Apps Script Workspace Code

Google Workspace Services Docs Calendar Drive Forms Gmail Sheets Slides

How to interact with Google Workspace services examples of using Classes in workspace services and how to access class methods.

DocumentApp Class Methods and Properties

The Document Service allows scripts to create, access, and modify Google Docs files.
https://developers.Google.com/apps-script/reference/document
DocumentApp is the parent class for the Documents, which contains properties that can be used to update the styling of elements, methods to create and open the documents.
Once you have created a Document you can select the Document as an object and use the Classes to interact with it.
You can create or select an existing document or with a bound script get the active document.

```
const doc = DocumentApp.create('New Doc 1');
const doc = DocumentApp.openById(id);
const doc = DocumentApp.getActiveDocument();
```

The body is an element representing a document body. First get the body so you can use the contents of the document and interact with it.

```
const body = doc.getBody();
```

Create a Document that you can use for the exercise.

```
function makeDoc() {
 const doc = DocumentApp.create('New Doc 1');
 Logger.log(doc.getUrl());
 Logger.log(doc.getId());
}
```

Create and update the document contents within the body object.

1. Select the document you want to use, get the body object
2. Add a paragraph of text to the page appendParagraph()
3. Set the paragraph as a heading. setHeading(DocumentApp.ParagraphHeading.HEADING1)
4. Create a style to apply to an element. If you add the style to the following elements will also inherit the style. Use an object to contain the style properties for it. Select an element and apply the style to it. setAttributes(style1)
5. Create an array with nested arrays for rows of data. This can be used to create a table element. Append the new table to the page using the data from the array. appendTable(myTable)
6. Add a new row of data to the existing table, append the table with a row that can be used to insert cells. appendTableRow()
7. Create an array of data you want to add, loop through the data and app the cells to the row you created. appendTableCell(ele)
8. Add a new horizontal rule to the body. appendHorizontalRule()
9. Add an image, first select the image url. Using fetch select the image data into a variable. UrlFetchApp.fetch(url); Next append the image to the body of the doc. body.appendImage(img);
10. Create a new list and append it to the body of the document. body.appendListItem('New item 1')
11. Add another page element like a horizontal rule.
12. Create a loop to add 10 more list items, continue the list count from the previous list by setting the list id to match the original list. setListId(list1)

13. To append to an existing list in the same element as the list, you need to select the child index value from the element to insert it into the body. Add a new item inserting it after the existing element which will automatically join the new elements into the existing list.
14. Create a loop to add 5 more list items into the list using the body.insertListItem()
15. Add a new paragraph and then append some text to that paragraph element. body.appendParagraph() and appendText()

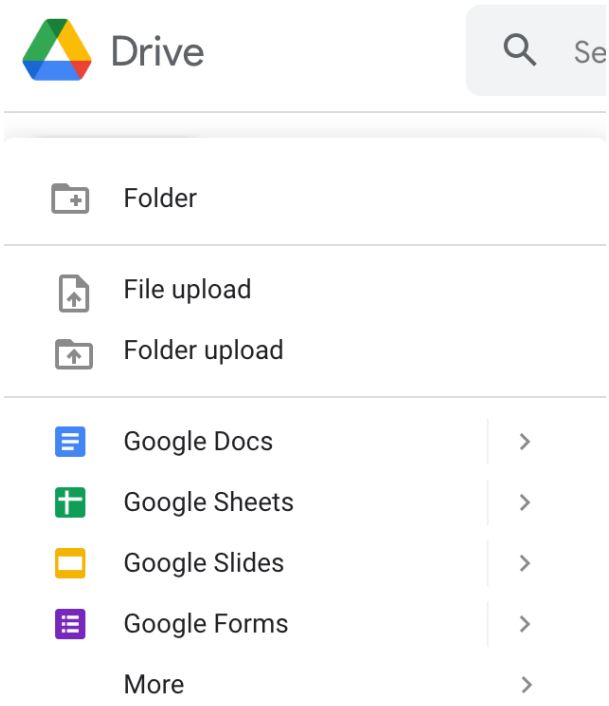
```
function updateDoc(){
 const id = '1i6Ybb7rvAtYrUAYgi0rCNKnw_qJmdlHv_NteVkQ_zLw';
 const doc = DocumentApp.openById(id);
 Logger.log(doc.getName());
 const body = doc.getBody();
 body.clear();
 const heading = body.appendParagraph('Laurence Svekis');
 Logger.log(heading);
 heading.setHeading(DocumentApp.ParagraphHeading.HEADING1);
 const style = {};
 style[DocumentApp.Attribute.FOREGROUND_COLOR] = '#ff0000';
 style[DocumentApp.Attribute.FONT_FAMILY] = 'Calibri';

 const style1 = {};
 style1[DocumentApp.Attribute.FOREGROUND_COLOR] = '#000000';


 const myTable = [
   ['Col 1','Col 2','Col 3'],
   ['1','Laurence','Svekis'],
```

```
  ['2','Jane','Doe']
];
const tab1 = body.appendTable(myTable);
tab1.setAttributes(style1);
Logger.log(tab1);
//tab1.appendTableRow().appendTableCell('3');
const arr = ['3','Jack','Jones'];
const row = tab1.appendTableRow();
Logger.log(row);
arr.forEach((ele)=>{
  row.appendTableCell(ele);
})

body.appendHorizontalRule();
const url = 'http://www.discoveryvip.com/img/d.png';
const img = UrlFetchApp.fetch(url);
body.appendImage(img);


const list1 = body.appendListItem('New item 1');
Logger.log(list1.getListId());
list1.appendText(' more text');
body.appendHorizontalRule();
for(let i=0;i<10;i++){
  const li = body.appendListItem(`Item ${i}`);
  li.setListId(list1);
}
```

```
 const myLi  = body.getChildIndex(list1);
 Logger.log(myLi);
 const newLi = body.insertListItem(myLi+1,'New Item ADDED**');
 for(let i=0;i<5;i++){
   const li = body.insertListItem(myLi+2+i,`NEW!!!!! ${i}`);
 }
 const par1 = body.appendParagraph('Hello World');
 Logger.log(par1);
 par1.appendText('Add New Text');
 heading.setAttributes(style);
}
```

How to send an email auto response on form submission, setup custom logging into a sheet of data.

Go to Drive and select to create a new form.



Within the form, setup questions to collect an email, and first and last name from the user. In the setting tab of the form you can customize the setting options.
Select the response tab, and connect the form to a new spreadsheet by pressing the 3 dots icon. This will create a new spreadsheet where we will write the Apps Script code.
In the new spreadsheet select under the extensions tab to open a new Apps Script which will be a bound script to the spreadsheet container.
To setup a trigger use the ScriptApp service and add a new trigger to run a function called added on the form submitted into the sheet.

```
function makeTrigger() {
 const sheet = SpreadsheetApp.getActive();
 ScriptApp.newTrigger('adder')
   .forSpreadsheet(sheet)
   .onFormSubmit()
   .create();
}
```

Form event object data - https://developers.Google.com/apps-script/guides/triggers/events#Google_sheets_events

How to create and log into a sheet form submitted data.

1. The event object from the form submission trigger will pass the object details into the function. Within the event object it captures the namedValues property which will contain the data of the submitted form.
2. Open the sheet you want to use to log info into. Select a sheet to use. Create an object of the data that you want to log into the sheet coming from the event object data.
3. You can use the me Object prototype of values to convert the object values into an arr. const arr = Object.values(userinfo);
4. Add the JSON string of the object into the arr with the push method. arr.push(json);
5. Append the row of data to the sheet.

```
function adder(user){
 const id = '14MXkJF5AykU9ND-z4strmly5zN_ant403-6YA6br0PY';
 const ss = SpreadsheetApp.openById(id);
 const sheet = ss.getSheetByName('log');
 const userinfo = {
   first : user.namedValues["First"][0],
   email : user.namedValues["Email Address"][0],
   last : user.namedValues["Last"][0],
   row : user.range.rowStart
 }
 const json = JSON.stringify(userinfo);
```

```
 const arr = Object.values(userinfo);
 arr.push(json);
 sheet.appendRow(arr);
 sender(userinfo);
}
```

How to send sheet data to an email as an autoresponder to a form submission event.

The object data from the form submission can be captured and used to send an autoresponder email to the user or/and to an admin email.

1. Create a test function with the event object data to test with.
2. Send the preset user object data into a function called sender which can be used to make the html and send the email.
3. Create a function called sender, build the html for the email in this function. You can use the for loop to loop through the contents of the object to output into a table.
4. Optionally, if you want to send a custom PDF attachment from the form data you can construct the html for the PDF as well. Give the blob a name using the .pdf extension.
5. Send the email with the MailApp service, if you have an attachment it can be attached using the advanced options for the mailapp service.

```
function test1(){
 const user = {"authMode":"FULL","namedValues":{"Email
Address":["gap****ses@gmail.com"],"Last":["Svekis"],"Timestamp":
["1/1/2022"],"First":["Laurence"]},"range":{"columnEnd":4,"colum
```

nStart":1,"rowEnd":2,"rowStart":2},"source":{},"triggerUid":"766
7511207665245324","values":["1/1/2022","gap****ses@gmail.com","L
aurence","Svekis"]}

```
 const userinfo = {
   first : user.namedValues["First"][0],
   email : user.namedValues["Email Address"][0],
   last : user.namedValues["Last"][0],
   row : user.range.rowStart
 }
 Logger.log(userinfo);
 sender(userinfo);
}

function sender(user){
 let html = `<h1>Hi ${user.first} ${user.last}</h1>`;
 html += '<div>Thank you for submitting your form</div>'
 html += '<table><tr>';
 for(key in user){
   const val = user[key];
   html += `<td>${val}</td>`;
 }
 html += '</tr></table>';
 html += '<div>Have a nice day, Laurence</div>';
 const blob = Utilities.newBlob(html,MimeType.HTML);
 blob.setName(`PDF for ${user.first}.pdf`);
 MailApp.sendEmail({
   recipient:user.email,
```

```
    subject:'Form submitted',
    htmlBody:html,
    cc : 'gap****ses@gmail.com',
    attachments:[blob.getAs(MimeType.PDF)]
  });
}
```

How to create forms with Google Form Service

Forms service allows scripts to create, access, and modify Google Forms.
Forms can be created with code, you can set the parameters, questions and set the type of questions all within the form service using Google Apps Script.

Create a form from scratch, send the form link to an email address. Forms will automatically collect results of the responses directly within the main form file, this can be accessed from your drive once the form is created.

1. Using FormApp create a form and give it a name
2. Set a text item question, which is required, add a title to the question.
3. Get the current users email from the Session object, this will return the Apps Script email of the users account that is writing the code.
4. Generate the form URL and id to send within an email to yourself
5. Create the mailApp service to send an email with the new form details.

```
function makeaForm() {
 const nameForm = 'my Form 2';
 const form = FormApp.create(nameForm );
 const que1 = form.addTextItem();
 que1.isRequired();
```

```
 que1.setTitle('Your first Name')
 Logger.log('URL'+form.getPublishedUrl());
 Logger.log('ID '+form.getId());
 const email = Session.getActiveUser().getEmail();
 Logger.log(email);
 const url = form.getPublishedUrl();
 const id = form.getId();
 let html  = `<h1>New Form ${nameForm}</h1>`;
 html += `<div>Your form is at ${url}</div>`;
 html += `<div>Your form id is ${id}</div>`;
 MailApp.sendEmail({
   to: email,
   subject : 'New form ' + nameForm,
   htmlBody : html
 });
}
```

How to add more items to an existing form

If you have the form id you can access and update the form file using Apps Script.

1. open the form file and add a new checkbox item to the form.

```
function updaterForm(){
 const id = '1bo5jc4APUEwAs_222aswIfPDcvU7ZEgjcuhDcpqXOiU';
 const form = FormApp.openById(id);
 const que2 = form.addCheckboxItem();
 que2.setTitle('What is your favorite Google App?');
 que2.setChoices([
   que2.createChoice('Docs'),
```

```
    que2.createChoice('Sheets'),
    que2.createChoice('Slides'),
    que2.createChoice('Forms')
  ])
}
```

How to update an existing form.

You can select existing form elements and add new ones to the form.

1. Add a new multiple choice element to the form.
2. Set the choice values using an array
3. Add a new checkbox item to the form, creating the choices with the createChoice method in the form element.
4. Separate the form questions, adding a new page with addPageBreakItem(), and also set a title for the page.
5. Add a new list item selection which will be a dropdown selection for the user. Add the choices within the selection.

```
function updaterForm1(){
 const id = '1bo5jc4APUEwAs_222aswIfPDcvU7ZEgjcuhDcpqXOiU';
 const form = FormApp.openById(id);
 const que3 = form.addMultipleChoiceItem();
 que3.setTitle('What app do you prefer?');
 que3.setChoiceValues(['Docs','Sheets','Slides']);
 //que3.showOtherOption();
 const que4 = form.addCheckboxItem();
 que4.setTitle('Are you enjoying this lesson?');
 que4.setChoices([
   que4.createChoice('Yes',true),
```

```
    que4.createChoice('No',false),
    que4.createChoice('Maybe',false)
  ])
  form.addPageBreakItem().setTitle('Page 2');
  const que5 = form.addListItem();
  que5.setTitle('Do you like?')
    .setChoices([
      que5.createChoice('Docs'),
      que5.createChoice('Sheets')
    ])
}
```

How to make a quiz with points using Apps Script.

1. Create a new form file
2. Set the form file to a quiz with form.setIsQuiz(true);
3. Create a new checkbox item in the form add points and set the choices for the selection.
4. Add the feedback with createFeedback() method creating feedback for both correct and incorrect answers.
5. Create a text input field, set points for the field and add validation for the field. Create the validation object and add it to the element question. FormApp.createTextValidation().requireNumberBetween(1,100).build()
6. Create the quiz, then go to your drive and open the quiz to make adjustments if needed.

```
function makeQuiz(){
  const nameForm = 'Quiz 3';
  const form = FormApp.create(nameForm);
  form.setIsQuiz(true);
```

```
 form.setTitle(nameForm);
 const que1 =  form.addCheckboxItem();
 que1.setTitle('What is your fav app?');
 que1.setPoints(10);
 que1.setChoices([
   que1.createChoice('Docs',true),
   que1.createChoice('Sheets',true),
   que1.createChoice('Slides',false),
   que1.createChoice('Drive',false)
 ])
 const cor = FormApp.createFeedback().setText('Your are correct
Docs :)').build();
 const wro = FormApp.createFeedback().setText('Wrong').build();
 que1.setFeedbackForCorrect(cor);
 que1.setFeedbackForIncorrect(wro);

 const que2 = form.addTextItem();
 que2.setTitle('5 + 10?');
 que2.setPoints(5);
 const val =
FormApp.createTextValidation().requireNumberBetween(1,100).build
();
 //.requireNumberEqualTo(15).build();
 que2.setValidation(val);
 Logger.log(form.getPublishedUrl());
}
```

SpreadsheetApp Class and Sheets Service Code examples

The Spreadsheet service allows scripts to create, access, and modify Google Sheets files. You can use sheets to store and hold data for other Apps Script applications. Sheets can serve as a database for data as the content is presented in a table format with rows and columns of cells each containing data.

You can create a new sheet using the SpreadsheetApp Class.

1. Create a new sheet, set the name of the sheet and the number of rows and columns in the create method. SpreadsheetApp.create('NewSheets',30,7)
2. Get the current user email within the Session service. Session.getActiveUser().getEmail()
3. Get the sheet URL and set it as content within a variable named body.
4. Using MailAp send an email with the link to the newly created sheet. MailApp.sendEmail(email,'Sheet',body);

```
function makeSheet() {
 const ss = SpreadsheetApp.create('NewSheets',30,7);
 Logger.log(ss.getUrl());
 const email = Session.getActiveUser().getEmail();
 const url = ss.getUrl();
 const body = `New Sheet at ${url}`;
 MailApp.sendEmail(email,'Sheet',body);
}
```

How to create new sheets checking to see if the sheet exists or not.

Using the spreadsheetApp class you can select all the sheets contained in a spreadsheet. Select and create a sheet if it does not exist. Sheet names must be

unique, if you try to create a sheet name within the spreadsheet that already exists you will see an error.

1. Select a spreadsheet to use for this snippet.
2. Create a loop to iterate 10 times. Set a name for the sheet you want to create. Using the getSheetByName() select the sheet by the created name.
3. Add a condition that will check if the sheet object is null, if it does not exist then a new sheet then insert the sheet with the chosen name into the spreadsheet.
4. Get all the existing sheets in the spreadsheet. Using forEach loop through them and list the names within the Logger.

```
function upSheet1(){
 const id = '1vliRFwXoINpk1Hg8x6Zb7Rgm2MI4yg9ZzKhE3Mp3FMA';
 const ss = SpreadsheetApp.openById(id);
 for(let i=0;i<10;i++){
   const sheetNamer = `S${i+1}`;
   let newName = ss.getSheetByName(sheetNamer);
   if(!newName){
      newName = ss.insertSheet();
      newName.setName(sheetNamer);
   }
 }
 const sheets = ss.getSheets();
 Logger.log(sheets);
 sheets.forEach((sheet,index)=>{
   Logger.log(sheet.getSheetName());
 })
}
```

How to select sheet cells and update content within those cells.

Selecting the range can then use the range to get values from cells as well as set values within cells. Use the range methods to make a selection of cells that you want to update. There are several ways to select the range, be careful that the range dimensions match the content dimension for the array so that data can be added into the selected range.

1. Select a sheet to work with. Append an array of content to the sheet.
2. Get a range by selecting the row and column to return one cell. Set a value for the cell.
3. Using the getRange() select a row, column and the number of rows for a multi dimensional array of range. Set values using the same size array with nested arrays for rows, and array items for the column numbers.
4. Using getRange select a row,column to start and the number of rows, and columns to select in the range. Set the values of those cells matching the dimensions for the selected range.
5. Select and add a value to a range using the notation name
6. Select a separate sheet, and range within the sheet with the name notation. Update the values of those cells.

```
function adder1(){
 const id = '1vliRFwXoINpk1Hg8x6Zb7Rgm2MI4yg9ZzKhE3Mp3FMA';
 const ss = SpreadsheetApp.openById(id);
 const sheet = ss.getSheets()[0];
 const arr = ['Hello','World','Website'];
 sheet.appendRow(arr);
 const range1 = sheet.getRange(1,5);
 range1.setValue('Laurence');
 const range2 = sheet.getRange(2,1,2);
 range2.setValues([[1],[2]]);
```

```
 const range3 = sheet.getRange(4,2,2,3);
 range3.setValues([[1,2,3],[4,5,6]]);
 const range4 = sheet.getRange('E3');
 range4.setValue('Svekis');
 const range5 = sheet.getRange('S6!A1:C3');
 range5.setValues([[1,2,3],[4,5,6],[7,8,9]]);
}
```

How to add content and create additional content for columns of data.

You can append rows of content, setting the content within an array in Apps Script. Using the LanguageApp you can select to translate strings into other languages. Language codes are at https://cloud.Google.com/translate/docs/languages

1. Select a sheet to update from a spreadsheet
2. Create an array of words or phrases
3. Loop through the array contents, using the LanguageApp to translate the array contents into different languages.
4. Append the new rows of content to the sheet.

```
function adder2(){
 const id = '1vliRFwXoINpk1Hg8x6Zb7Rgm2MI4yg9ZzKhE3Mp3FMA';
 const ss = SpreadsheetApp.openById(id);
 const sheet = ss.getSheetByName('S7');
 const arr = ['Coding','Car','Hello','World','Website'];
 arr.forEach((val,ind)=>{
   const sp = LanguageApp.translate(val,'en','es');
   const lv = LanguageApp.translate(val,'en','lv');
   const ko = LanguageApp.translate(val,'en','ko');
   sheet.appendRow([ind+1,val,sp,lv,ko]);
```

```
 })
}
```

How to remove empty sheets from your spreadsheet

Using getDataRange() you can select the entire available set of data within a sheet. This can then be used to get the values, returning all the values within that selected sheet.

1. Select a spreadsheet to get all the sheets within that spreadsheet
2. Loop through all the sheets, get the data from the sheets and the values.
3. Check the values array for a length, if the length is less than 1 then delete the sheet using the selected sheet object.

```
function adder2(){
 const id = '1vliRFwXoINpk1Hg8x6Zb7Rgm2MI4yg9ZzKhE3Mp3FMA';
 const ss = SpreadsheetApp.openById(id);
 const sheet = ss.getSheetByName('S7');
 const arr = ['Coding','Car','Hello','World','Website'];
 arr.forEach((val,ind)=>{
   const sp = LanguageApp.translate(val,'en','es');
   const lv = LanguageApp.translate(val,'en','lv');
   const ko = LanguageApp.translate(val,'en','ko');
   sheet.appendRow([ind+1,val,sp,lv,ko]);
 })
}
```

How to Update Sheets and Spreadsheet content and prepend rows to sheets.

There are many methods that can be used within sheets, we can also use JavaScript methods to do a lot of adjustments when creating code.

How to create a new sheet and get the id to use in the sheet selection.

```
function maker() {
 const ss = SpreadsheetApp.create('test2');
 Logger.log(ss.getId());
}
```

Once your sheet is created, you might want to reorder the sheets, create a sheet with a specific name and use that sheet. You can check to see if the sheet exists, add headings if they do not exist. Also using JavaScript methods and custom JavaScript functions that can help check and create content for your sheets.

1. Select the spreadsheet to use
2. Use a specific sheet by name, check to see if it exists and if it does not create it.
3. Set a sheet as active so that you can reorganize the sheets in the tab menu. You can select all the sheets and then check to see the sheet order, this is array based which means that the index value of the first sheet will be 0. By selecting the sheet names you can check to see if they match, if not you can move the sheet within the Spreadsheet class using the moveActiveSheet(), which can only be done with the sheet you intend to move being the active sheet.
4. Check if the first sheet matches the sheet order that you want.
5. Adding headings to the sheet, create an array of heading values that you want to use. Get the values of the headings array, and select the first row of content from the sheet. Both will be arrays, there are a number of ways to check if the

contents are the same, one way is to convert the arrays into strings and check to see if the string values match. This can be done with JSON.stringify. If the values don't match you can use the range of the first row and set the values to the array.

6. To capitalize content in JavaScript there isn't a built-in method, but you can create one by selecting the content, converting it to lowercase, then selecting the first letter using charAt(0) and converting that to uppercase with toUpperCase(). Then join the string values together with a slice of the string value at position 1 which will remove the first letter from the string value.
7. Sheet Object can append rows to the end of the sheet data, to add the content at the top of the sheet you can use the insert methods. There is a method to insertRowsBefore(2,1) which will allow you to select the row that you want to do the insert and the number of rows you want to insert. To get the range you can select the range and chain the methods together to have a new object that contains the newly inserted rows.
8. Once you have the range that you want to insert, they are blank and you can update the contents of the cells using the setValues() which will write the values with the nested array data. Each row is an array nested within the main array object.
9. You can also insert blocks of nested arrays, by selecting the range and then using the insertRowsBefore() setting a value that matches the dimensions of the object you want to insert. Number of rows and columns for the data to be inserted.
10. In the example we create a function that will generate a random number from 1 to 1000 which can be used to quickly populate values in testing and create the values into the array blocks needed to match the content range.

```
function updater1(){
 const id = '12ShJG5pi-CFGLB6Aw8PXzemqIFKuAcQmHwTg5dREt8s';
 const ss = SpreadsheetApp.openById(id);
 Logger.log(ss.getUrl());
```

```
const testName = 'test1';
let sheet = ss.getSheetByName(testName);
Logger.log(sheet);
if(!sheet){
  sheet = ss.insertSheet();
  sheet.setName(testName);
}
ss.setActiveSheet(sheet);
let first = ss.getSheets()[0];
if(first.getName() != testName){
  ss.moveActiveSheet(0);
  first = ss.getSheets()[0];
}
Logger.log(first.getName());
const headings1 = ['first','second','third','fourth'];
const headings = headings1.map(name => {
  let newValue = name.toLowerCase();
  let firstLetter = newValue.charAt(0).toUpperCase();
  let letters = newValue.slice(1);
  return firstLetter + letters;
  });
const headerRange = sheet.getRange(1,1,1,headings.length);
const headingValues = headerRange.getValues()[0];
if(!arrStringChecker(headings,headingValues)){
  headerRange.setValues([headings]);
}
```

```
 for(let i=0;i<5;i++){
   //sheet.appendRow([i,rNum(),rNum(),rNum()]);
   const ran1 =
sheet.insertRowsBefore(2,1).getRange(2,1,1,headings.length);
   const newData = adderValues(headings.length);
   ran1.setValues([newData]);
 }

 const tempArr = [];
 const rows = 3;
 for(let i=0;i<rows;i++){
   tempArr.push(adderValues(headings.length));
 }
 Logger.log(tempArr);

sheet.insertRowsBefore(2,rows).getRange(2,1,rows,headings.length
).setValues(tempArr);


 if(arrStringChecker(headings,headingValues)){
   Logger.log('match');
   //Logger.log(headings);
   //Logger.log(headingValues);
 }else{
   Logger.log('NO match');
   //Logger.log(headings);
   //Logger.log(headingValues);
```

```
    headerRange.setValues([headings]);
  }
}
```

Spreadsheet range selections and updates of Cells

Selecting the range allows you to update the cells within that selection. A range can be selected as a single cell or a collection of cells. Ranges provide access and can allow you to modify spreadsheet ranges. Ranges can be a single cell in the sheet or a group of adjacent cells within a sheet.

How to select ranges and how to select all content into a default range.

1. Select all the available cells that have data, this will return a block of cells with the dimensions of the last row and the last column of content. Even if cells have no content the getDataRange() returns all the full blocks with the furthest most cells of data.
2. Select a range just from the headings row you can also select this from the full data range object once you have the values in a nested array, using regular array methods you can then manipulate the array data. allData.getValues().slice(1);
3. Select the range you want to add borders too, getRange(2,5,sheet.getLastRow()-1,1) apply a solid border.

```
function sheet1() {
 const id = '12ShJG5pi-CFGLB6Aw8PXzemqIFKuAcQmHwTg5dREt8s';
 const ss = SpreadsheetApp.openById(id);
 const sheet = ss.getSheetByName('test1');
 //Logger.log(sheet);
 const allData = sheet.getDataRange();
```

```
 //Logger.log(allData.getValues());
 const range1 = sheet.getRange(1,1,1,sheet.getLastColumn());
 //Logger.log(range1.getValues()[0]);
 const data1 = allData.getValues().slice(1);
 //Logger.log(data1);
 data1.forEach(row =>{
   //Logger.log(row);
 })
 //Logger.log(allData.getValues()[0]);
 const range2 = sheet.getRange(2,5,sheet.getLastRow()-1,1);
 Logger.log(sheet.getLastRow());
 Logger.log(sheet.getLastColumn());
range2.setBorder(true,true,false,false,false,false,'red',Spreads
heetApp.BorderStyle.SOLID_THICK);
}
```

Adding a block of formulas to a range of multiple cells.

1. Select the sheet you want to use. Calculate the number of rows using the getLastRow() value and subtracting 1 so that you can account for the heading row.
2. Create an array to use to add the formulas into. Loop through all the rows that you want to create, update the values of the string formula, adjusting for each new row to get the content. Add the new formula in an array with a string contained in the array. Even if it's just the one column the value needs to be placed within its own array to set the proper structure for the range.
3. Set the formulas for all the cells in the range, the array with the formulas must match the dimensions of the range cells.

```
function sheet2() {
```

```
 const id = '12ShJG5pi-CFGLB6Aw8PXzemqIFKuAcQmHwTg5dREt8s';
 const ss = SpreadsheetApp.openById(id);
 const sheet = ss.getSheetByName('test1');
 const rows = sheet.getLastRow()-1;
 const range2 = sheet.getRange(2,5,rows,1);
 const sums = [];
 for(let i=0;i<rows;i++){
   const row = i + 2;
   const val = [`=SUM(A${row}:D${row})`];
   sums.push(val);
 }
 range2.setFormulas(sums);
 Logger.log(sums);
}
```

How to select and update the entire range contents of all the cells within the range.

1. Create a function to use that will generate a random hex color. You can use a typical JavaScript function for this.
2. Select the sheet and the range you want to interact with. Using the setBackground() or setFontColor() apply a random color to all the cells within the range. To apply different values you can create an array with the same dimensions as the range and apply the array of values into the colors of the range.
3. Get all of the values in the sheet and randomize them with randomize();
4. Create a separate sheet, add some values to that sheet. Select the values, randomize() them and removeDuplicates()

5. Merge all the cells from the selected range. Once merged the cell range won't randomize anymore since it will only have the one selected value because of the merge.

```
function ranBack(){
     return `#${Math.random().toString(16).substring(2,8)}`;
}
function sheet3() {
 const id = '12ShJG5pi-CFGLB6Aw8PXzemqIFKuAcQmHwTg5dREt8s';
 const ss = SpreadsheetApp.openById(id);
 const sheet = ss.getSheetByName('test1');
 const rows = sheet.getLastRow()-1;
 const range2 = sheet.getRange(2,5,rows,1);
 range2.setBackground(ranBack());
 range2.setFontColor(ranBack());
 const range3 = sheet.getRange(2,1,rows,4);
 Logger.log(range3.getValues());
 range3.randomize();
}

function sheet4(){
 const id = '12ShJG5pi-CFGLB6Aw8PXzemqIFKuAcQmHwTg5dREt8s';
 const ss = SpreadsheetApp.openById(id);
 const sheet = ss.getSheetByName('test2');
 const range = sheet.getDataRange();
 range.randomize();
 range.removeDuplicates();
 range.merge();
```

```
}
```

Conditional Rules Format

Conditional rules can be selected, added and updated with Google Apps Script. These are rules that can be applied to content to add visual cell properties to the content.
Each conditional format rule may contain a single boolean condition.
Class BooleanCondition can access boolean conditions in ConditionalFormatRules.
These are criteria that can be used to evaluate the cell content, and if the value condition equates to true the rule is applied.
More info on Spreadsheet objects.
https://developers.Google.com/apps-script/reference/spreadsheet/

How to add a conditional rule to a range within your sheet.

1. Get all the rules for the sheet object getConditionalFormatRules();
2. Create a rule with conditions using newConditionalFormatRule()
3. Add the new rule into the rules array.
4. Set the conditional rules for the sheet with sheet.setConditionalFormatRules(rules);

```
function updater2() {
 const id = '12ShJG5pi-CFGLB6Aw8PXzemqIFKuAcQmHwTg5dREt8s';
 const ss = SpreadsheetApp.openById(id);
 const sheet = ss.getSheetByName('test1');
```

```
 Logger.log(sheet.getName());
 const range1 = sheet.getRange('A2:C9');
 Logger.log(range1.getValues());
 const con1 = SpreadsheetApp.newConditionalFormatRule()
   .whenNumberBetween(0, 500)
   .setBackground('#dddddd')
   .setRanges([range1])
   .build();
 const rules = sheet.getConditionalFormatRules();
 Logger.log(rules);
 rules.push(con1);
 sheet.setConditionalFormatRules(rules);

}
```

How to see the rules and return values contained within the rules condition.

1. Select the sheet and get all the rules using the sheet.getConditionalFormatRules();
2. Loop through the rules array and get the values

```
function updater3() {
 const id = '12ShJG5pi-CFGLB6Aw8PXzemqIFKuAcQmHwTg5dREt8s';
 const ss = SpreadsheetApp.openById(id);
 const sheet = ss.getSheetByName('test1');
 const rules = sheet.getConditionalFormatRules();
 rules.forEach(rule => {
   Logger.log(rule.getRanges());
```

```
    /*
    const ranges = rule.getRanges();
    for (let i = 0; i < ranges.length; i++) {
      Logger.log(ranges[i].getA1Notation());
      ranges[i].setFontColor('#00ff00');
    }
    */
    const boo = rule.getBooleanCondition();
    Logger.log(rule.getBooleanCondition().getBackground());
    Logger.log(rule.getGradientCondition());
  })
  Logger.log(rules);
}
```

How to clear rules and add new rules

1. Select all the rules from the sheet. To remove all existing rules use sheet.clearConditionalFormatRules();
2. Create a new SpreadsheetApp.newConditionalFormatRule() that can be applied to the sheet rules.
3. Add the new rule to the rules array, then reapply the rules to the sheet. sheet.setConditionalFormatRules(rules);

```
function updater4() {
  const id = '12ShJG5pi-CFGLB6Aw8PXzemqIFKuAcQmHwTg5dREt8s';
  const ss = SpreadsheetApp.openById(id);
  const sheet = ss.getSheetByName('test1');
  sheet.clearConditionalFormatRules();
```

```
 const rules = sheet.getConditionalFormatRules();
 Logger.log(rules);
 const range1 = sheet.getRange('A2:C9');
 const con1 = SpreadsheetApp.newConditionalFormatRule()
   .whenNumberBetween(0, 500)
   .setBackground('#00FF00')
   .setRanges([range1])
   .build();
 rules.push(con1);
 sheet.setConditionalFormatRules(rules);
}
```

How to update a Conditional Format Rule

1. Create a new rule with newConditionalFormatRule()
2. Select the rules array with sheet.getConditionalFormatRules() from the sheet object
3. Using the index value, update the value of the rule object in the array.
4. Set all the rules to the sheet rules object, updating the existing rules with the new object data as a whole. sheet.setConditionalFormatRules(rules)

```
function updater5() {
 const id = '12ShJG5pi-CFGLB6Aw8PXzemqIFKuAcQmHwTg5dREt8s';
 const ss = SpreadsheetApp.openById(id);
 const sheet = ss.getSheetByName('test1');
 const rules = sheet.getConditionalFormatRules();
 const range1 = sheet.getRange('A2:D9');
 const con1 = SpreadsheetApp.newConditionalFormatRule()
   .whenNumberBetween(0, 500)
```

```
    .setBackground('#0000FF')
    .setRanges([range1])
    .build();
 rules[0] = con1;
 sheet.setConditionalFormatRules(rules);
}
```

CalendarApp Methods for Google Calendar

CalendarApp allows Apps Script to read and update Google Calendar. Using the Calendar service you will have access to the user's Google Calendar in addition it can also access additional calendars that the user has permissions for. CalendarApp class can be used to interact with the calendar object update, edit and delete calendar entries and events.

You can add and have multiple calendars within the calendar service for your account. All accounts come with a default calendar which is the email address and name of the user for the Google Account.

How to view all calendars for your account.

1. Using the CalendarApp select all calendars this will return an array of calendar objects. CalendarApp.getAllCalendars();
2.

```
function cal1() {
 const cals = CalendarApp.getAllCalendars();
```

```
 Logger.log(cals);
 cals.forEach(cal => {
   Logger.log(cal.getName());
 })
}
```

How to get events within a date range from the CalendarApp class.

1. Select the calendar you want to use, you can use the default if you do not have the id of the calendar. CalendarApp.getDefaultCalendar();
2. Create search start and end data using the Date() constructor.
3. Set some options for the events response, you can limit the number of responses and apply a search string value to help narrow search results. Options can be used in object format.
4. Using getEvents return matching event objects cal.getEvents(start, end, opt);
5. Loop through the event objects array and get event information, output it into the Logger.
6. Get the guest list from the event, loop through the guests array and get the guest email and status. event.getGuestList();

Date constructor parameter options to create a Date object

new Date()

new Date(value)

new Date(dateString)

new Date(dateObject)

new Date(year, monthIndex)

new Date(year, monthIndex, day)

new Date(year, monthIndex, day, hours)

new Date(year, monthIndex, day, hours, minutes)

new Date(year, monthIndex, day, hours, minutes, seconds)

new Date(year, monthIndex, day, hours, minutes, seconds, milliseconds)

```
function cal2() {
 const cal = CalendarApp.getDefaultCalendar();
 //Logger.log(cal.getName());
 const start = new Date('1/1/2022');
 const end = new Date('1/1/2023');
 //Logger.log(start);
 //Logger.log(end);
 const opt = {
   max: 10,
   search: 'Test'
 };
 const events = cal.getEvents(start, end, opt);
 //Logger.log(events);
 events.forEach(event => {
   Logger.log(event.getTitle());
   Logger.log(event.isOwnedByMe());
   //event.addGuest('gap****ses+10@gmail.com');
   const list = event.getGuestList();
   list.forEach(guest => {
     Logger.log(guest.getEmail());
     Logger.log(guest.getStatus());
   })
   Logger.log(list);
 })
```

```
}
```

How to select and get event information from a set day.

This code snippet will demonstrate how to send out an email to all guests whose status is still invited meaning they have not responded to the guest invite. Invites can only be sent at the time of event creation, all guests added afterwards will get the default of not being sent an invite when added to an event. To send out an email to let a guest know about the event, this can be done using the MailApp service and sending an individual email to them.

1. Get the calendar to use, set a Date value and then getEventsForDay() to return all events for the selected day.
2. Select as the event the first item in the events array. Get the guest list of the users for the event.
3. Loop through the event guest list, get the event details and status of each guest.
4. Check the guest status if its invited then send an email with the event details to them. guest.getStatus() === 'invited'
5. You can add more guests using the guest email address, please note these guests will not be notified by the system of the event in their calendar. addGuest()

```
function cal3() {
 const cal = CalendarApp.getDefaultCalendar();
 const day = new Date('2/17/2022');
 const events = cal.getEventsForDay(day);
 Logger.log(events);
 const event = events[0];
 const list = event.getGuestList();
```

```
 list.forEach(guest => {
   Logger.log(guest.getEmail());
   Logger.log(guest.getStatus());
   const subject = event.getTitle();
   let body = `You have been invited Check you calendar for
${day} ${event.getDescription()}`;
   if (guest.getStatus() === 'invited') {
     MailApp.sendEmail(guest.getEmail(), subject, body);
   }
 })
 //events[0].addGuest('gap****ses+12@gmail.com');
}
```

How to create an event and send invites to guests when creating the event.

The event creation method has advanced options that allow you to send invites, add guests, set a location and description for the event.

1. Select your calendar object
2. Set a start and end date, including time if you are not planning an all day event. The time will adjust depending on the calendar timezone, setting with UTC time will set the event for that timezone and will show in the guests calendars within their own timezone.
3. Add the options options, like description, location, guests, and whether to send the guests invites. Invites can only be sent at the time of creating the event. The description and location can be updated.
4. Create the event using createEvent() createEvent('New Event 2', start, end, opts) Get the ID of the event to use to select the event in the next function.

```
function cal4() {
 const cal = CalendarApp.getDefaultCalendar();
 const start = new Date('2/18/2022 20:00:00 UTC');
 const end = new Date('2/18/2022 22:00:00 UTC');
 const opts = {
   description: 'This is my event',
   location: 'Toronto',
   guests: 'gap****ses+10@gmail.com,gap****ses+8@gmail.com',
   sendInvites: true
 };
 const event = cal.createEvent('New Event 2', start, end, opts);
 Logger.log(event.getId());
}
```

How to select an existing event by ID and how to get the ID of an event.

Once an event is created you can also update the event, below are some common updates to an event object.

1. Select the event, using the day of the event you can list all of the events on that day. getEventsForDay(day);
2. Loop through the event objects and get the titles and ids. Output them into the Logger. Select the id of the event you want to select and update.
3. Get the event by id
4. Set the description of the event event.setDescription('New updated');
5. Set the title of an event event.setTitle('Another Title');
6. Set the color of the event, please note this will show in the color value and in your calendar. event.setColor("8");
7. Add more guests to the event, please note that no invite email will go out. event.addGuest()

Pale Blue ("1"). Pale Green ("2"). Mauve ("3"). Pale Red ("4"). Yellow ("5"). Orange ("6"). Cyan ("7"). Gray ("8"). Blue ("9"). Green ("10"). Red ("11").

```
function cal5() {
 const id = '7lcmefedab3a82sd9n59k0odi8@Google.com';
 const cal = CalendarApp.getDefaultCalendar();
 const day = new Date('2/18/2022');
 const events = cal.getEventsForDay(day);
 Logger.log(events);
 events.forEach(event => {
   Logger.log(event.getId());
   Logger.log(event.getTitle());
   const ev = cal.getEventById(event.getId());
   Logger.log(ev.getColor());
   ev.setTag('test', 'New');
 })
 const event = cal.getEventById(id);
 event.setDescription('New updated');
 event.setTitle('Another Title');
 event.setColor("8");
 event.addGuest('gap****ses+5@gmail.com');
}
```

Apps Script Drive Service DriveApp

Drive service will let scripts create, find, and modify files and folders within the users Google Drive. The DriveApp class allows scripts to interact with files and folders by allowing the script to create, find, and modify files and folders within Google Drive.

MimeType is an Enum (enumeration) that provides access to MIME-type declarations without typing the strings explicitly.
https://developers.Google.com/apps-script/reference/base/mime-type
PLAIN_TEXT - Representation of MIME type for a plain text file (typically .txt).

How to create a folder using DriveApp Class

1. Set the folder name in the parameters, to add the folder to the root folder of your drive you can use the DriveApp class directly, and if no folder is specified the files and folders get created directly in the main drive folder.

```
function maker1() {
 const folder = DriveApp.createFolder('New Folder');
 Logger.log(folder.getId());
}
```

How to select a folder and add new folders into it.

1. Select the folder, this is most commonly done using the id of the target folder.
2. To create a folder you need to specify the name of the folder in the createFolder method. If you want to add it to a specific location, select the folder object using DriveApp class first.
3. Add editors to the folder using addEditor()

```
function maker2() {
 const id = '1XI1d1VGFGICMJMTXAzyAbqpVfO2fLK5K';
```

```
 const main = DriveApp.getFolderById(id);
 for (let i = 0; i < 5; i++) {
   const temp = `Folder ${i + 1}`;
   const folder = main.createFolder(temp);
   Logger.log(folder.getOwner().getEmail());
   folder.addEditor('gap****ses+5@gmail.com');
   Logger.log(folder.getEditors());
 }
}
```

How to loop through a folderIterator object

You can select the starting point for the folder selection within the folder object, which can be selected using the folder id value.

1. Select all the editors of the folder, this will return an array object which can use typical array methods including the forEach() to loop through the contents. Get the email of the editor from the DriveUser object.
2. Get all the folders from a selected folder. This returns a folder Iterator object
3. Using the while loop check if the hasNext() returns true, if it does select the next folder in the list using the next() method.
4. Output the folder names into the Logger execution log output.

```
function sel1() {
 const id = '1XI1d1VGFGICMJMTXAzyAbqpVfO2fLK5K';
 const main = DriveApp.getFolderById(id);
 main.addEditor('gap****ses+9@gmail.com');
 const editors = main.getEditors();
 Logger.log(editors);
 editors.forEach(editor => {
```

```
    Logger.log(editor.getEmail());
  })
  const folders = main.getFolders();
  //Logger.log(folders);
  //Logger.log(folders.hasNext())
  while (folders.hasNext()) {
    const folder = folders.next();
    Logger.log(folder.getName());
  }

}
```

How to create files adding editors to the file that gets created.

Using the creatFile and setting a filename and contents will create a file type which cannot be viewed within the browser. Although still recognized as a file the file type is not set and therefore the file is not able to be opened properly

```
function maker3() {
  const id = '1XI1d1VGFGICMJMTXAzyAbqpVfO2fLK5K';
  const main = DriveApp.getFolderById(id);
  for (let i = 0; i < 5; i++) {
    const temp = `File ${i + 1}`;
    const content = `Laurence Svekis ${i + 1}`;
    const file = main.createFile(temp, content);
    Logger.log(file.getOwner().getEmail());
    file.addEditor('gap****ses+5@gmail.com');
    Logger.log(file.getEditors());
```

```
 }
}
```

Setting a file type with MimeType and file extension in the name

1. Create a file name that you want to use. You can have files and folders with the same name within your drive, although not suggested as human reading of the files names that are the same will be confusing. Add the file extension in the file name that will match the mimetype.
2. Create content using a string value to be added into the file.
3. Create the file, setting a mime type for the file.
4. Get the owner information email and add an editor to the file.

```
function maker4() {
 const id = '1XI1d1VGFGICMJMTXAzyAbqpVfO2fLK5K';
 const main = DriveApp.getFolderById(id);
 for (let i = 0; i < 5; i++) {
   const temp = `File ${i + 1}.txt`;
   const content = `Laurence Svekis ${i + 1}`;
   const file = main.createFile(temp, content,
MimeType.PLAIN_TEXT);
   Logger.log(file.getOwner().getEmail());
   file.addEditor('gap****ses+5@gmail.com');
   Logger.log(file.getEditors());
 }
}
```

How to select files within a folder, get properties and loop through the Fileiterator object.

1. Select the folder you want to get all the field from.
2. Create a variable to hold the object of files Fileiterator object.
3. Using the hasNext() check if there is a next file in the Fileiterator object. Set this condition within a while loop.
4. Get the boolean of the file properties like the trashed and starred values.
5. Check if the file is a text file, if its not then move it into the trash otherwise add a star to the file.

```
function sel2() {
 const id = '1XI1d1VGFGICMJMTXAzyAbqpVfO2fLK5K';
 const main = DriveApp.getFolderById(id);
 const files = main.getFiles();
 while (files.hasNext()) {
   const file = files.next();
   //Logger.log(file.getName());
   //Logger.log(file.getMimeType());
   Logger.log(file.isTrashed());
   Logger.log(file.isStarred());
   Logger.log('text/plain' == MimeType.PLAIN_TEXT)
   if (file.getMimeType() != MimeType.PLAIN_TEXT) {
     file.setTrashed(true);
   } else {
     file.setTrashed(true);
     file.setStarred(false);
   }
 }
}
```

Files and Folders within your Google Drive

How to search for a file with a value that is contained within the file name. If you know the location of the files and want to retrieve files only that have a file name with a given value as part of the name, you can select all the files and then using the string method check if the file names contain the string value.

How to Search files using string methods.

1. Select the folder you want to search.
2. Return all the files within that folder
3. Loop through the files and check the filename which will be returned as a string value. Since it's a string you can use string methods to check if the string includes a value.
4. Apply a condition depending on the results: update the file with a star or move it to trashed.

```
function sel1() {
 const id = '1XI1d1VGFGICMJMTXAzyAbqpVfO2fLK5K';
 const main = DriveApp.getFolderById(id);
 //const files = main.getFilesByName('File');
 const files = main.getFiles();
 Logger.log(files);
 while (files.hasNext()) {
   const file = files.next();
   const fileName = file.getName();
   if (fileName.includes('File')) {
     file.setStarred(true);
   } else {
```

```
      file.setTrashed(true);
    }
    Logger.log(file);
  }
}
```

How to make copies of files and move files within the drive.

1. Select the folder you want to search
2. Get the files as file objects so that you can rename and move the files.
3. Using makeCopy() create a copy of the file, setting a new name and moving it to a target folder.
4. Move the selected file into a folder by its id location. file.moveTo(moveFolder)

```
function sel2() {
 const id = '1XI1d1VGFGICMJMTXAzyAbqpVfO2fLK5K';
 const main = DriveApp.getFolderById(id);
 const moveFolder = main.createFolder('Files');
 const files = main.getFiles();
 let counter = 1;
 while (files.hasNext()) {
   const file = files.next();
   const fileName = file.getName();
   const newNamer = `${counter}-${fileName}`;
   Logger.log(newNamer);
   file.makeCopy(newNamer, moveFolder);
   file.moveTo(moveFolder);
   counter++;
```

```
 }
}
```

How to update the contents of a file, and change the name

1. Select the file folder you want to use to update the containing files.
2. Add a viewer to the file
3. Set the name of the file with the extension .html
4. Set the content of the file as HTML structured content file.setContent(html);
5. Go to the drive location of the file, download it and open the downloaded file in your browser as an html file.

```
function sel3() {
 const id = '1U85PLCGjVZCOUlGW9xX1h9HBcdH6QqTb';
 const main = DriveApp.getFolderById(id);
 const files = main.getFiles();
 let counter = 1;
 while (files.hasNext()) {
   const file = files.next();
   file.addViewer('gap****ses+5@gmail.com');
   file.setName(`New ${counter}.html`);
   Logger.log(file.getMimeType());
   const html = `<h1>Laurence Svekis</h1><div>${counter}</div>`;
   file.setContent(html);
   counter++;
 }
}
```

How to get the id of a file

1. Select the folder with the files, iterate through the file in order to return all the file ids. Each file will have a unique id value. For html and text files you cannot open them directly in the drive so the id is not in the browser as it is with other file types and folders.

```
function sel4() {
 const id = '1U85PLCGjVZCOUlGW9xX1h9HBcdH6QqTb';
 const main = DriveApp.getFolderById(id);
 const files = main.getFiles();
 while (files.hasNext()) {
   const file = files.next();
   Logger.log(file.getId());
 }
}
```

How to get the file editors array

1. Having the id of the file you can now directly select the file object.
2. If you have the name of the file you can search for the file to return it as an object.
3. Return the array of file editors into the Logger

```
function sel5() {
 const fid = '1efWWo8JAph3hZIc3vZoYAToQ2vrWAdb1';
 const id = '1U85PLCGjVZCOUlGW9xX1h9HBcdH6QqTb';
 const files = DriveApp.getFilesByName('New 8.html');
 const file = DriveApp.getFileById(fid);
 while (files.hasNext()) {
   const file = files.next();
```

```
    Logger.log(file.getId());
  }
  Logger.log(files);
  Logger.log(file);
  const editors = file.getEditors();
  Logger.log(editors);

}
```

How to select an html file or text file and turn it into a PDF that gets emailed.

1. Select the file object, then convert the file into a blob. Not all file types can be converted easily.
2. Get the email of the active user, or set a value for the email.
3. Create some html content
4. Using the MailApp service send an html email, add an attachment with the file as a blob and set it as a mime type of PDF.

```
function maker1() {
  const fid = '1efWWo8JAph3hZIc3vZoYAToQ2vrWAdb1';
  const file = DriveApp.getFileById(fid);
  const blob = file.getBlob();
  const email = Session.getActiveUser().getEmail();
  let html = `<h1>New File as PDF</h1>`;
  html += `<div>Your file is at ${file.getUrl()}</div>`;
  MailApp.sendEmail({
    to: email,
    subject: file.getName(),
    htmlBody: html,
```

```
   attachments: [blob.getAs(MimeType.PDF)]
 });
}
```

How to create and add Sheets and Docs files into a drive.

1. Using the DocumentApp class, create a file. Get the id of the file.
2. Once you have the file as a file object you can then use the moveTo to move the file into another folder by selecting the folder as a folder object.
3. Using the SpreadsheetApp class, create a sheet file. Select the id of the file, and get it as a file object using the DriveApp class.
4. Using the moveTo() and set the argument to the value of the destination folder object.

```
function maker3() {
 const id = '1U85PLCGjVZCOUlGW9xX1h9HBcdH6QqTb';
 const main = DriveApp.getFolderById(id);
 const ss = SpreadsheetApp.create('New Sheet');
 const fid = ss.getId();
 const file = DriveApp.getFileById(fid);
 Logger.log(file);
 file.moveTo(main);
}

function maker2() {
 const id = '1U85PLCGjVZCOUlGW9xX1h9HBcdH6QqTb';
 const main = DriveApp.getFolderById(id);
 const doc = DocumentApp.create('Test');
```

```
 const fid = doc.getId();
 const file = DriveApp.getFileById(fid);
 Logger.log(file);
 file.moveTo(main);
}
```

Gmail Service GmailApp Class

Gmail Service allows the Apps Script to send email, create emails, manage labels, select messages and threads in addition to other Gmail activities. The difference between GmailApp and the Mail Service is that the MailApp is a simpler class that only allows the sending of email. If you only want to send an email use the MailApp class, if you want to interact with content in your Gmail account select the GmailApp.

How to get the Gmail inbox threads

1. Using the GmailApp.getInboxThreads() get all the threads in the inbox
2. You can see a count of threads in the inbox GmailApp.getInboxUnreadCount()
3. Using the Thread Object array loop through all the threads and get the messages within each thread.
4. Create a message array and loop through all the messages getting the subject and id and listing them into the log.

```
function emails1() {
 const threads = GmailApp.getInboxThreads();
 Logger.log(GmailApp.getInboxUnreadCount());
 Logger.log(threads);
 threads.forEach((thread) => {
   Logger.log(thread.getMessageCount());
```

```
    const messages = thread.getMessages();
    messages.forEach(message => {
      Logger.log(message.getSubject());
      Logger.log(message.getId());
    })
  })
}
```

How to reply to a message and get message info from the GmailMessage object

1. Select the message object by its id
2. Using forward - send the message forwarding it to a new email.
3. Get the message details and create a string value that contains all the message details.
4. Using reply send all the body contents back in a reply to the message.

```
function emails2() {
 const id = '17e59d8de80a7e6c';
 const message = GmailApp.getMessageById(id);
 Logger.log(message);
 message.forward('gap****ses+25@gmail.com');
 let html = `<div>Subject : ${message.getSubject()}</div>`;
 html += `<div>Plain Body : ${message.getPlainBody()}</div>`;
 //html += `<div>Form: ${message.getFrom()}</div>`;
 html += `<div>ID: ${message.getId()}</div>`;
 //html += `<div>Raw : ${message.getRawContent()}</div>`;
 html += `<div>HTML Body : ${message.getBody()}</div>`;
 Logger.log(html);
 message.reply(html);
```

```
}
```

How to update message object, send it to trash

1. Select the message by its id
2. Get the thread value thread = message.getThread();
3. Get all the messages in a thread object
4. Using moveToTrash() move the message object to trash.

```
function emails3() {
 const id = '17e59d8de80a7e6c';
 const message = GmailApp.getMessageById(id);
 const thread = message.getThread();
 Logger.log(thread.getMessageCount());
 thread.getMessages().forEach(mes => {
   Logger.log(mes.getSubject());
   //mes.moveToTrash();
   Logger.log(mes.isInTrash());
 })

}
```

How to add labels to threads

1. create a label to add to your thread. GmailApp.createLabel('A New Label')
2. Select the thread to apply the label to. You can also select all the account labels and loop through them as an array
3. Add the thread to the label label.addToThread(thread)
4. Move the thread to the inbox thread.moveToInbox()

```
function labelAdd() {
 const label = GmailApp.createLabel('Aa New Label');
 Logger.log(label);
}

function emails4() {
 const id = '17e59d8de80a7e6c';
 const message = GmailApp.getMessageById(id);
 const thread = message.getThread();
 const lName = 'Aa New Label';
 const label = GmailApp.getUserLabelByName(lName);
 const labels = GmailApp.getUserLabels();
 //Logger.log(labels);
 labels.forEach(lab => {
   //Logger.log(lab.getName());
 })
 Logger.log(label);
 label.addToThread(thread);
 //Logger.log(thread.isInTrash());
 thread.moveToInbox();
 //thread.

}

function emails5() {
 const lName = 'Aa New Label';
```

```
 const label = GmailApp.getUserLabelByName(lName);
 const threads = label.getThreads();
 threads.forEach(thread => {
   Logger.log(thread.getId());
 })
}
```

How to add email attachments to your drive with Apps Script.

1. Select the thread object that contains messages you want to check for attachments.
2. Select the messages from the thread, and loop through the messages.
3. Get the attachments from the message, loop through the attachments. Create blobs from the attachments and then create a file with the blob object.

```
function emails6() {
 const id = '17e59d8de80a7e6c';
 const thread = GmailApp.getThreadById(id);
 //Logger.log(thread);
 const messages = thread.getMessages();
 const folder = DriveApp.createFolder('Attachments');
 messages.forEach(message => {
   const att = message.getAttachments();
   //Logger.log(att);
   att.forEach(attached => {
     Logger.log(attached);
     const blob = attached.copyBlob();
     const file = folder.createFile(blob);
   })
```

```
 })
}
```

Slides Service with SlidesApp Class

Slides Service within Apps Script allows you to create, access, and modify Google Slides files. The SlidesApp Class can create, edit , update and open Slides Presentations.

Create a slide presentation to use for the following exercises

```
function slides1() {
 const slide = SlidesApp.create('Slide1');
 Logger.log(slide.getId());
}
```

Add text and insert a shape with text into a new slide

1. Select the presentation and append a new slide to it.
2. Using the slide object add new text with insertText()
3. Add a new shape insert the shape adding a text box with text.
4. Add more text into the shape.

```
function slides2(){
 const id = '1_pBdlYKSQGnt-EaYQNEIeT68h8xGzkWedZAIBFtiQBA';
 const sl = SlidesApp.openById(id);
 Logger.log(sl.getUrl());
 const slide = sl.appendSlide();
 slide.insertTextBox('Laurence Svekis');
```

```
 const shape =
slide.insertShape(SlidesApp.ShapeType.TEXT_BOX,10,100,400,200);
 const textRange = shape.getText();
 textRange.setText('Laurence');
 textRange.insertText(5,'Svekis');
}
```

How to add a web image to your presentation.

1. Select the presentation.
2. Add a new slide with appendSlide()
3. Insert the image using the url of the image.
4. Set a background by selecting the slide background and then setting a hex value for the fill color.

```
function slides3(){
 const id = '1_pBdlYKSQGnt-EaYQNEIeT68h8xGzkWedZAIBFtiQBA';
 const sl = SlidesApp.openById(id);
 const slide = sl.appendSlide();
 const url = 'http://www.discoveryvip.com/img/d.png';
 slide.insertImage(url);
 slide.getBackground().setSolidFill('#00ff00');
 Logger.log(slide.getLayout());
}
```

How to add layouts selecting layouts from the presentation.

1. Select the presentation, and append a new slide.
2. Select from the layouts array a layout to use for the new slide.
3. Set a background color of the slide.
4. Get all the shapes in the slide and add text to them.

```
function slides4(){
 const id = '1_pBdlYKSQGnt-EaYQNEIeT68h8xGzkWedZAIBFtiQBA';
 const sl = SlidesApp.openById(id);
 const layouts = sl.getLayouts();
 const slide = sl.appendSlide(layouts[4]);
 slide.getBackground().setSolidFill('#00ffff');
 const shapes = slide.getShapes();
 Logger.log(shapes);
 shapes.forEach(shape =>{
   shape.getText().setText('Laurence');
 })
 Logger.log(layouts);
}
```

How to set random background colors to your slides.

1. Select the presentation and all the slides in an array
2. Loop through all the slides, add a new textbox within a shape.
3. Generate a random hex value for a color. Select the slide background and then set the fill color of the slide background.

```
function slides5(){
 const id = '1_pBdlYKSQGnt-EaYQNEIeT68h8xGzkWedZAIBFtiQBA';
 const sl = SlidesApp.openById(id);
 const slides = sl.getSlides();
 slides.forEach((slide,index) =>{
   const output = `${index+1} SLIDE`;
```

```
   const temp =
slide.insertShape(SlidesApp.ShapeType.TEXT_BOX,10,100,400,200).g
etText().setText(output);
   const col = `#${Math.random().toString(16).substring(2,8)}`;
   slide.getBackground().setSolidFill(col);
 })
}
```

How to generate images from all the slides in your presentation and save the images to your drive.

In order to access the URLs of your presentation you will need to get the authentication token. Since this exercise requires using the UrlFetchApp to make a fetch request to return the web contents from a URL, it will need to authenticate the fetch request first. Without authentication the response from the page will be the text message that this request is not authenticated. You can connect to your Google Workspace files with the GET method, and send the authorization headers as in the para below.

Below is an example of what the slide presentation and object id will return as a string JSON value from the URL.

```
{height=900.0, width=1600.0,
contentUrl=https://lh4.Googleusercontent.com/TroTwXedtaMw_LL_8Q-
TmXwLwkNl2wngC9TOAkhyzlr1VokBjpwu9PF5dH5WmvvrhAWKaTIK_GSDnoCNQ3T
d8fubPe6fIicp1cV1GUWSQqKGg7nXJbfJA-PrWRw6NMjSO9zwepAIEQTKZ4pYEjo
YELcPWs7vLezw5SFITln1upUCiK5FvEtZJufhHTMB-zdXzXNb4D_Fsb-8BfZmmnW
litODv-YhGlnkYuvFKiLR=s1600}
```

1. Select the slides presentation to use and all the slides in the presentation.

2. Loop through the slides and get each slide object id. This will be needed to create the unique web URL to view the file object data.
3. Create the URL to connect to using the UrlFetchApp.fetch(url1,para) include the headers from the parameters object. The response will be a string value containing the contentUrl property with the URL where the image of the slide is contained.
4. Make a second fetch request for the image to the contentUrl, this will return the image, which then can be converted into a blob with Apps Script.
5. Add the blob as a new PNG file to your drive. This will generate the image that will be saved into your drive as each slide connects to the contentUrl image.

```
function slides6(){
 const id = '1_pBdlYKSQGnt-EaYQNEIeT68h8xGzkWedZAIBFtiQBA';
 const sl = SlidesApp.openById(id);
 const url =
`https://slides.Googleapis.com/v1/presentations/${id}/pages/`;
 const slides = sl.getSlides();
 const para = {
   method: "GET",
   headers: { Authorization: "Bearer " +
ScriptApp.getOAuthToken() },
   contentType: "application/json",
   muteHttpExceptions: true
 };
 slides.forEach((slide,index) =>{
   const obID = slide.getObjectId();
   const url1 = `${url}${obID}/thumbnail`;
   Logger.log(url1);
   const rep = JSON.parse(UrlFetchApp.fetch(url1,para));
```

```
    Logger.log(rep);
    const blob = UrlFetchApp.fetch(rep.contentUrl).getBlob();
    DriveApp.createFile(blob).setName(`Image ${index}.png`)
    Logger.log(blob);
  })
}
```