRevenue.revenue =
productRevenues[productId];
    }
  });

  // Write results to the summary sheet
  summarySheet.getRange(1, 1).setValue('Total Revenue
for Each Product');
  summarySheet.getRange(2, 1,
Object.keys(productRevenues).length, 2).setValues(
    Object.entries(productRevenues).map(([productId,
revenue]) => [productId, revenue])
  );

  summarySheet.getRange(1, 4).setValue('Total Revenue
for the Dataset');
  summarySheet.getRange(2, 4).setValue(totalRevenue);

  summarySheet.getRange(1, 6).setValue('Product with
Highest Revenue');
  summarySheet.getRange(2,
6).setValue(productWithMaxRevenue.name);
```

```
  summarySheet.getRange(2,
7).setValue(productWithMaxRevenue.revenue);

  Logger.log('Analysis completed. Results written to
Summary sheet.');
}
```

Instructions for the Exercise:

1. Create a Google Sheet with the name "SalesData" and populate it with sample sales data.
2. Open the script editor in Google Sheets (Extensions -> Apps Script).
3. Paste the provided solution into the script editor.
4. Save the script, and then run the analyzeSalesData function.
5. Check the "Summary" sheet in your spreadsheet for the calculated results.

This exercise covers reading data from a sheet, performing calculations, and updating another sheet with the results. Feel free to modify the exercise or add additional features to make it more complex based on your learning goals.