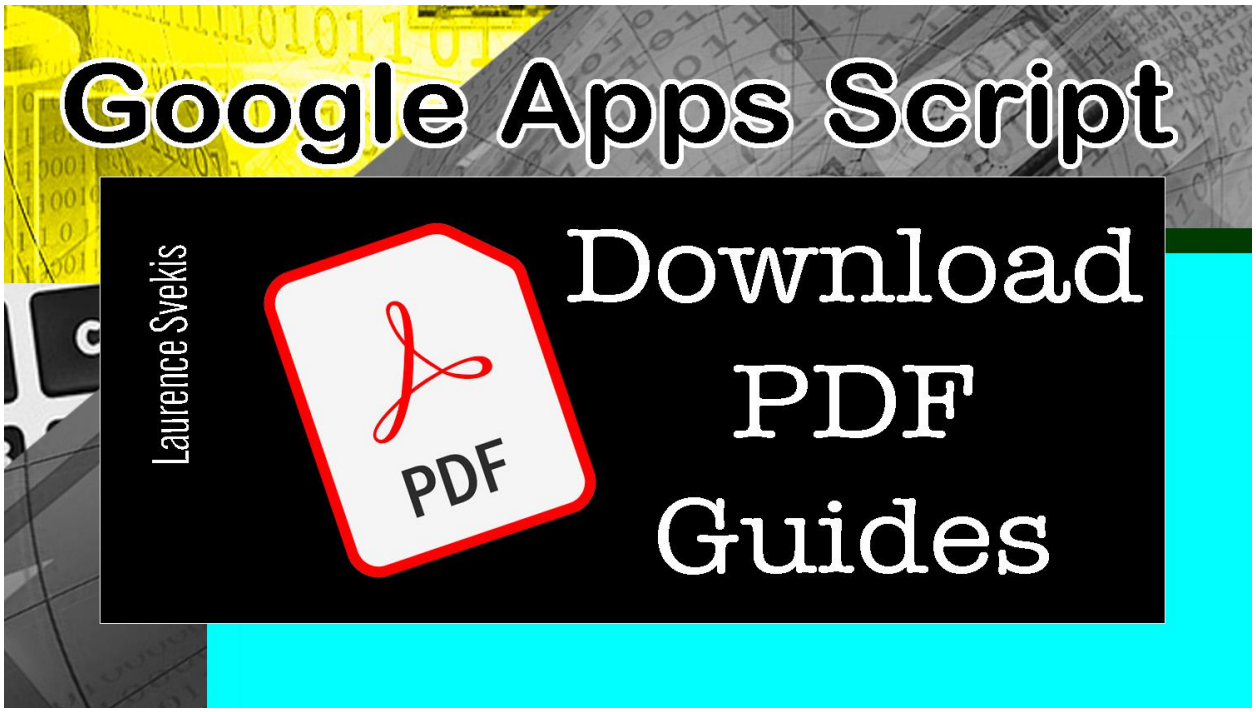
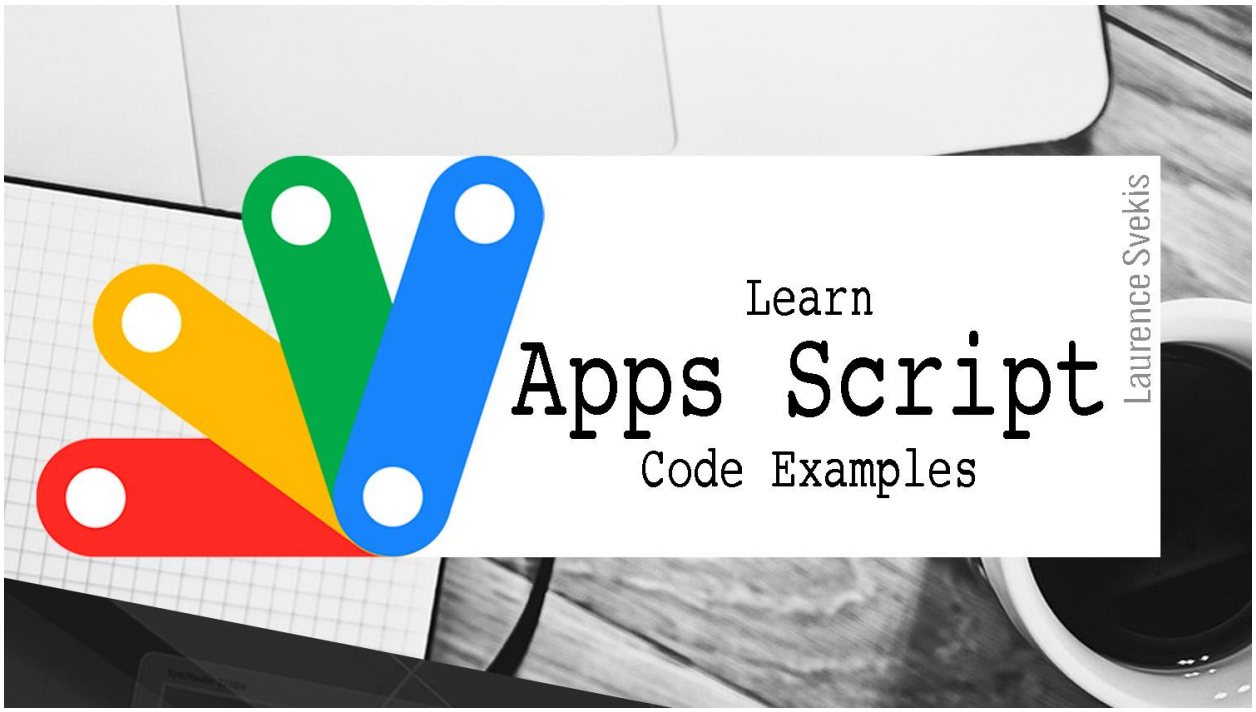
Apps Script Guide March 2023



Apps Script Fun Coding Exercises 1



Count Cells Function

Create a function that takes a spreadsheet as a parameter and returns the total number of cells with values in it.

```
function countCells(spreadsheet) {
  var sheet =
SpreadsheetApp.openById(spreadsheet).getActiveSheet();
  var range = sheet.getDataRange();
```

```
  var values = range.getValues();
  var count = 0;
  for (var i = 0; i < values.length; i++) {
    for (var j = 0; j < values[0].length; j++) {
      if (values[i][j] != "") {
        count++;
      }
    }
  }
  return count;
}
```

Get Sheet Names Function

Create a function that takes a spreadsheet as a parameter and returns an array of all the sheet names in the spreadsheet.

```
function getSheetNames(spreadsheet) {
  var sheets =
SpreadsheetApp.openById(spreadsheet).getSheets();
  var sheetNames = [];
  for (var i = 0; i < sheets.length; i++) {
    sheetNames.push(sheets[i].getName());
  }
  return sheetNames;
}
```

Count Rows in Sheet

Create a function that takes a spreadsheet and a sheet name as parameters and returns the total number of rows in that sheet.

```
function countRows(spreadsheet, sheetName) {
  var sheet =
SpreadsheetApp.openById(spreadsheet).getSheetByName(she
etName);
  var range = sheet.getDataRange();
  var numRows = range.getNumRows();
  return numRows;
}
```

Get values from Columns

Create a function that takes a spreadsheet and a sheet name as parameters and returns an array of all the values in the first column of that sheet.

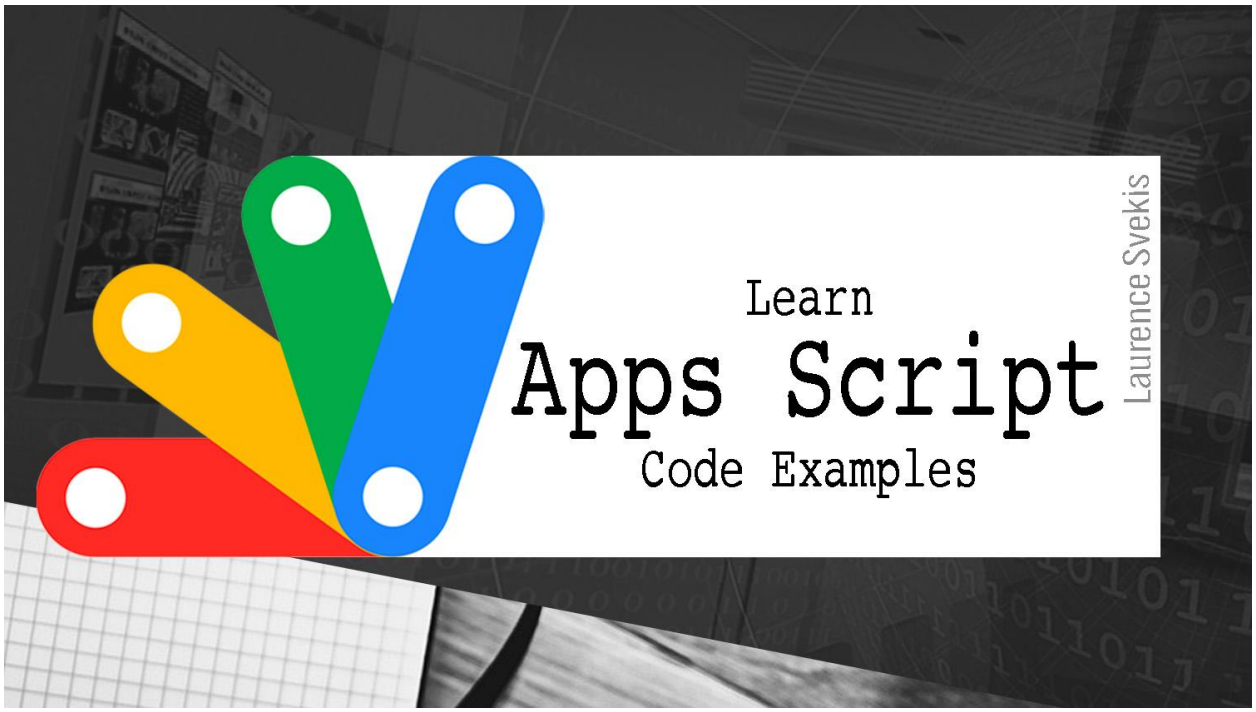
```
function getColumnValues(spreadsheet, sheetName) {
  var sheet =
SpreadsheetApp.openById(spreadsheet).getSheetByName(she
etName);
  var range = sheet.getDataRange();
  var values = range.getValues();
  var columnValues = [];
  for (var i = 0; i < values.length; i++) {
    columnValues.push(values[i][0]);
  }
  return columnValues;
}
```

Get Column Values by Letter

Create a function that takes a spreadsheet, a sheet name, and a column letter (e.g., "A", "B", "C") as parameters and returns an array of all the values in that column.

```
function getColumnValuesByLetter(spreadsheet,
sheetName, columnLetter) {
  var sheet =
SpreadsheetApp.openById(spreadsheet).getSheetByName(she
etName);
  var range = sheet.getRange(columnLetter + ":" +
columnLetter);
  var values = range.getValues();
  var columnValues = [];
  for (var i = 0; i < values.length; i++) {
    columnValues.push(values[i][0]);
  }
  return columnValues;
}
```

Apps Script Fun Coding Exercises 2



Sending Email Reminders

Create a script that sends an email reminder to yourself every day at a specific time.

```
function sendEmailReminder() {
  var emailAddress = "yourEmailAddress";
  var subject = "Daily Reminder";
```

```
  var message = "Don't forget to complete your daily
tasks!";
  MailApp.sendEmail(emailAddress, subject, message);
}

function scheduleEmailReminder() {
  ScriptApp.newTrigger("sendEmailReminder")
    .timeBased()
    .everyDays(1)
    .atHour(9)
    .create();
}
```

Send Google Form Confirmations

Create a script that automatically sends an email when a Google Form is submitted.

```
function sendConfirmationEmail(e) {
  var email = e.namedValues["Email"][0];
  var subject = "Thank you for submitting the form!";
  var message = "Thank you for your submission. We will
get back to you soon.";
  MailApp.sendEmail(email, subject, message);
}

function createFormTrigger() {
  var form = FormApp.getActiveForm();
  ScriptApp.newTrigger("sendConfirmationEmail")
    .forForm(form)
    .onFormSubmit()
```

```
    .create();
}
```

Generate Unique ID values

Create a script that automatically generates a unique ID for a new row in a Google Sheet.

```
function generateUniqueId() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  var range = sheet.getRange(sheet.getLastRow(), 1);
  var id = Utilities.getUuid();
  range.setValue(id);
}
```

Deleting Specific Rows

Create a script that deletes all the rows in a Google Sheet that have a specific value in a certain column.

```
function deleteRowsWithValue(value, column) {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  var data = sheet.getDataRange().getValues();
  var newData = [];
  for (var i = 0; i < data.length; i++) {
    if (data[i][column - 1] !== value) {
      newData.push(data[i]);
    }
```

```
  }
  sheet.getDataRange().clearContent();
  sheet.getRange(1, 1, newData.length,
newData[0].length).setValues(newData);
}
```

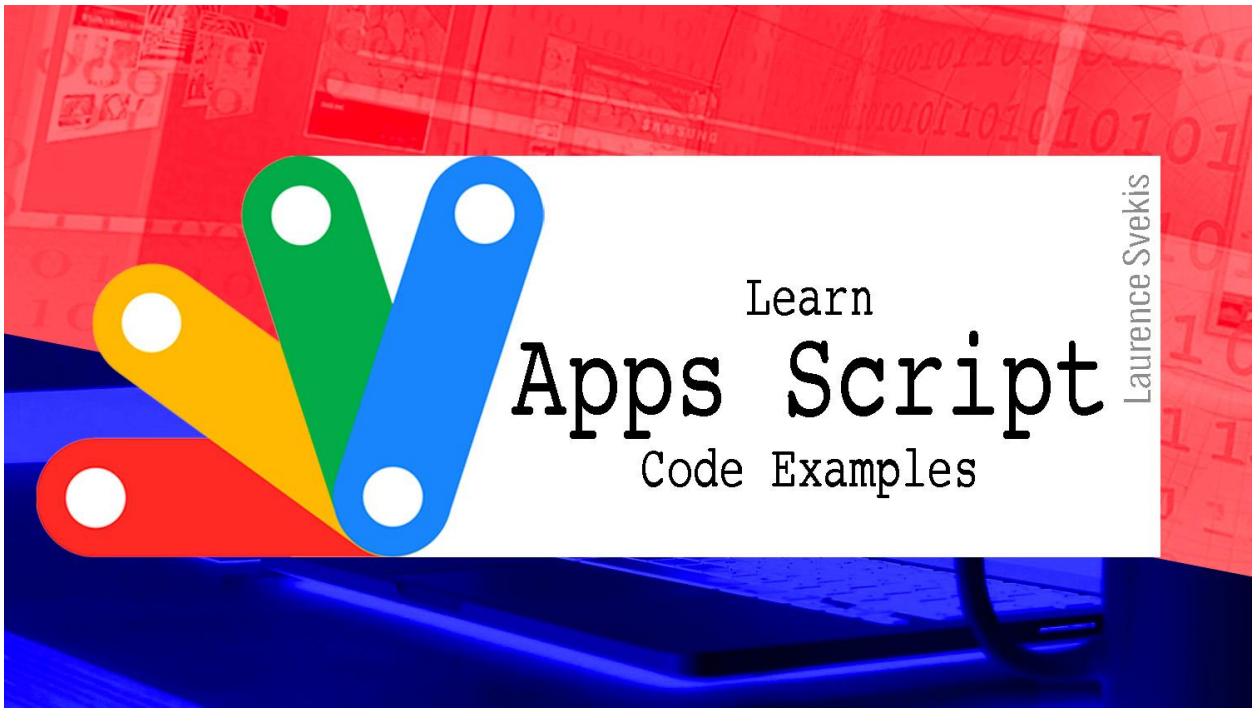
Sort Sheet Data

Create a script that automatically sorts a Google Sheet based on the values in a specific column.

```
function sortSheet() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  var range = sheet.getRange(2, 1, sheet.getLastRow() -
1, sheet.getLastColumn());
  range.sort({ column: 2, ascending: true });
}
```

Note: These are just basic examples of what you can do with Google Apps Script. You can modify these scripts and build upon them to make more complex and useful applications.

Apps Script Fun Coding Exercises 3



How to send a reminder email

Create a script that automatically sends a reminder email to team members who haven't completed a task in a Google Sheet.

```
function sendReminderEmails() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
```

```
  var data = sheet.getDataRange().getValues();
  var today = new Date();
  for (var i = 1; i < data.length; i++) {
    var dueDate = new Date(data[i][2]);
    var daysDiff = Math.floor((today - dueDate) /
86400000);
    if (daysDiff >= 2 && data[i][3] == "") {
      var recipient = data[i][1];
      var subject = "Reminder: Task is overdue";
      var message = "Please complete the task as soon
as possible.";
      MailApp.sendEmail(recipient, subject, message);
    }
  }
}
```

Add to Sheet new email data

Create a script that adds a new row to a Google Sheet when a new email is received in a specific Gmail label.

```
function onGmailMessage(e) {
  var label = GmailApp.getUserLabelByName("Label
Name");
  if (label && e.labelIds.indexOf(label.getId()) != -1)
{
    var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
    sheet.appendRow([e.subject, e.getBody()]);
  }
}
```

How to create a PDF from Sheet

Create a script that automatically generates a PDF file and saves it to Google Drive when a new row is added to a Google Sheet.

```
function createPDF() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  var range = sheet.getRange("A1:C10");
  var blob =
range.getBlob().getAs("application/pdf").setName("file.
pdf");
  var folder = DriveApp.getFolderById("folderId");
  var file = folder.createFile(blob);
}
```

Add new Calendar event

Create a script that automatically adds a new event to a Google Calendar when a new row is added to a Google Sheet.

```
function createCalendarEvent(e) {
  var title = e.namedValues["Title"][0];
  var start = new Date(e.namedValues["Start"][0]);
  var end = new Date(e.namedValues["End"][0]);
  var calendar = CalendarApp.getDefaultCalendar();
  calendar.createEvent(title, start, end);
}

function createSheetTrigger() {
```

```
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  ScriptApp.newTrigger("createCalendarEvent")
    .forSpreadsheet(sheet)
    .onFormSubmit()
    .create();
}
```

How to create a new doc

Create a script that automatically creates a new Google Doc when a new row is added to a Google Sheet and populates it with data from the row.

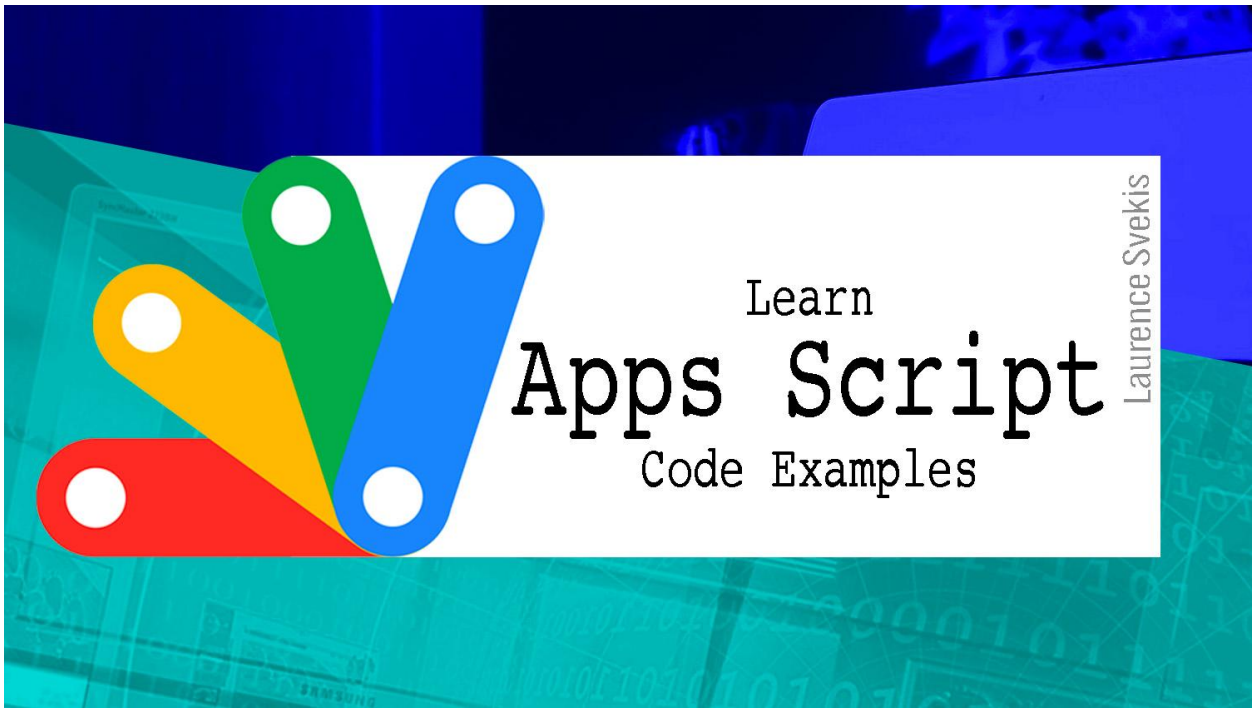
```
function createNewDoc(e) {
  var title = e.namedValues["Title"][0];
  var body = e.namedValues["Body"][0];
  var folder = DriveApp.getFolderById("folderId");
  var newDoc = DocumentApp.create(title);
  var newFile = DriveApp.getFileById(newDoc.getId());
  folder.addFile(newFile);
  newFile.getParents().next().removeFile(newFile);
  newDoc.getBody().setText(body);
}

function createSheetTrigger() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  ScriptApp.newTrigger("createNewDoc")
    .forSpreadsheet(sheet)
```

```
      .onFormSubmit()
      .create();
}
```

Note: These are just basic examples of what you can do with Google Apps Script. You can modify these scripts and build

Apps Script Fun Coding Exercises 4



[Adding a UI menu option]
[How to create a new sheet]
[Sending Custom emails from data]
[Add image button to Doc]
[Creating calendar events and trigger]

Adding a UI menu option

Create a script that adds a custom menu to a Google Sheet and runs a function when a menu item is clicked.

```
function onOpen() {
  var ui = SpreadsheetApp.getUi();
  ui.createMenu("Custom Menu")
    .addItem("Function 1", "function1")
    .addItem("Function 2", "function2")
    .addToUi();
}

function function1() {
  // code to run when menu item "Function 1" is clicked
}

function function2() {
  // code to run when menu item "Function 2" is clicked
}
```

How to create a new sheet

Create a script that automatically creates a new Google Sheet and copies data from a template sheet.

```
function createNewSheet() {
  var template =
DriveApp.getFileById("templateSheetId");
  var folder = DriveApp.getFolderById("folderId");
  var newFile = template.makeCopy("New Sheet", folder);
```

```
  var newSheet =
SpreadsheetApp.openById(newFile.getId()).getSheets()[0]
;
  var data =
SpreadsheetApp.getActiveSpreadsheet().getDataRange().ge
tValues();
  newSheet.getRange(1, 1, data.length,
data[0].length).setValues(data);
}
```

Sending Custom emails from data

Create a script that sends a customized email to each recipient in a Google Sheet.

```
function sendCustomizedEmails() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  var data = sheet.getDataRange().getValues();
  for (var i = 1; i < data.length; i++) {
    var recipient = data[i][0];
    var subject = "Hello, " + data[i][1] + "!";
    var message = "This is a customized message for
you.";
    MailApp.sendEmail(recipient, subject, message);
  }
}
```

Add image button to Doc

Create a script that adds a new row to a Google Sheet when a button is clicked in a Google Doc.

```
function addButtonToDocument() {
  var doc = DocumentApp.getActiveDocument();
  var body = doc.getBody();
  var button = body.appendInlineImage(
    UrlFetchApp

.fetch("https://www.gstatic.com/images/icons/material/system/1x/add_white_24dp.png")
      .getBlob()
      .setName("Add Row")
  );
  button.setLinkUrl("javascript:addRowToSheet()");
}

function addRowToSheet() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Sheet1");
  sheet.appendRow(["new data"]);
}
```

Creating calendar events and trigger

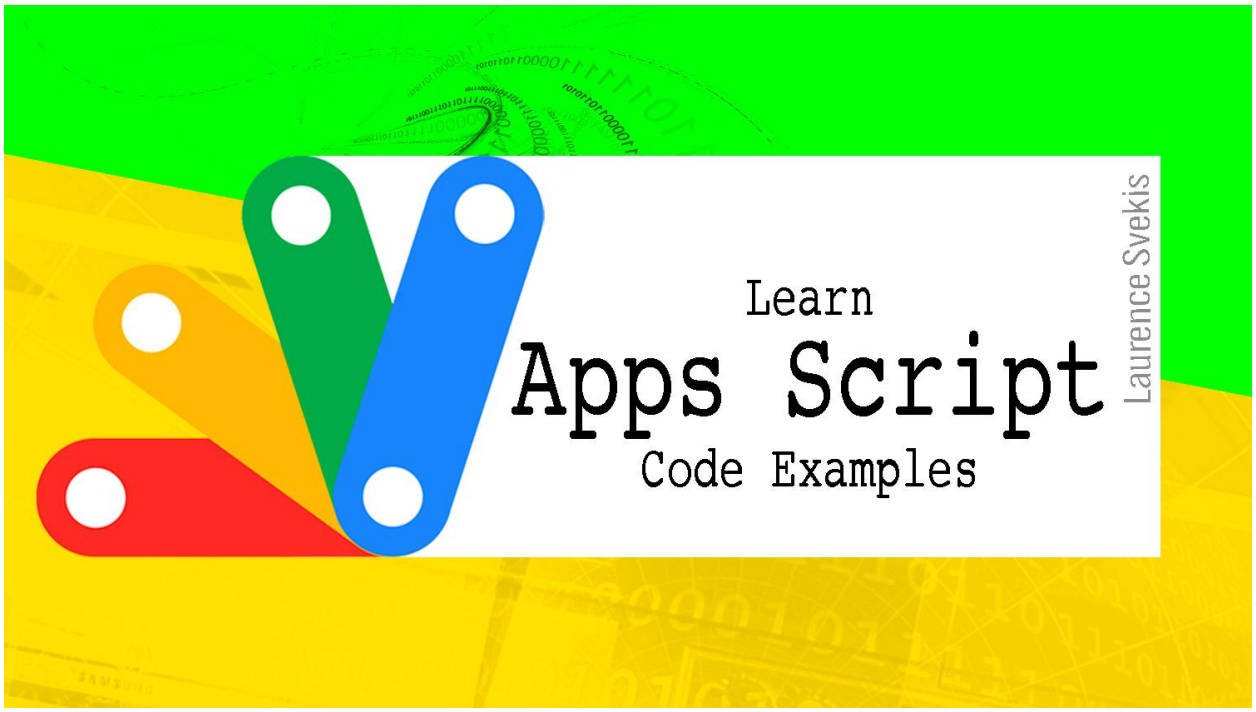
Create a script that automatically creates a new Google Calendar event when a new row is added to a Google Sheet.

```
function createCalendarEvent(e) {
```

```
  var title = e.namedValues["Title"][0];
  var start = new Date(e.namedValues["Start"][0]);
  var end = new Date(e.namedValues["End"][0]);
  var calendar = CalendarApp.getDefaultCalendar();
  calendar.createEvent(title, start, end);
}

function createSheetTrigger() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  ScriptApp.newTrigger("createCalendarEvent")
    .forSpreadsheet(sheet)
    .onFormSubmit()
    .create();
}
```

Apps Script Fun Coding Exercises 5



Imports data from CSV file

Create a script that automatically imports data from a CSV file and populates a Google Sheet.

```
function importCSV() {
  var csvUrl = "https://example.com/data.csv";
  var csvContent =
UrlFetchApp.fetch(csvUrl).getContentText();
  var data = Utilities.parseCsv(csvContent);
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  sheet.getRange(1, 1, data.length,
data[0].length).setValues(data);
```

```
}
```

Create a New Form

Create a script that automatically creates a new Google Form and populates it with questions from a Google Sheet.

```
function createNewForm() {
  var title = "Form Title";
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  var headers = sheet.getRange(1, 1, 1,
sheet.getLastColumn()).getValues()[0];
  var questions = [];
  for (var i = 1; i < headers.length; i++) {
    var question = {title: headers[i], type:
FormApp.ItemType.TEXT};
    questions.push(question);
  }
  var form = FormApp.create(title);
  var formId = form.getId();
  var items = form.addItems(questions);
  var url = form.getPublishedUrl();
  sheet.getRange(2, 1).setValue(formId);
  sheet.getRange(2, 2).setValue(url);
}
```

Create a PDF from Doc

Create a script that automatically converts a Google Doc to PDF and saves it to Google Drive when a new row is added to a Google Sheet.

```
function createPDF() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  var fileId = sheet.getRange(sheet.getLastRow(),
1).getValue();
  var doc = DocumentApp.openById(fileId);
  var blob =
doc.getAs("application/pdf").setName(doc.getName() +
".pdf");
  var folder = DriveApp.getFolderById("folderId");
  var file = folder.createFile(blob);
}

function createSheetTrigger() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  ScriptApp.newTrigger("createPDF")
    .forSpreadsheet(sheet)
    .onEdit()
    .create();
}
```

Send customized Emails

Create a script that automatically sends a customized email to multiple recipients using data from a Google Sheet.

```
function sendCustomizedEmails() {
```

```
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  var data = sheet.getRange(2, 1, sheet.getLastRow() -
1, sheet.getLastColumn()).getValues();
  var template =
HtmlService.createTemplateFromFile("email_template");
  for (var i = 0; i < data.length; i++) {
    var recipient = data[i][0];
    var subject = "Customized Subject Line";
    template.recipient = recipient;
    var html = template.evaluate().getContent();
    MailApp.sendEmail({
      to: recipient,
      subject: subject,
      htmlBody: html,
      attachments: []
    });
  }
}
```

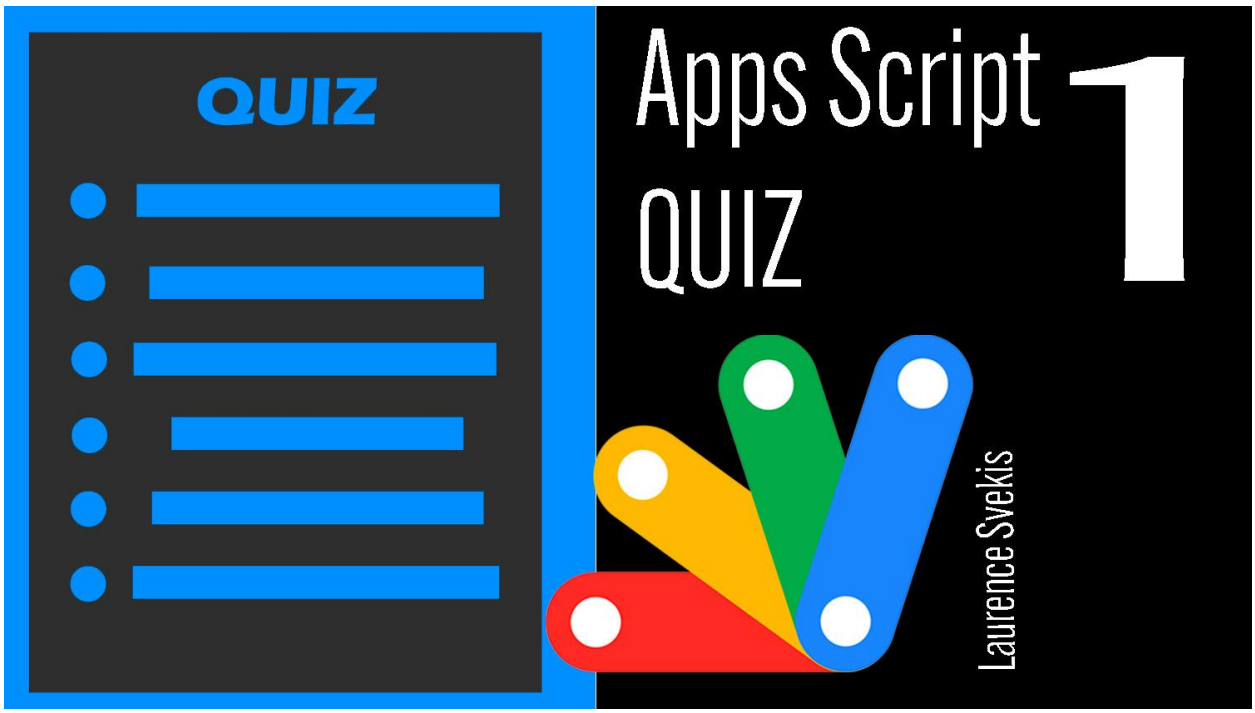
In this script, we first retrieve data from a Google Sheet and store it in a variable called data. We then create an HTML email template using the HtmlService class.

Inside the loop, we retrieve the recipient's email address from the data variable, set the subject line, and populate the template with the recipient's name using template.recipient = recipient;. We then evaluate the template and retrieve its HTML content using template.evaluate().getContent();.

Finally, we send the email using the MailApp.sendEmail() method, passing in the recipient's email address, subject line, HTML content, and any attachments as parameters.

Note that in order to use the MailApp class, you'll need to have authorized the script to send emails by going to File > Project properties > Scopes and adding the https://www.googleapis.com/auth/gmail.send scope.

Google Apps Script Quiz 1



[What is the output of the following code snippet?]

[What is the output of the following code snippet?]

[What does the following code snippet do?]

[What is the output of the following code snippet?]

What is the output of the following code snippet?

```
var x = 10;
var y = "5";
console.log(x + y);
```

a) 510

b) 15

c) 105

d) TypeError

Solution: b

What is the output of the following code snippet?

```
function myFunction() {
  var name = "John";
  Logger.log(name);
}
myFunction();
```

a) John

b) undefined

c) null

d) ReferenceError

Solution: a

What does the following code snippet do?

```
function add(a, b) {
  return a + b;
}
```

a) multiplies two numbers

b) subtracts two numbers

c) adds two numbers

d) divides two numbers

Solution: c

What is the output of the following code snippet?

```
var x = 10;
var y = "5";
console.log(x - y);
```

a) 5

b) 15

c) 105

d) TypeError

Solution: a

What is the correct way to define a function in Google Apps Script?

a) functionName = function() {}

b) function functionName() {}

c) functionName() = function() {}

d) None of the above

Solution: b

What is the output of the following code snippet?

```
var x = 10;
var y = 5;
console.log(x * y);
```

a) 50

b) 5

c) 2

d) 15

Solution: a

What does the following code snippet do?

```
function multiply(a, b) {
  return a * b;
}
```

a) multiplies two numbers

b) subtracts two numbers

c) adds two numbers

d) divides two numbers

Solution: a

What is the output of the following code snippet?

```
var x = 10;
var y = 5;
console.log(x / y);
```

a) 2

b) 50

c) 5

d) 0.5

Solution: d

What is the correct way to create a new sheet in Google Sheets using Google Apps Script?

a) var sheet = new Sheet();

b) var sheet = SpreadsheetApp.create("Sheet1");

c) var sheet = SpreadsheetApp.getActiveSheet();

d) None of the above

Solution: b

What does the following code snippet do?

```
function myFunction() {
  var x = 10;
  var y = 5;
  var z = x + y;
  return z;
}
```

a) adds two numbers and returns the result

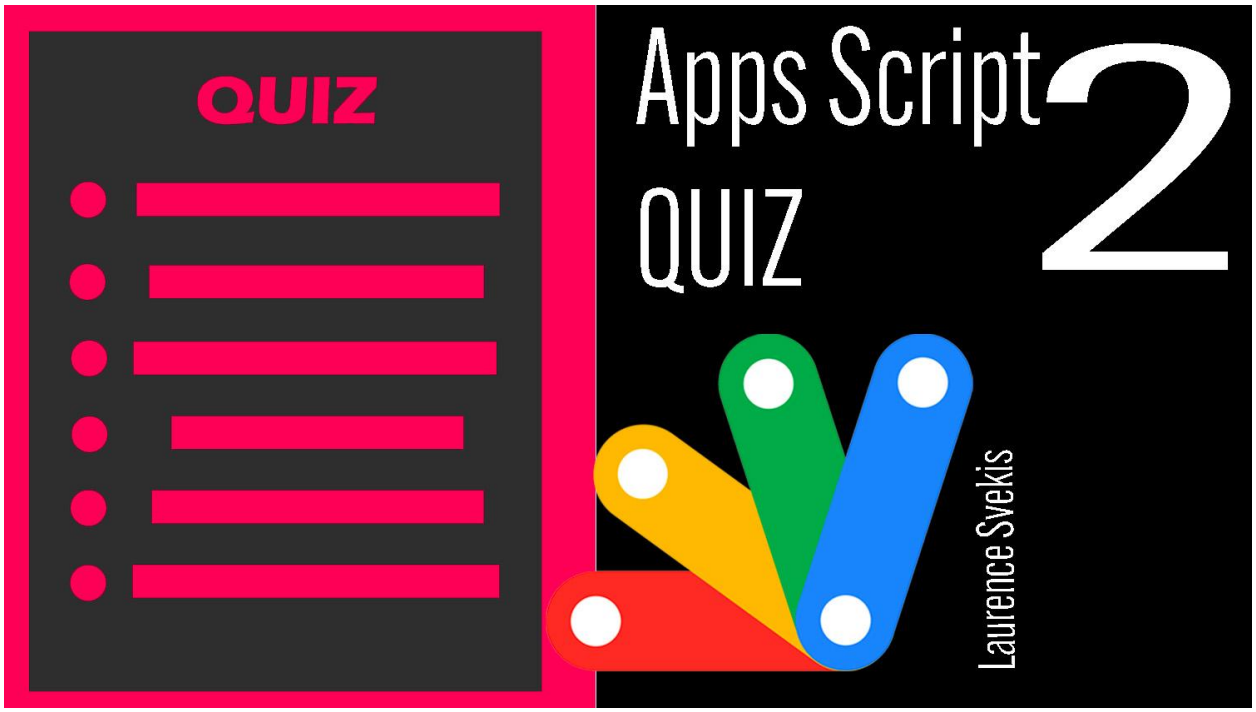
b) subtracts two numbers and returns the result

c) multiplies two numbers and returns the result

d) divides two numbers and returns the result

Solution: a

Google Apps Script Quiz 2



[Which method is used to read the value of a cell in Google Sheets?]

[Which object is used to create a new file in Google Drive?]

[Which method is used to send an email in Google Apps Script?]

[Which method is used to get the current date and time in Google Apps Script?]

[Which method is used to add a new row to a Google Sheets spreadsheet?]

[Which method is used to get the value of a URL parameter in Google Apps Script?]

[Which method is used to get the current user's email address in Google Apps Script?]

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

a. setValue()

b. getValue()

c. getRange()

d. getValues()

Solution: b. getValue()

Which object is used to create a new file in Google Drive?

a. Drive

b. Spreadsheet

c. DocumentApp

d. File

Solution: d. File

Which method is used to send an email in Google Apps Script?

a. MailApp.sendEmail()

b. GmailApp.sendEmail()

c. EmailApp.sendEmail()

d. SendMail.send()

Solution: b. GmailApp.sendEmail()

Which method is used to get the current date and time in Google Apps Script?

a. new Date()

b. getDateTime()

c. getCurrentTime()

d. getTime()

Solution: a. new Date()

Which method is used to add a new row to a Google Sheets spreadsheet?

a. appendRow()

b. addRow()

c. insertRow()

d. createRow()

Solution: a. appendRow()

Which method is used to get the value of a URL parameter in Google Apps Script?

a. getParameter()

b. getUrlParameter()

c. getQueryString()

d. getParameterByName()

Solution: d. getParameterByName()

Which method is used to get the current user's email address in Google Apps Script?

a. getUserEmail()

b. getEmail()

c. getCurrentUserEmail()

d. getActiveUserEmail()

Solution: b. getEmail()

Which method is used to create a new calendar event in Google Apps Script?

a. createEvent()

b. addEvent()

c. insertEvent()

d. newEvent()

Solution: a. createEvent()

Which object is used to access the user's Google Contacts in Google Apps Script?

a. Contacts

b. ContactsApp

c. ContactsService

d. GoogleContacts

Solution: b. ContactsApp

Which method is used to get the value of a checkbox in a Google Sheets cell?

a. isChecked()

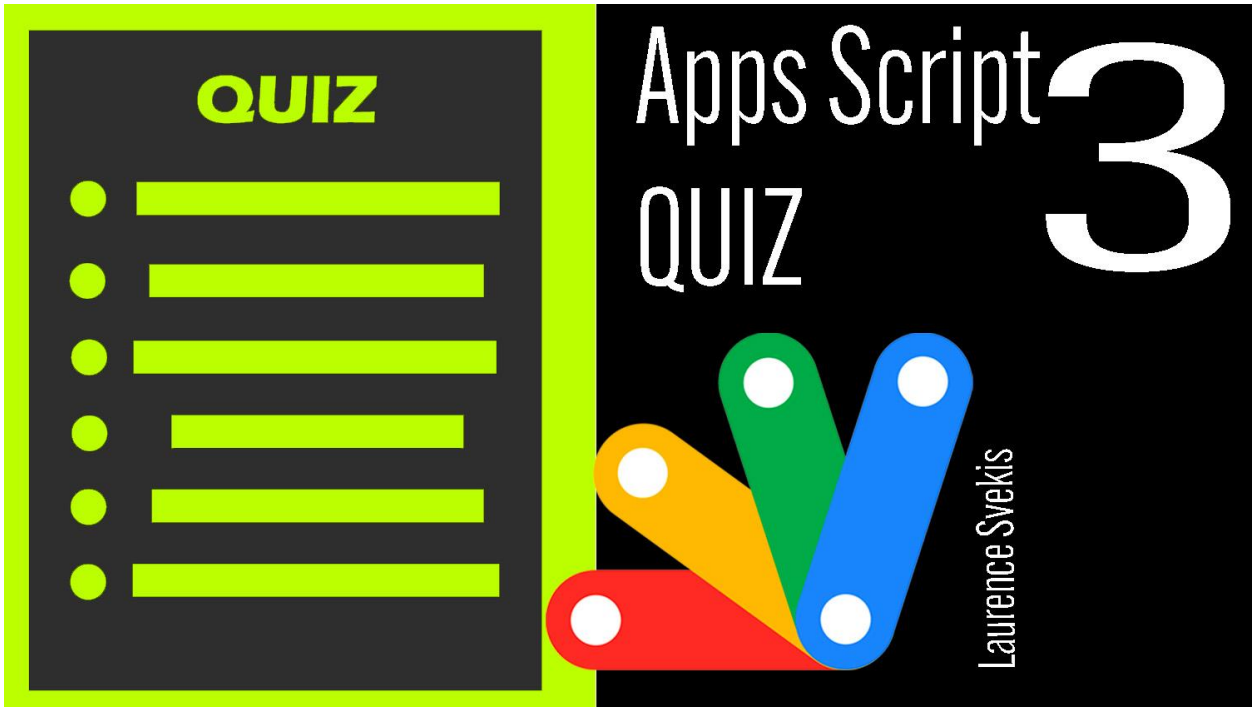
b. getChecked()

c. getCheckboxValue()

d. getValue()

Solution: d. getValue()

Google Apps Script Quiz 3



[Which of the following is a data type in Google Apps Script?]

Which of the following is a data type in Google Apps Script?

a) Integer

b) String

c) Boolean

d) All of the above

Solution: d) All of the above

Which method is used to create a new sheet in a Google Sheets document using Apps Script?

a) createSheet()

b) addSheet()

c) newSheet()

d) sheetCreate()

Solution: b) addSheet()

Which operator is used to concatenate strings in Apps Script?

a) +

b) *

c) /

d) -

Solution: a) +

Which method is used to retrieve the value of a cell in Google Sheets using Apps Script?

a) getCellValue()

b) getCell()

c) getValue()

d) retrieveCellValue()

Solution: c) getValue()

Which method is used to delete a row in a Google Sheets document using Apps Script?

a) deleteRow()

b) removeRow()

c) eraseRow()

d) eliminateRow()

Solution: a) deleteRow()

Which method is used to create a new file in Google Drive using Apps Script?

a) createFile()

b) newFile()

c) addFile()

d) makeFile()

Solution: a) createFile()

Which method is used to send an email using Apps Script?

a) sendEmail()

b) emailSend()

c) mailSend()

d) send()

Solution: a) sendEmail()

Which method is used to get the current date and time in Apps Script?

a) getCurrentTime()

b) getCurrentDate()

c) new Date()

d) getDateAndTime()

Solution: c) new Date()

Which method is used to get the user's email address in Apps Script?

a) getUserEmail()

b) getEmail()

c) getCurrentUserEmail()

d) Session.getActiveUser().getEmail()

Solution: d) Session.getActiveUser().getEmail()

Which method is used to create a menu in a Google Sheets document using Apps Script?

a) createMenu()

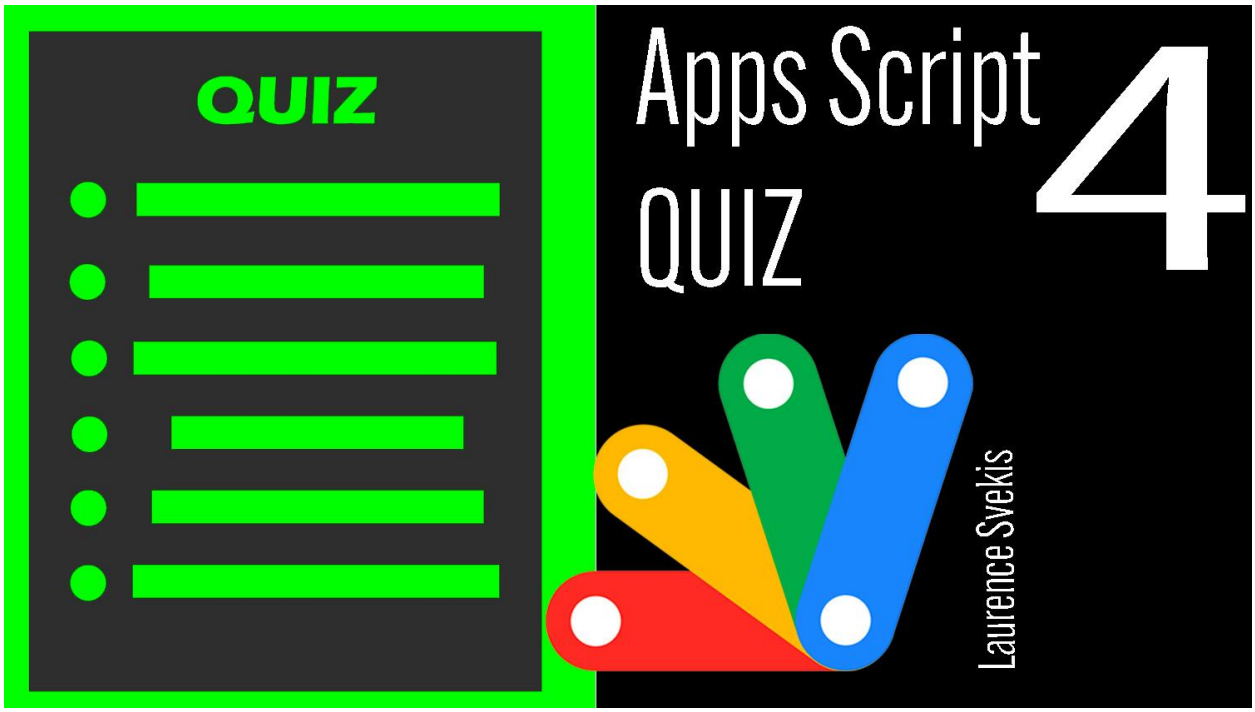
b) addMenu()

c) newMenu()

d) insertMenu()

Solution: b) addMenu()

Google Apps Script Quiz 4



Question 1: Which of the following is not a data type in Google Apps Script?

A) String

B) Boolean

C) Integer

D) Double

Solution: C) Integer

Question 2: What is the output of the following code?

```
var x = 10;
var y = 5;
var z = x + y;
Logger.log(z);
```

A) 10

B) 5

C) 15

D) Error

Solution: C) 15

Question 3: What is the correct syntax for defining a function in Google Apps Script?

A) function myFunction() {}

B) def myFunction() {}

C) var myFunction() {}

D) function = myFunction() {}

Solution: A) function myFunction() {}

Question 4: Which of the following is not a valid loop structure in Google Apps Script?

A) for loop

B) while loop

C) do-while loop

D) foreach loop

Solution: D) foreach loop

Question 5: What is the output of the following code?

```
var x = "5";
var y = 10;
var z = x + y;
Logger.log(z);
```

A) 15

B) 510

C) Error

D) undefined

Solution: B) 510

Question 6: Which of the following methods can be used to convert a string to a number in Google Apps Script?

A) Number()

B) parseInt()

C) parseFloat()

D) All of the above

Solution: D) All of the above

Question 7: What is the output of the following code?

```
var x = true;
var y = false;
Logger.log(x && y);
```

A) true

B) false

C) Error

D) undefined

Solution: B) false

Question 8: Which of the following is not a valid comparison operator in Google Apps Script?

A) ==

B) !=

C) <=

D) ><

Solution: D) ><

Question 9: What is the output of the following code?

```
var x = [1, 2, 3];
Logger.log(x.length);
```

A) 3

B) "1, 2, 3"

C) Error

D) undefined

Solution: A) 3

Question 10: Which of the following is not a valid way to declare an array in Google Apps Script?

A) var arr = [1, 2, 3];

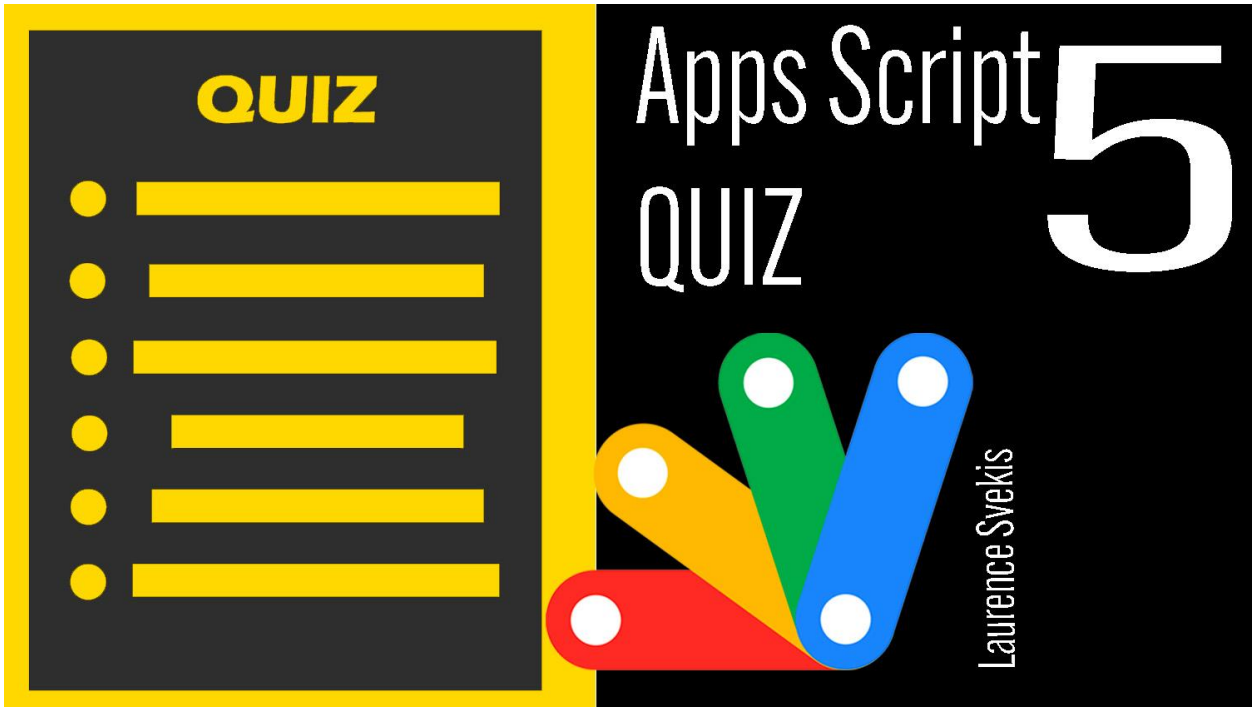
B) var arr = new Array(1, 2, 3);

C) var arr = {1, 2, 3};

D) All of the above are valid

Solution: C) var arr = {1, 2, 3};

Google Apps Script Quiz 5



Which of the following is not a data type in Google Sheets?

a. String

b. Integer

c. Boolean

d. Double

Solution: b. Integer. Google Sheets does not have a data type specifically for integers. Numeric values are stored as floating point numbers, which can include decimals.

Explanation: Google Sheets supports several data types, including strings (text), numbers (integers and decimals), booleans (true/false values), dates, and times.

What function can be used to count the number of cells in a range that meet a specific condition?

a. COUNT

b. COUNTA

c. COUNTIF

d. SUMIF

Solution: c. COUNTIF. The COUNTIF function allows you to count the number of cells in a range that meet a specific condition.

Explanation: The COUNT function counts the number of cells in a range that contain numerical values. The COUNTA function counts the number of cells in a range that are not empty. The SUMIF function adds up the values in a range that meet a specific condition.

What is the syntax for a FOR loop in Apps Script?

a. for (var i = 0; i < 10; i++)

b. for i = 0 to 10

c. for (i = 0; i <= 10; i++)

d. for (i = 0; i < 10; i++)

Solution: d. for (i = 0; i < 10; i++). This is the correct syntax for a FOR loop in Apps Script.

Explanation: A FOR loop is used to execute a block of code a specified number of times. The syntax for a FOR loop in Apps Script is similar to other programming languages: for (initialization; condition; increment) { code to be executed }.

Which of the following methods can be used to add a new sheet to a Google Sheets spreadsheet?

a. addSheet()

b. createSheet()

c. insertSheet()

d. all of the above

Solution: d. all of the above. All three of these methods can be used to add a new sheet to a Google Sheets spreadsheet.

Explanation: The addSheet() method adds a new sheet at the end of the spreadsheet. The createSheet() method adds a new sheet at a specific index, and can also set the sheet name and properties. The insertSheet() method adds a new sheet at a specific location, such as before or after an existing sheet.

Which of the following methods can be used to get the value of a cell in a Google Sheets spreadsheet?

a. getCell()
b. getValue()
c. getRange()
d. all of the above

Solution: b. getValue(). The getValue() method can be used to get the value of a single cell in a Google Sheets spreadsheet.

Explanation: The getCell() method returns a Range object that represents a single cell. The getRange() method returns a Range object that represents a range of cells. The getValue() method

can be used with either a single cell or a range of cells to get the value of the cell or cells.

What does the following code do? var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Sheet1");

a. Creates a new spreadsheet with one sheet named "Sheet1".

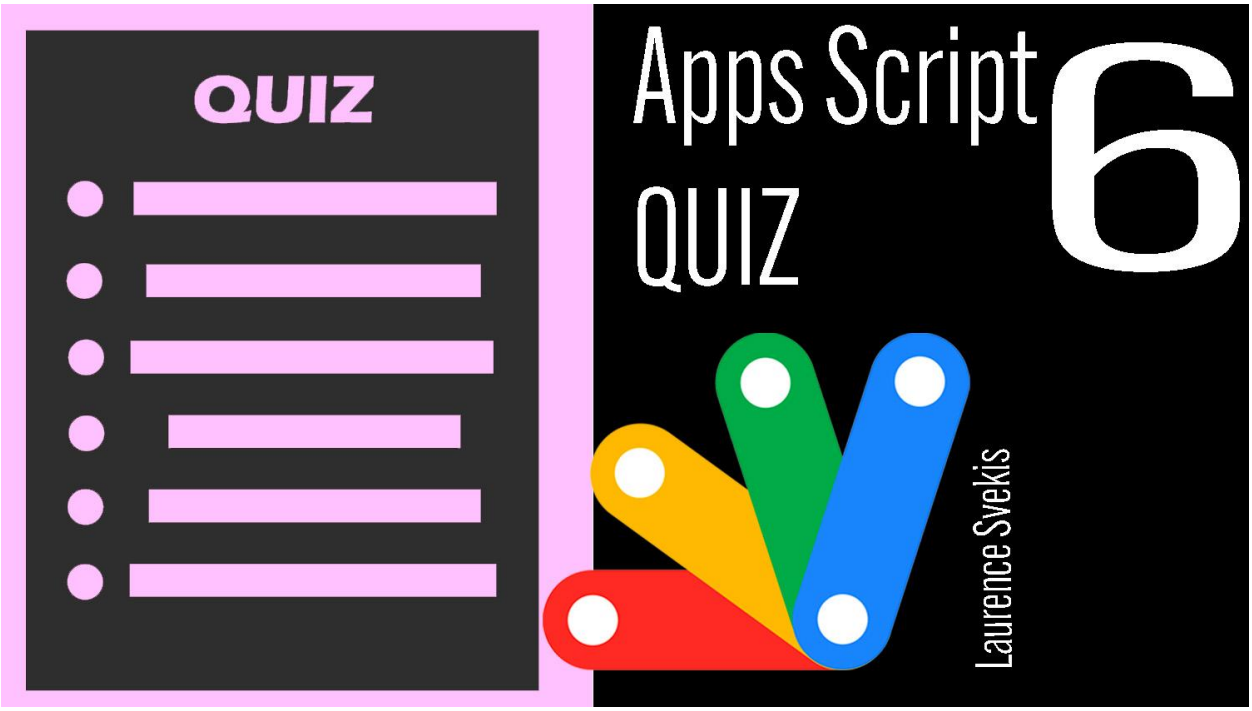
b. Gets the active sheet in the current spreadsheet.

c. Gets the sheet named "Sheet1" in the current spreadsheet.

d. None of the above.

Solution: c. Gets the sheet named "Sheet1" in the current spreadsheet.

Google Apps Script Quiz 6



[What is Apps Script?]

[Which of the following is not a trigger type in Apps Script?]

[Which method can be used to create a new menu in a Google Sheets spreadsheet?]

[What does the following code do? var sheet = SpreadsheetApp.getActiveSheet();]

[Which method can be used to insert a row into a Google Sheets spreadsheet?]

[What function can be used to concatenate two strings in Apps Script?]

[Which method can be used to copy a range of cells in a Google Sheets spreadsheet?]

What is Apps Script?

a. A programming language used to create add-ons for Google Sheets, Docs, and Forms.
b. A mobile app development platform.
c. A web development framework.
d. A game engine for building mobile games.
Solution: a. A programming language used to create add-ons for Google Sheets, Docs, and Forms.

Explanation: Apps Script is a scripting language developed by Google that allows developers to create add-ons and automate tasks in Google Sheets, Docs, and Forms.

Which of the following is not a trigger type in Apps Script?

a. Time-driven
b. Form-submit
c. Spreadsheet-edit
d. Text-message
Solution: d. Text-message. Text-message is not a trigger type in Apps Script.

Explanation: Apps Script supports several trigger types, including time-driven triggers, form-submit triggers, spreadsheet-edit triggers, and many others. However, text-message triggers are not supported.

Which method can be used to create a new menu in a Google Sheets spreadsheet?

a. createMenu()
b. addMenu()
c. insertMenu()
d. all of the above
Solution: a. createMenu(). The createMenu() method can be used to create a new menu in a Google Sheets spreadsheet.

Explanation: The createMenu() method is used to create a new custom menu in a Google Sheets spreadsheet. The addMenu() and insertMenu() methods are not valid methods in Apps Script.

What does the following code do? var sheet = SpreadsheetApp.getActiveSheet();

a. Creates a new spreadsheet with one sheet named "Sheet1".
b. Gets the active sheet in the current spreadsheet.
c. Gets the sheet named "Sheet1" in the current spreadsheet.
d. None of the above.

Solution: b. Gets the active sheet in the current spreadsheet.

Explanation: This code uses the getActiveSheet() method to get a reference to the active sheet in the current spreadsheet.

Which method can be used to insert a row into a Google Sheets spreadsheet?

a. insertRow()
b. addRow()
c. appendRow()
d. pushRow()
Solution: a. insertRow(). The insertRow() method can be used to insert a row into a Google Sheets spreadsheet.

Explanation: The insertRow() method is used to insert a new row at a specified index in a Google Sheets spreadsheet. The addRow() and appendRow() methods add a new row at the end of the sheet, and the pushRow() method is not a valid method in Apps Script.

What function can be used to concatenate two strings in Apps Script?

a. CONCATENATE()
b. CONCAT()
c. ADD()
d. JOIN()
Solution: b. CONCAT(). The CONCAT() function can be used to concatenate two strings in Apps Script.

Explanation: The CONCAT() function is used to combine two or more strings into a single string. The CONCATENATE() function is also available, but CONCAT() is recommended as it is simpler to use.

Which method can be used to copy a range of cells in a Google Sheets spreadsheet?

a. copy()
b. duplicate()
c. copyTo()
d. all of the above
Solution: c. copyTo(). The copyTo() method can be used to copy a range of cells in a Google Sheets spreadsheet.

Explanation: The copy() method creates a new copy of the range, but does not place it anywhere. The duplicate() method creates a new sheet with a duplicate of the range, but does not copy the range to an existing sheet. The copyTo() method copies the range to a new location in the same or another sheet.

Google Apps Script Coding Examples

Send Email using Gmail API

```
function sendEmail() {
  var recipient = "john.doe@example.com";
  var subject = "Test email";
  var body = "This is a test email sent from Google Apps Script";

  GmailApp.sendEmail(recipient, subject, body);
}
```

This script uses the GmailApp class in Google Apps Script to send an email to a recipient. The recipient's email address is specified in the recipient variable, the subject of the email is specified in the subject variable, and the body of the email is specified in the body variable. The sendEmail method of the GmailApp class is then called with these parameters to send the email.

Get Data from Google Sheets

```
function getData() {
  var ss = SpreadsheetApp.getActiveSpreadsheet();
  var sheet = ss.getSheetByName("Sheet1");
  var data = sheet.getDataRange().getValues();

  for (var i = 0; i < data.length; i++) {
    Logger.log(data[i][0] + " " + data[i][1]);
  }
}
```

This script uses the SpreadsheetApp class in Google Apps Script to get data from a Google Sheets spreadsheet. The active spreadsheet is retrieved with the getActiveSpreadsheet method, the sheet with the name "Sheet1" is retrieved with the getSheetByName method, and the data in the sheet is retrieved

with the getDataRange and getValues methods. The data is then logged to the console using a for loop.

Data From Sheets as Email Table

```
function tableMaker(){
  const id = '1P9R_brKeReNmrY';
  const sheet =
SpreadsheetApp.openById(id).getSheetByName('users');
  const data = sheet.getDataRange().getValues();
  const header = data[0];
  const rows = data.slice(1);
  let html = '<table style="border:1px solid black">';
  data.forEach(row=>{
    html += '<tr>';
    row.forEach(cell=>{
      html += '<td style="border:1px solid
#ddd">'+cell+'</td>';
    })
    html += '</tr>';
  })
  html += '</table>';
  const email = Session.getActiveUser().getEmail();
```

```
  const subject = 'My Table';
  MailApp.sendEmail({
    to:email,
    subject:subject,
    htmlBody:html
  });
}
```

1. Sets the id variable to a specific Google Sheets ID.
2. Opens the Google Sheets file with the given id, selects the sheet named "users", and retrieves all the data from the sheet.
3. Extracts the header row and the data rows from the retrieved data.
4. Initializes an HTML string with an opening <table> tag and a border style.
5. Loops through each row of data (including the header row), and for each row:
    a. a. Adds an opening <tr> tag to the HTML string.
    b. b. Loops through each cell in the row and adds a <td> tag with the cell value and a border style to the HTML string.
    c. c. Adds a closing </tr> tag to the HTML string.
6. Adds a closing </table> tag to the HTML string.
7. Gets the email address of the active user.

8. Sets the subject of the email to "My Table".
9. Sends an email to the active user's email address with the table HTML as the email's HTML body.

In summary, this function retrieves data from a specific Google Sheets file, creates an HTML table from the data, and sends an email with the table as the email's HTML body to the active user's email address.

Add a Custom Menu to Google Sheets

```
function onOpen() {
  var ui = SpreadsheetApp.getUi();
  ui.createMenu("Custom Menu")
      .addItem("Option 1", "option1")
      .addItem("Option 2", "option2")
      .addToUi();
}

function option1() {
  Browser.msgBox("Option 1 selected");
}

function option2() {
  Browser.msgBox("Option 2 selected");
```

```
}
```

This script adds a custom menu to a Google Sheets spreadsheet using the SpreadsheetApp and Ui classes in Google Apps Script. The onOpen function is called when the spreadsheet is opened, and it creates a new menu called "Custom Menu" with two options: "Option 1" and "Option 2". When each option is selected, a message box is displayed with the selected option.

Create a Google Calendar Event

```
function createEvent() {
  var calendar = CalendarApp.getDefaultCalendar();
  var title = "Test event";
  var startTime = new Date("March 15, 2023 08:00:00");
  var endTime = new Date("March 15, 2023 09:00:00");
  var description = "This is a test event created with
Google Apps Script";

  var event = calendar.createEvent(title, startTime,
endTime, {
    description: description
  });

  Logger.log("Event created: " + event.getTitle());
```

```
}
```

This script uses the CalendarApp class in Google Apps Script to create a new event in the user's default Google Calendar. The title of the event is specified in the title variable, the start time is specified in the startTime variable, the end time is specified in the endTime variable, and the description is specified in the description variable. The createEvent method of the CalendarApp class is then called with these parameters and an object containing the event description to create the event. Finally, the title of the created event is logged to the console using the Logger.log method.

Access the Google Drive API

```
function listFiles() {
  var folderId = "XXXXXXXXXXXXXXXXXXXXXXX"; // Replace
with your folder ID
  var folder = DriveApp.getFolderById(folderId);
  var files = folder.getFiles();

  while (files.hasNext()) {
    var file = files.next();
    Logger.log(file.getName());
  }
```

```
}
```

This script uses the DriveApp class in Google Apps Script to access the Google Drive API and retrieve a list of files in a specified folder. The ID of the folder is specified in the folderId variable, and the folder is retrieved with the getFolderById method of the DriveApp class. The files in the folder are then retrieved with the getFiles method, and a while loop is used to iterate through the files and log their names to the console using the Logger.log method.

Folder files listed to Sheet and creating new files into a folder

```
function dataFolder(){
  const sid =
'1P9R_b-dTdoBAAWAGF6kPZyKSXJ6r2P_LFgrKeReNmrY';
  const id = '1a591VSrO-dj11PJk04cm6PtpiRVepssw';
  const folder = DriveApp.getFolderById(id);
  const sheet =
SpreadsheetApp.openById(sid).getSheetByName('files');
  const files = folder.getFiles();
  Logger.log(files);
  while (files.hasNext()){
```

```
    const file = files.next();

sheet.appendRow([file.getName(),file.getId(),file.getSi
ze()]);
    Logger.log(file.getName());
  }
}

function newFiles(){
  const id = '1a591VSrO-dj11PJk04cm6PtpiRVepssw';
  const folder = DriveApp.getFolderById(id);
  const doc = DocumentApp.create('new Doc');
  const docID = doc.getId();
  const file = DriveApp.getFileById(docID);
  file.moveTo(folder);
}

function newFiles2(){
  const folder =
DriveApp.getFolderById('1a591VSrO-dj11PJk04cm6PtpiRVeps
sw');
  const doc = DocumentApp.create('new Doc 2');
  DriveApp.getFileById(doc.getId()).moveTo(folder);
}
```

dataFolder() function

The dataFolder() function does the following:

- Sets the sid variable to a specific Google Sheets ID.
- Sets the id variable to a specific Google Drive folder ID.
- Gets the folder with the given id.
- Opens the Google Sheets file with the given sid, selects the sheet named "files".
- Gets a list of files in the folder.
- Loops through each file in the folder, and for each file:
    - Appends a new row to the "files" sheet in the Google Sheets file with the file name, ID, and size.
    - Logs the file name to the console.

newFiles() function

The newFiles() function does the following:

- Sets the id variable to a specific Google Drive folder ID.
- Gets the folder with the given id.
- Creates a new Google Docs document with the name "new Doc".
- Gets the ID of the newly created document.
- Gets the file with the ID of the newly created document.
- Moves the file to the folder with the given id.

newFiles2() function

The newFiles2() function does the following:

- Gets the folder with a specific Google Drive folder ID.
- Creates a new Google Docs document with the name "new Doc 2".
- Gets the ID of the newly created document.
- Gets the file with the ID of the newly created document.
- Moves the file to the folder with the specific Google Drive folder ID.

In summary, the dataFolder() function logs the names of files in a Google Drive folder and adds the file name, ID, and size to a specific Google Sheets file. The newFiles() and newFiles2() functions create a new Google Docs document with a specific name and move it to a specific Google Drive folder.

Sending Emails From Google Sheets:

One useful application of Google Apps Script is automating email sending. You can use Google Sheets as a data source and send personalized emails to recipients based on the information in the sheet. Here is an example code:

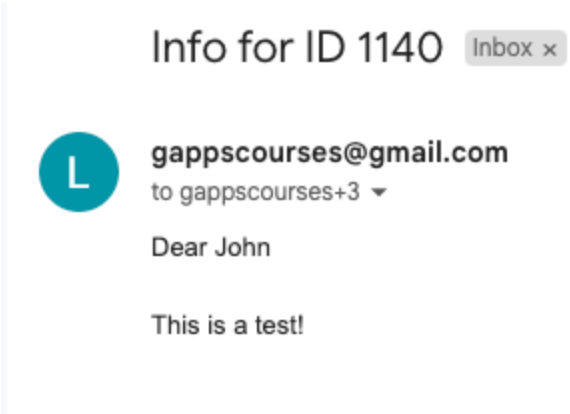
```
function sendEmails() {
  var sheet = SpreadsheetApp.getActiveSheet();
  var dataRange = sheet.getDataRange();
  var data = dataRange.getValues();
```

```
  for (var i = 1; i < data.length; i++) {
    var row = data[i];
    var emailAddress = row[0];
    var message = 'Dear ' + row[1] + ',\n\n' + 'This is a test
email.';
    var subject = 'Test Email';
    MailApp.sendEmail(emailAddress, subject, message);
  }
}
```

This code loops through the rows of a sheet, retrieves the email address and name of each recipient, and sends a personalized email using the MailApp.sendEmail() function.

Sheet Data Custom Emails

	A	B	C
1	**Name**	**Email**	**ID**
2	Laurence	gappscourses+1@gmail.(	100
3	Laurence	gappscourses+4@gmail.(	50
4	Laura	gappscourses+2@gmail.(	50
5	Laura	gappscourses+5@gmail.(	350
6	John	gappscourses+3@gmail.(	1140




```
function sender(){
  const id = '1P9R_FgrKeReNmrY';
  const sheet =
SpreadsheetApp.openById(id).getSheetByName('users');
  const data = sheet.getDataRange().getValues();
  const users = data.slice(1);
  users.forEach(user =>{
    const email = user[1];
    const message = `Dear ${user[0]} \n\nThis is a
test!`;
    const subject = `Info for ID ${user[2]}`;
    MailApp.sendEmail(email,subject,message);
  })
}
```

- Sets the id variable to a specific Google Sheets ID.
- Opens the Google Sheets file with the given id and selects the sheet named "users".

- Gets the data range from the "users" sheet and gets all values from it.
- Slices the data array to remove the first row (header row) and create a new array named users with only the user information.
- Loops through each user in the users array using the forEach() method.
- For each user:
    - Sets the email variable to the email address of the user, which is located at index 1 in the user array.
    - Constructs a message string that includes the user's name and a test message.
    - Sets the subject variable to a string that includes the user's ID, which is located at index 2 in the user array.
    - Sends an email to the user using the MailApp.sendEmail() method, passing the email, subject, and message variables as arguments.

In summary, the sender() function retrieves user data from a Google Sheets file, constructs personalized messages for each user, and sends an email to each user with their name, ID, and a test message.

Creating Google Calendar Events:

Another useful application of Google Apps Script is creating Google Calendar events programmatically. Here is an example code:

```
function createEvent() {
  var calendar = CalendarApp.getDefaultCalendar();
  var event = calendar.createEvent('Test Event', new Date('March 2, 2023 09:00:00 EST'), new Date('March 2, 2023 10:00:00 EST'), {
    description: 'This is a test event'
  });
  Logger.log('Event ID: ' + event.getId());
}
```

This code creates a new Google Calendar event with a title, start and end date/time, and a description. It also logs the ID of the newly created event for reference.

Accessing and Modifying Google Sheets:

Google Sheets is a popular application for managing and analyzing data. Google Apps Script can be used to access and modify data in Sheets programmatically. Here is an example code:

```
function modifySheet() {
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  var data = sheet.getDataRange().getValues();
  for (var i = 0; i < data.length; i++) {
    var row = data[i];
    if (row[0] == 'John') {
      sheet.getRange(i + 1, 2).setValue('Active');
    }
  }
}
```

This code retrieves data from the active sheet, loops through each row, and checks if the name in the first column is 'John'. If it is, the script sets the value of the second column to 'Active' using the sheet.getRange() and .setValue() functions.

Update Sheet Access Sheet Data with Apps Script

```
function modData(){
  const id = '1P9RFgrKeReNmrY';
  const sheet =
SpreadsheetApp.openById(id).getSheetByName('users');
  const data = sheet.getDataRange().getValues();
```

```
  data.forEach((row,index)=>{
    if(row[0] == 'Laurence'){
      const range = sheet.getRange(index+1,1);
      range.setBackground('blue');
      //range.setValue('Active');
    }
  })
}
```

The modData() function does the following:

- Sets the id variable to a specific Google Sheets ID.
- Opens the Google Sheets file with the given id and selects the sheet named "users".
- Gets the data range from the "users" sheet and gets all values from it.
- Loops through each row of data using the forEach() method, passing the current row and its index as arguments.
- For each row of data:
    - Checks if the value in the first column (column A) is equal to "Laurence".
    - If the value is "Laurence":
        - Gets the range of the cell in the first column and the same row as the current row using the getRange() method, passing the row and column as arguments.

- Sets the background color of the cell using the setBackground() method, passing the color as an argument. In this case, the cell's background color is set to blue.
- Commented out: Sets the value of the cell to "Active" using the setValue() method.

In summary, the modData() function modifies the data in a Google Sheets file by changing the background color of the cell in the first column and the same row as the row with the name "Laurence" to blue. It also includes commented-out code that sets the value of the same cell to "Active".

Accessing and Modifying Google Docs:

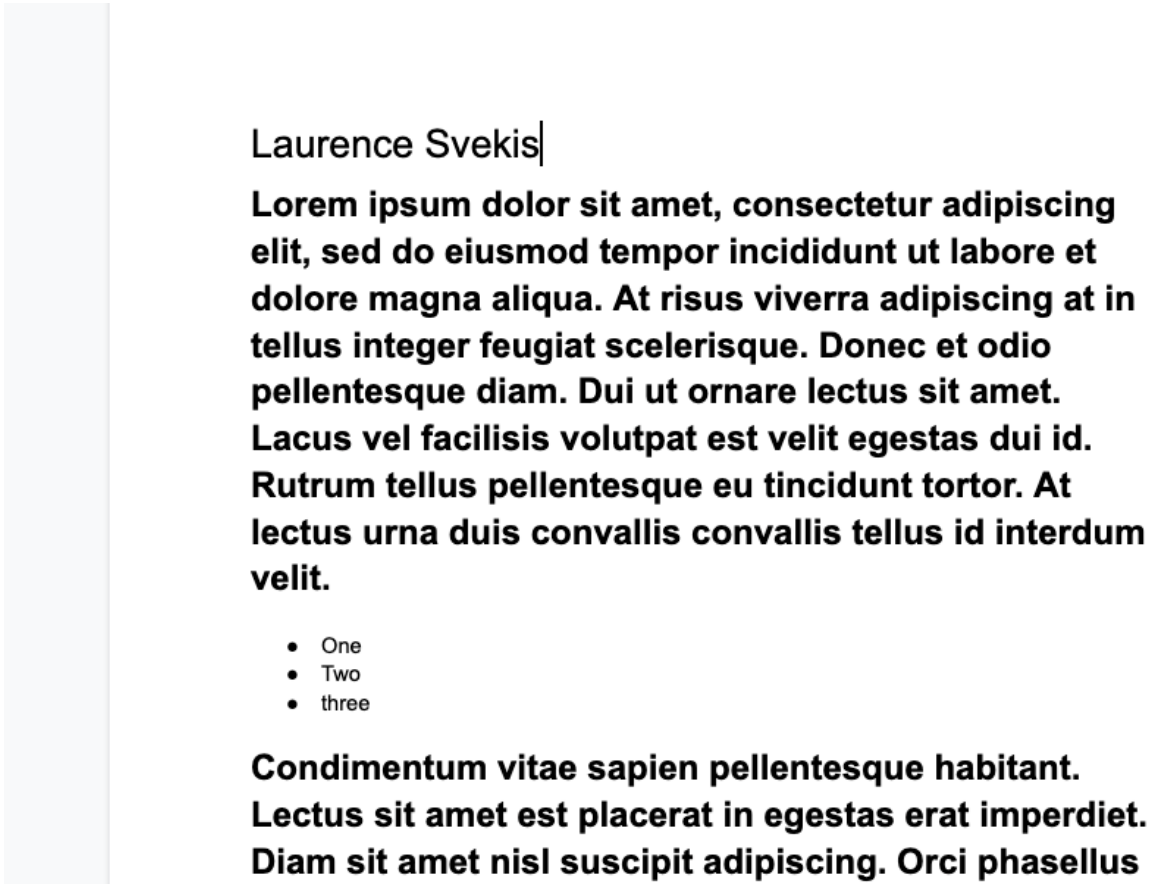
Google Docs is a popular application for creating and sharing documents. Google Apps Script can be used to access and modify documents programmatically. Here is an example code:

```
function modifyDoc() {
  var doc = DocumentApp.getActiveDocument();
  var body = doc.getBody();
  var paragraphs = body.getParagraphs();
  for (var i = 0; i < paragraphs.length; i++) {
    var paragraph = paragraphs[i];
```

```
    if (paragraph.getText().indexOf('Lorem ipsum') !==
-1) {
      paragraph.setAttributes({
        bold: true
      });
    }
  }
}
```

This code retrieves the active Google Doc, loops through each paragraph, and checks if the text contains the phrase 'Lorem ipsum'. If it does, the script sets the bold attribute of that paragraph to true using the paragraph.setAttributes() function.

Update Styling of element with specific text in the contents




```
function modDoc(){
  const id = '1SCE46vJwZCXXVY';
  const doc = DocumentApp.openById(id);
  const body = doc.getBody();
  const paras = body.getParagraphs();
  const style = {};
  style[DocumentApp.Attribute.BOLD] = true;
  style[DocumentApp.Attribute.FONT_SIZE] = 18;
  paras.forEach(para =>{
    if(para.getText().indexOf('Lorem ipsum') !== -1){
```

```
      para.setAttributes(style);
    }
  })
}
```

This code modifies a Google Document by applying some formatting to specific paragraphs.

1. The first line defines a variable id that stores the ID of the Google Document that will be modified.
2. The second line opens the Google Document with the given ID using the openById method of the DocumentApp class, and assigns it to the variable doc.
3. The third line gets the body of the document using the getBody method of the Document class, and assigns it to the variable body.
4. The fourth line gets all the paragraphs in the document using the getParagraphs method of the Body class, and assigns them to the variable paras.
5. The fifth line defines a JavaScript object called style that will be used to apply the formatting. The object has two properties: BOLD and FONT_SIZE, which are constants defined in the Attribute enum of the DocumentApp class. The BOLD property is set to true, and the FONT_SIZE property is set to 18.

6. The code then iterates over each paragraph in the document using a forEach loop.
7. For each paragraph, it checks if the text of the paragraph contains the string "Lorem ipsum" using the indexOf method of the String class. If the string is found (i.e., indexOf returns a value greater than or equal to 0), the code applies the formatting defined in the style object to the paragraph using the setAttributes method of the Paragraph class.
8. Once the loop has finished, the code exits and the modified document is saved automatically.

Creating Google Forms:

Google Forms is a powerful tool for creating surveys, quizzes, and other types of forms. Google Apps Script can be used to create and customize Google Forms programmatically. Here is an example code:

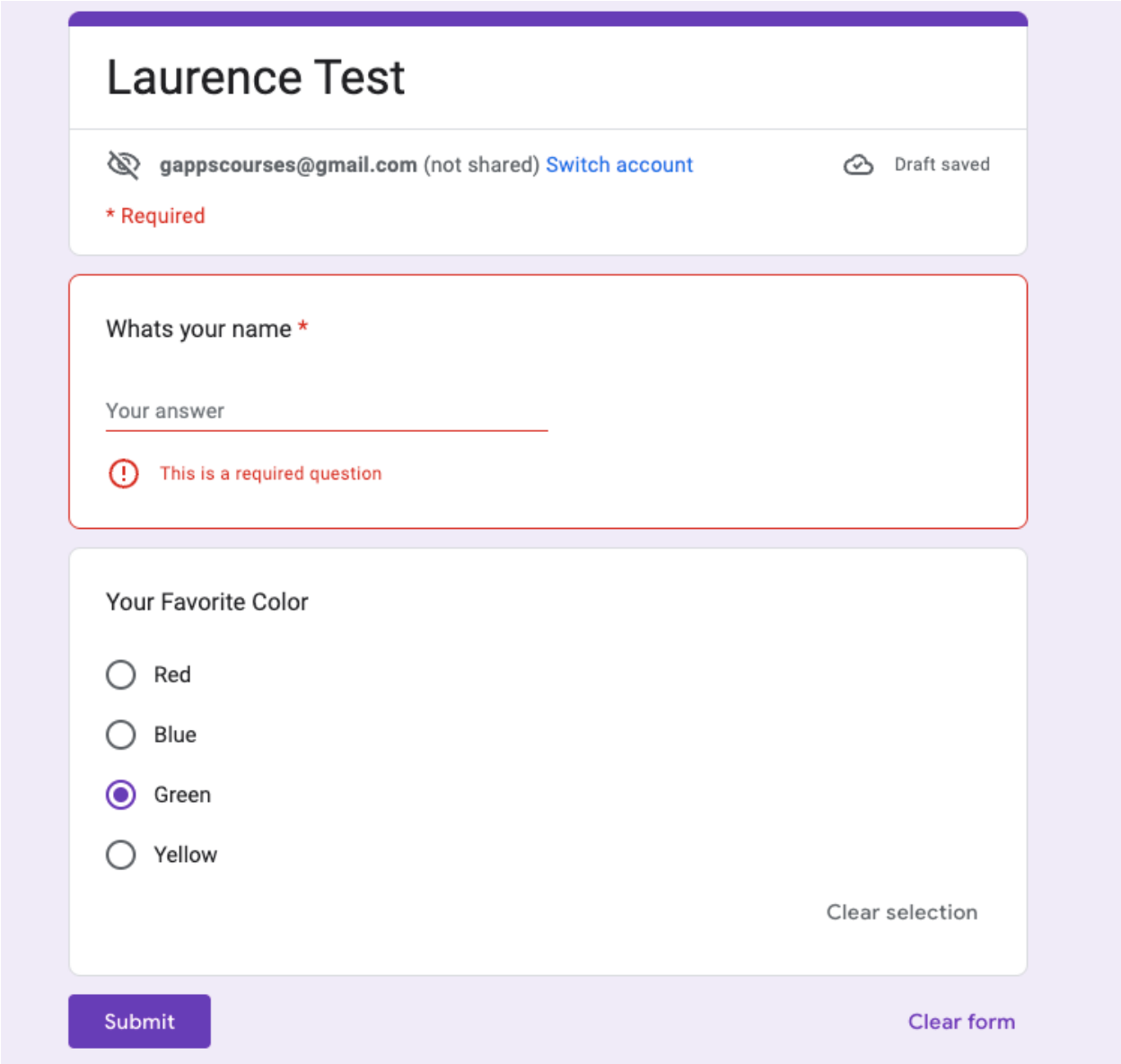
```
function createForm() {
  var form = FormApp.create('Test Form');
  form.addTextItem()
    .setTitle('What is your name?')
    .setRequired(true);
  form.addMultipleChoiceItem()
```

```
    .setTitle('What is your favorite color?')
    .setChoices([
      form.createChoice('Red'),
      form.createChoice('Blue'),
      form.createChoice('Green')
    ])
    .setRequired(true);
  Logger.log('Form URL: ' + form.getPublishedUrl());
}
```

This code creates a new Google Form with a title, a text question, and a multiple choice question with three choices. It also logs the URL of the published form for reference. The form can be further customized using other methods available in the Form service of Google Apps Script.

These are just a few examples of what you can do with Google Apps Script. With this powerful scripting language, you can automate various tasks and extend the functionality of Google Workspace applications according to your needs.

Create and view form with code




```
function makeForm(){
  const form = FormApp.create('Laurence Test');
  form.addTextItem()
  .setTitle('Whats your name')
  .setRequired(true);
  const item1 = form.addMultipleChoiceItem();
```

```
  item1.setTitle('Your Favorite Color')
  .setChoices([
    item1.createChoice('Red'),
    item1.createChoice('Blue'),
    item1.createChoice('Green'),
    item1.createChoice('Yellow')
  ])
  Logger.log(form.getPublishedUrl());
}
```

The makeForm() function creates a new Google Form called "Laurence Test" and adds two types of form items to it: a text item and a multiple choice item.

The first item added is a text item which asks the user to enter their name. The setTitle() method sets the title of the item to "What's your name?" and the setRequired() method makes answering this question mandatory.

The second item added is a multiple choice item which asks the user to select their favorite color. The setTitle() method sets the title of the item to "Your Favorite Color" and the setChoices() method sets the possible answer options to "Red", "Blue", "Green", and "Yellow".

Finally, the Logger.log() method is used to log the URL of the published form to the console, which can be used to share the form with others.

Creating and Updating Google Slides:

```
function makeSlides(){
  const pre = SlidesApp.create('Laurence Test2');
  const slide = pre.getSlides()[0];
  slide.insertTextBox('Laurence Svekis',20,20,250,250);
}

function upSlides(){
  const id = '1qTkK20Av6Rm5G5i3NhP276fO3ey48';
  const pre = SlidesApp.openById(id);
  const slide = pre.getSlides()[0];
  const eles = slide.getShapes();
  eles.forEach(ele =>{
    Logger.log(ele.getText().asString());
    ele.getText().replaceAllText('Laurence','Mr');
  })
}
```

The first function makeSlides() creates a new Google Slides presentation titled "Laurence Test2". The function then retrieves the first slide in the presentation and inserts a new text box on it using the insertTextBox() method. The text box is positioned at coordinates (20, 20) with a width and height of 250.

The second function upSlides() opens an existing Google Slides presentation with the ID 1q276fO3ey48. It retrieves the first slide in the presentation and gets all the shapes on it using the getShapes() method. The function then loops through all the shapes and logs the text content of each shape to the console using the getText() method. Finally, it replaces all occurrences of the string "Laurence" in the text of each shape with "Mr" using the replaceAllText() method.

Accessing and Modifying Google Forms:

Google Forms is a powerful tool for creating surveys, quizzes, and other types of forms. Google Apps Script can be used to access and modify existing Google Forms programmatically. Here is an example code:

```
function modifyForm() {
  var form =
FormApp.openByUrl('https://forms.google.com/...');
  var items = form.getItems();
```

```
  for (var i = 0; i < items.length; i++) {
    var item = items[i];
    if (item.getTitle() == 'Favorite color') {

item.asMultipleChoiceItem().setChoiceValues(['Red',
'Blue', 'Green']);
    }
  }
  Logger.log('Form URL: ' + form.getPublishedUrl());
}
```

This code opens an existing Google Form by URL, retrieves all the form items, and modifies the choices of the multiple choice item with the title 'Favorite color'. It also logs the URL of the published form for reference.

Select and update the form choices

```
function modForm(){
  const id = '1Q1ZUposHg';
  const form = FormApp.openById(id);
  const items = form.getItems();
  items.forEach(item =>{
    Logger.log(item.getTitle());
```

```
  })
  const mul = items[1].asMultipleChoiceItem();
  mul.setChoiceValues(['Purple','Pink','Teal']);
}
```

The code modForm() does the following:

1. The variable id is assigned the ID of an existing Google Form.
2. The openById() method of FormApp is used to open the Google Form with the given ID, and its reference is stored in the variable form.
3. The getItems() method of form is used to get all the items in the form, and they are stored in the items array.
4. A forEach() loop is used to iterate over the items in the items array. For each item, its title is logged to the console using the getTitle() method of the item.
5. The second item in the form (at index 1 in the items array) is assumed to be a multiple choice question and is assigned to the variable mul using the asMultipleChoiceItem() method.
6. The setChoiceValues() method of mul is used to set the available choices for the multiple choice question to ['Purple','Pink','Teal'].

Overall, the code opens a Google Form, gets all the items in the form, logs their titles, and then modifies the second item in the form to change its available choices for answers.

Using Google Sheets as a Database:

Google Sheets can be used as a database to store and retrieve data. Google Apps Script can be used to interact with the Sheets database programmatically. Here is an example code:

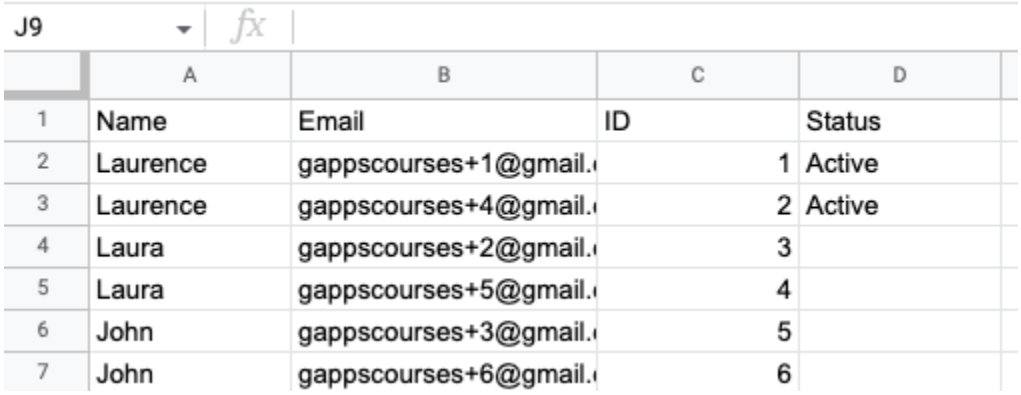
```
function getCustomerData(customerId) {
  var sheet =
SpreadsheetApp.openById('...').getSheetByName('Customer
s');
  var data = sheet.getDataRange().getValues();
  for (var i = 1; i < data.length; i++) {
    var row = data[i];
    if (row[0] == customerId) {
      return {
        name: row[1],
        email: row[2],
        phone: row[3]
      };
    }
  }
```

```
  return null;
}
```

This code retrieves customer data from a Google Sheets database by ID. It loops through all the rows of the 'Customers' sheet, matches the customer ID with the first column of each row, and returns an object with the name, email, and phone number of the customer if a match is found. If no match is found, it returns null.

Sheet Data as JSON output in WebApp

{"name":"Laurence","email":"gappscourses+1@gmail.com","id":1,"status":"Active"}



J9

	A	B	C	D
1	Name	Email	ID	Status
2	Laurence	gappscourses+1@gmail.(	1	Active
3	Laurence	gappscourses+4@gmail.(	2	Active
4	Laura	gappscourses+2@gmail.(	3	
5	Laura	gappscourses+5@gmail.(	4	
6	John	gappscourses+3@gmail.(	5	
7	John	gappscourses+6@gmail.(	6	

```
function getUserInfo(id){
  const sid = '1P9R_brKeReNmrY';
  const sheet =
SpreadsheetApp.openById(sid).getSheetByName('users');
```

```
  const data =
sheet.getDataRange().getValues().slice(1);
  let rep = null;
  data.forEach(ele =>{
    if(ele[2] == id){
      rep =
{name:ele[0],email:ele[1],id:ele[2],status:ele[3]}
    };
  })
  return rep;
}

function tester(){
  Logger.log(getUserInfo(3));
}

function doGet(e){
  let data = null;
  if('id' in e.parameters){
    let val = e.parameters['id'][0];
    data = getUserInfo(val);
  }
  const output = JSON.stringify(data);
```

```
  return
ContentService.createTextOutput(output).setMimeType(Con
tentService.MimeType.JSON);
}
```

The getUserInfo() function takes an id parameter and returns an object containing information about a user. It first sets a variable sid to a string value, which is a Google Sheets ID. The function then uses SpreadsheetApp to open the specified sheet and get the sheet named 'users'. The getDataRange() method is used to get all the data in the sheet, and then the slice(1) method is called to exclude the header row. The forEach() method is then used to iterate over each row of data, and if the third column of the row matches the id parameter, an object is created with the user's name, email, id, and status, which is assigned to the rep variable. Finally, the rep variable is returned.

The tester() function calls the getUserInfo() function with a parameter value of 3 and logs the result using the Logger.log() method.

The doGet() function is an HTTP request handler function that takes a single e (event) parameter. It first sets a data variable to null. If the id parameter is present in the e.parameters object, then it sets the val variable to the first value of the id parameter

array, and calls the getUserInfo() function with the val parameter, and sets the returned value to the data variable. The function then uses JSON.stringify() to convert the data variable to a JSON string, and returns a ContentService object with the JSON string as the response body and the MIME type set to ContentService.MimeType.JSON.

In summary, the getUserInfo() function retrieves user information from a Google Sheet based on the id parameter value, the tester() function tests the getUserInfo() function by logging the result, and the doGet() function handles HTTP requests by calling the getUserInfo() function and returning the result as a JSON string.

Sending Emails With Attachments:

Google Apps Script can be used to send emails with attachments using the Gmail service. Here is an example code:

```
function sendEmailWithAttachment() {
  var recipient = 'example@example.com';
  var subject = 'Test Email with Attachment';
  var body = 'This is a test email with an
attachment.';
  var attachment = DriveApp.getFileById('...');
```

```
  GmailApp.sendEmail(recipient, subject, body, {
    attachments: [attachment]
  });
}
```

This code sends an email with an attachment to a recipient using the GmailApp.sendEmail() function. The attachment is retrieved by ID using the DriveApp.getFileById() function.

File as an attachment in email

```
function sendAtt(){
  const id = '1SCECXXVY';
  const rep = Session.getActiveUser().getEmail();
  const sub = 'Test Email';
  const body = 'Hello World Testing....';
  const att = DriveApp.getFileById(id);
  MailApp.sendEmail(rep,sub,body,{
    attachments:[att]
  });
}
```

This code defines a function sendAtt() that sends an email with an attachment to the active user's email address. The function

starts by setting a constant id to a string value representing the ID of the file to be attached. The Session.getActiveUser().getEmail() method is used to get the email address of the active user, which is assigned to the constant rep. The constants sub and body are then set to strings representing the email subject and body, respectively. The DriveApp.getFileById() method is called to get a file with the specified ID, and the resulting File object is assigned to the constant att. Finally, the MailApp.sendEmail() method is called with the rep parameter as the recipient email address, sub as the email subject, body as the email body, and an object with an attachments property that contains an array with the att file object. This sends an email to the active user with the specified subject, body, and attachment.

Creating and Modifying Google Calendar Events:

Google Calendar is a powerful tool for managing events and schedules. Google Apps Script can be used to create and modify Google Calendar events programmatically. Here is an example code:

```
function createCalendarEvent() {
  var calendar = CalendarApp.getCalendarById('...');
```

```
  var event = calendar.createEvent('Test Event',
    new Date('2023-03-03T10:00:00.000Z'),
    new Date('2023-03-03T11:00:00.000Z'),
    {description: 'This is a test event.'});
  Logger.log('Event ID: ' + event.getId());
}

function modifyCalendarEvent() {
  var calendar = CalendarApp.getCalendarById('...');
  var eventId = '...';
  var event = calendar.getEventById(eventId);
  event.setTitle('New Title');
  event.setDescription('New Description');
  event.setTime(new Date('2023-03-03T12:00:00.000Z'),
    new Date('2023-03-03T13:00:00.000Z'));
  Logger.log('Event ID: ' + event.getId());
}
```

The first function creates a new Google Calendar event with a title, start and end times, and a description. It logs the ID of the newly created event for reference.

The second function modifies an existing Google Calendar event by ID. It changes the title, description, and start and end times of the event. It also logs the ID of the modified event for reference.

These are just a few examples of what you can do with Google Apps Script. With this powerful scripting language, you can automate various tasks and extend the functionality of Google Workspace applications according to your needs.