Apps Script Coding Examples

Google Apps Script is a scripting platform that lets you extend Google Workspace by adding custom functionality to your Google Sheets, Docs, Forms, and Gmail. Google Apps Script is a cloud-based scripting language that provides easy ways to automate tasks across Google products and third-party services.

Top Tips for Google Apps Script

10 tips to help you improve your Google Apps Script coding:

1. Use the Google Apps Script Debugger: Google Apps Script has a built-in debugger that can help you find and fix errors in your code.
2. Test your code with sample data: Before you run your code on real data, test it with a small set of sample data to make sure it works as expected.

3. Use proper naming conventions: Naming your variables and functions in a consistent and descriptive manner will make your code easier to read and understand.
4. Comment your code: Adding comments to your code will help you (and others) understand what your code is doing, especially if it's complex.
5. Use arrays and objects effectively: Arrays and objects are powerful data structures in Google Apps Script. Learn how to use them effectively to simplify your code and make it more readable.
6. Use built-in functions and methods: Google Apps Script has many built-in functions and methods that you can use to perform common tasks. Take advantage of these to save time and make your code more efficient.
7. Avoid hardcoding values: Avoid hardcoding values in your code, such as cell references or URLs, as this can make your code difficult to maintain. Instead, use variables and functions to make your code more flexible.
8. Use error handling: Use error handling techniques to handle unexpected errors and keep your code running smoothly.
9. Write modular code: Write modular code by breaking your code into small, reusable functions that can be used in multiple places.
10. Keep your code organized: Keeping your code organized by using proper indentation, separating code into different functions and sections, and using meaningful variable names will make it easier to read, understand, and maintain.

Converting a Google Document to plain text

https://youtu.be/csnsdM3NuJk

Converting a Google Document to plain text:

```
function convertToPlainText() {
  // Get the active document
  const document = DocumentApp.getActiveDocument();

  // Get the body of the document
  const body = document.getBody();

  // Get the plain text representation of the document
  const plainText = body.getText();

  // Log the plain text to the console
  Logger.log(plainText);
}
```

In this example, the convertToPlainText() function uses the DocumentApp class to convert a Google Document to plain text. The DocumentApp.getActiveDocument() method is used to get the active document, and the document.getBody() method is used to get the body of the document. The body.getText() method is then used to get the plain text representation of the document, which is logged to the console using the Logger.log() method.

Send the text version of the Doc using Apps Script to the email address of the apps script account

```
function converterText(){
  const doc = DocumentApp.openById(ID);
  const body = doc.getBody();  //DocumentBodySection
  const txt = body.getText();
  const email = Session.getActiveUser().getEmail();
```

```
  //Logger.log(email);
  MailApp.sendEmail(email,'My Doc Text',txt);
}
```

Google Sheets add numbers from columns

https://youtu.be/O5PgoPCmcQU

Google Apps Script that adds two numbers and displays the result in a Google Sheets cell:

```
function sumTwoNumbers() {
  var sheet = SpreadsheetApp.getActiveSheet();
  var cell1 = sheet.getRange("A1").getValue();
  var cell2 = sheet.getRange("B1").getValue();
  var result = cell1 + cell2;
  sheet.getRange("C1").setValue(result);
}
```

Explanation:

function sumTwoNumbers(): This is the declaration of the function sumTwoNumbers, which contains the code that will be executed.

var sheet = SpreadsheetApp.getActiveSheet();: This line creates a variable sheet and assigns the active sheet to it using the SpreadsheetApp.getActiveSheet() method.

var cell1 = sheet.getRange("A1").getValue();: This line creates a variable cell1 and assigns the value of cell A1 to it using the getValue() method.

var cell2 = sheet.getRange("B1").getValue();: This line creates a variable cell2 and assigns the value of cell B1 to it using the getValue() method.

var result = cell1 + cell2;: This line creates a variable result and assigns the sum of cell1 and cell2 to it.

sheet.getRange("C1").setValue(result);: This line sets the value of cell C1 to the value of result using the setValue() method.

To run this script, you would need to open a Google Sheet and then open the Apps Script editor by clicking on the "Tools" menu and selecting "Script editor." Then, paste the code into the editor, save the script, and run the sumTwoNumbers() function.

Sheet Data and Cell Values

https://youtu.be/lakZoOLDFfs

Adding numbers together, creating random numbers to populate the sheet data.

F5 | fx | 436

	A	B	C	D	E	F	G	H	I
1	67	19	105	191	3234	3616	1	7233	14466
2	23	99	133	255		953		510953	512416
3	80	51	229	360		720		720720	722160
4	42	62	163	267		433968		534433968	534868470
5	19	45	154	218		436		436436	437308
6	16	80	142	238		476		476476	477428
7	41	85	156	282		564		564564	565692
8	0	37	115	152		304		304304	304912
9	74	82	200	356		712		712712	714136
10	40	81	177	298		596		596596	597788

```
const ID = '132trk8qIk';

function adder(){
```

```
  const sheet =
SpreadsheetApp.openById(ID).getSheetByName('add');
  const data = sheet.getDataRange().getValues();
  data.forEach((row,index) =>{
    let total = 0;
    row.forEach(cell => {
      if(cell){
      total = total + parseInt(cell);
      }
    })
    const rowValue = index+1;
    const range =
sheet.getRange(rowValue,row.length+1,1,1);
    range.setValue(total);
  })
}


function makeNums(){
  const ss = SpreadsheetApp.openById(ID);
  const sheet = ss.getSheetByName('add');
  for(let i=0;i<20;i++){
    const arr = [getRan(),getRan(),getRan()];
    sheet.appendRow(arr);
  }
  Logger.log(sheet);
}

function getRan(){
  return Math.floor(Math.random()*100);
}
```

Add Image to Sheet Cell

https://youtu.be/cbfxoPjqvhs

add an image to a cell in a Google Sheets spreadsheet using Google Apps Script:

```
function addImageToCell() {
  var sheet = SpreadsheetApp.getActiveSheet();
  var imageUrl = "https://www.example.com/image.png";
  var img = UrlFetchApp.fetch(imageUrl).getBlob();

  sheet.insertImage(img, 1,1);
}
```

Explanation:

First, we get a reference to the active sheet using SpreadsheetApp.getActiveSheet().

Next, we get a reference to the cell where we want to insert the image using sheet.getRange("A1"). You can specify any cell you want by changing the argument to getRange().

Then, we specify the URL of the image we want to insert using the imageUrl variable.

We use the UrlFetchApp.fetch() method to retrieve the image data from the specified URL. This method returns a Response object, which we get the binary data (the image) from using getBlob().

Finally, we use the sheet.insertImage() method to insert the image into the specified cell. The first argument is the image data (the blob we obtained in step 4), and the second argument is the cell reference.

```
function imageAdder(){
  const ss = SpreadsheetApp.openById(ID);
  const sheet = ss.getSheetByName('my');
  const imageURL =
'http://www.discoveryvip.com/img/d.png';
  const img = UrlFetchApp.fetch(imageURL).getBlob();
  sheet.insertImage(img,20,20);
  Logger.log(img);
}
```

New CellImage in Sheets

To add an image into a specific cell this can be done using the newCellImage method.

```
function imageAdderCell(){
  const ss = SpreadsheetApp.openById(ID);
  const sheet = ss.getSheetByName('my');
  const range = sheet.getRange(1,1,3,3);
  const imageURL =
'http://www.discoveryvip.com/img/d.png';
  const image = SpreadsheetApp
  .newCellImage()
  .setSourceUrl(imageURL)
  .setAltTextTitle('discoveryvip')
```

```
  .build();
  range.setValue(image);
}
```

Logging to the console

https://youtu.be/mxCnyr5Ov7w

basic syntax and structure of a Google Apps Script.

```
function helloWorld() {
  // This line logs "Hello, world!" to the console
  Logger.log("Hello, world!");
}
```

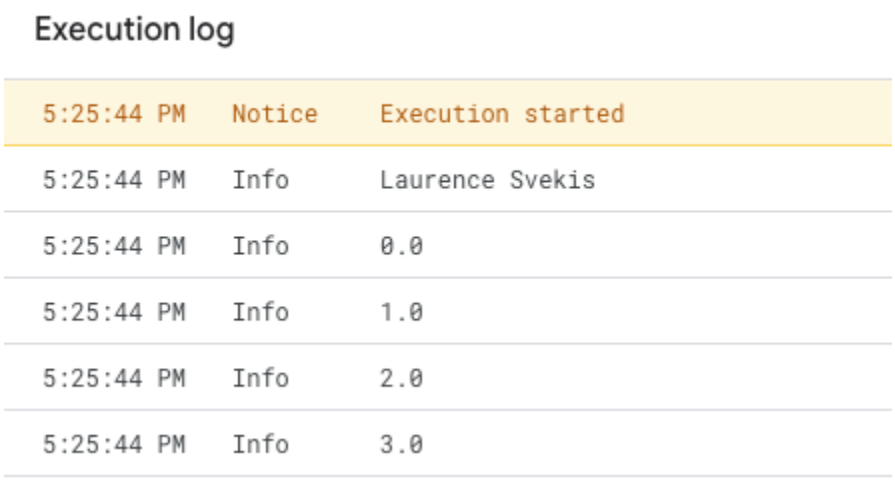
This is a simple function named helloWorld that logs the string "Hello, world!" to the Google Apps Script logging console.

Here's what's happening in this example:

The function keyword declares that this is a function.
The helloWorld function name follows the function keyword.
The code inside the curly braces {} is the body of the function.
The Logger.log statement logs the message "Hello, world!" to the console.
You can run this script by selecting the code and clicking the "Run" button in the Google Apps Script editor, or by pressing Ctrl + Enter (Cmd + Enter on a Mac). When you run this script, the message "Hello, world!" will appear in the logging console.

This is just a simple example, but Google Apps Script provides a wide range of functionality that you can use to automate tasks, manipulate data, and interact with other Google Workspace services. With a little bit of code, you can streamline your work and save yourself a lot of time.

```
function hello(){
  const message = 'Laurence Svekis';
  Logger.log(message);
  for(let i=0;i<20;i++){
    Logger.log(i);
  }
}
```



Adding numbers and showing an alert

Sheets Add Numbers UI Alert.
Google Apps Script that adds up two numbers and displays the result in a dialog box:

```
function addNumbers() {
```

```
  var num1 = 5;
  var num2 = 3;
  var result = num1 + num2;
  SpreadsheetApp.getUi().alert("The result is: " +
result);
}
```

Explanation:

The function addNumbers is defined.
Two variables num1 and num2 are defined and assigned values of 5 and 3 respectively.
The result of num1 + num2 is calculated and stored in the variable result.
The SpreadsheetApp.getUi().alert method is used to display a dialog box with the text "The result is: 8", which is the value of the result variable.
This script can be run in a Google Spreadsheet by opening the script editor (under the "Tools" menu), copying and pasting the code into the editor, and then clicking the "Run" button

```
function showAl(){
  const num1 = 100;
  const num2 = 144;
  const result = num1 + num2;
  const output = `The result of ${num1} + ${num2} =
${result}`;
  SpreadsheetApp.getUi().alert(output);
}
```