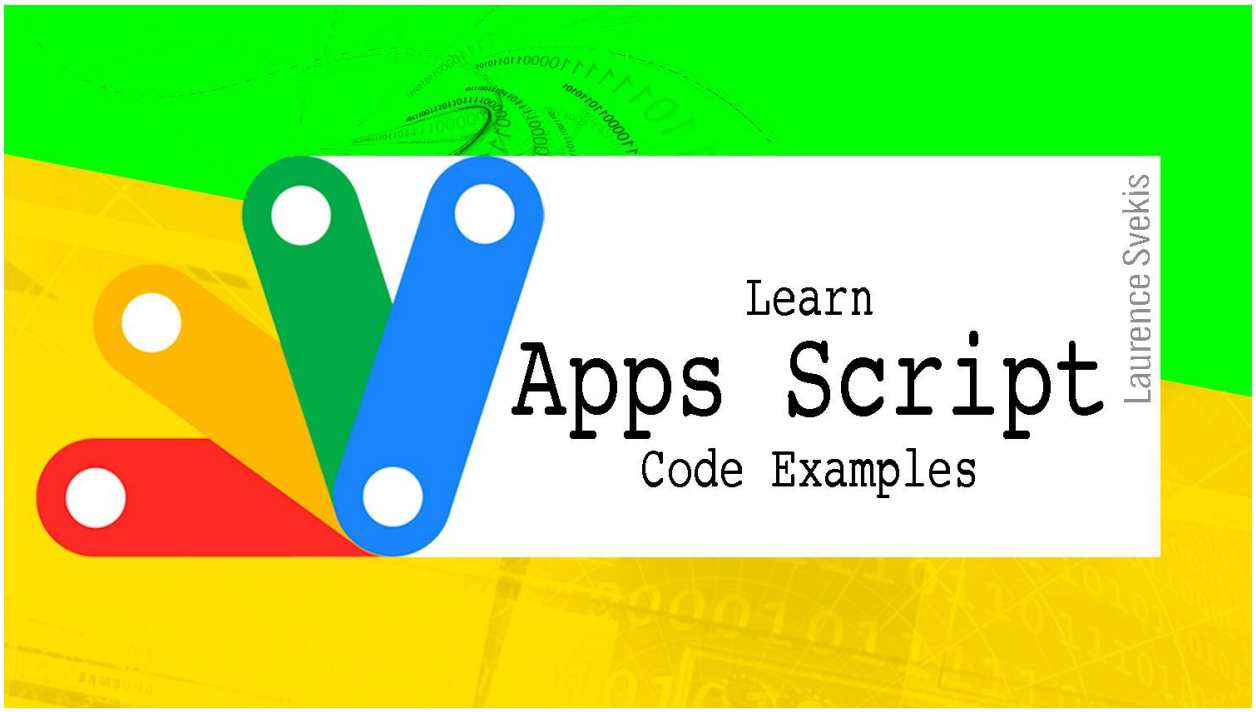
Apps Script Fun Coding Exercises 5



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);
}
```

Send a Slack Message

Create a script that automatically sends a Slack message when a new row is added to a Google Sheet.
https://api.slack.com/messaging/webhooks

```
function sendSlackMessage() {
  var webhookUrl =
"https://hooks.slack.com/services/T00000000/B00000000/X
XXXXXXXXXXXXXXXXXXXXX";
  var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("S
heet1");
  var data = sheet.getRange(sheet.getLastRow(), 1, 1,
sheet.getLastColumn()).getValues()[0];
  var message = "New row added to Google Sheet:\n" +
data.join("\n");
  var payload = {text: message};
  var options = {method: "POST", contentType:
"application/json", payload: JSON.stringify(payload)};
  UrlFetchApp.fetch(webhookUrl, options);
}
```

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.