Guide to coding apps script

Increase the power of your favorite Google apps — like Calendar, Docs, Drive, Gmail, Sheets, and Slides.

Apps Script lets you do more with Google, all on a modern JavaScript platform in the cloud. Build solutions to boost your collaboration and productivity.

https://developers.google.com/apps-script

Google Apps Script Home

Apps Script is a rapid application development platform that makes it fast and easy to create business applications that integrate with G Suite.

https://script.google.com/home/

https://www.google.com/script/start/

Google Drive

https://drive.google.com/drive/my-drive

Apps Script Permissions

https://myaccount.google.com/permissions?pli=1

Standalone Scripts

https://developers.google.com/apps-script/guides/standalone

A standalone script is any script that is not bound to a Google Sheets, Docs, Slides, or Forms file or Google Sites. These scripts appear among your files in Google Drive.

Creating a standalone script

The easiest way to create a standalone script is to visit script.google.com and at the top left, click addNew project.

You can also create standalone scripts from Google Drive after a few steps of setup:

Go to Google Drive and click New > More > Connect more apps.

When the "Connect apps to Drive" window appears, type "script" into the search box and press Enter.

Click Connect next to the listing for Google Apps Script.

Now that you've connected the app, you can create a script by selecting New > More > Google Apps Script.

Container-bound Scripts

https://developers.google.com/apps-script/guides/bound

A script is bound to a Google Sheets, Docs, Slides, or Forms file if it was created from that document rather than as a standalone script. The file a bound script is attached to is referred to as a "container". Bound scripts generally behave like standalone scripts except that they do not appear in Google Drive, they cannot be detached from the file they are bound to, and they gain a few special privileges over the parent file.

Note that scripts can also be bound to Google Sites, but these scripts are almost always deployed as web apps. Scripts bound to Sheets, Docs, Slides, or Forms can also become web apps, although this is uncommon.

Creating a bound script

To create a bound script, open a Google Sheets, Docs, Slides, or Forms file, then select **Tools > Script editor**. To reopen the script in the future, do the same thing. Because bound scripts do not appear in Google Drive, that menu is the only way to find or open the script.

Special methods

Bound scripts can call a few methods that standalone scripts cannot:

getActiveSpreadsheet(), getActiveDocument(), and getActiveForm() allow bound scripts to refer to their parent file without referring to the file's ID.

getUi lets bound scripts access the user interface for their parent file to add custom menus, dialogs, and sidebars.

In Google Sheets, getActiveSheet(), getActiveRange(), and getActiveCell() let the script determine the user's current sheet, selected range of cells, or selected individual cell. setActiveSheet(sheet) and setActiveRange(range) let the script change those selections.

In Google Docs, getCursor() and getSelection() let the script determine the position of the user's cursor or selected text. setCursor(position) and setSelection(range) let the script change those locations.

Apps Script Code Examples

https://github.com/googleworkspace/apps-script-samples

```
function myFunction1() {
   const doc = DocumentApp.create('Test 1');
}

function myFunction2() {
   const doc = DocumentApp.create('Test 2');
   const body = doc.getBody();
   Logger.log(body);
   body.appendParagraph('Hello World');
}

function myFunction3() {
   const id = '1Rznazp0sPf9eSxFmHRHrWFK0H4lmju6FXDbuI3gZwLk';
   const doc = DocumentApp.openById(id);
   const body = doc.getBody();
   const para = body.appendParagraph('Hello World 2');
   para.appendText('new text');
   Logger.log(para);
}


function myFunction4() {
   const email = Session.getActiveUser().getEmail();
   Logger.log(email);
   const id = '1Rznazp0sPf9eSxFmHRHrWFK0H4lmju6FXDbuI3gZwLk';
   const doc = DocumentApp.openById(id);
   const body = doc.getBody();
   let emailContent = body.editAsText().getText();
   Logger.log(doc.getName());
   //doc.setName('my email doc');
   const subject = doc.getName();
   const url = doc.getUrl();
   emailContent += ' ' + url;
   GmailApp.sendEmail(email, subject, emailContent);



}
```

```



const TOTAL = 5000;
const newName = 'GoodBye world';
/*  */
//
function doGet() {
   const html = '<h1>GoodBye World</h1>';
   return HtmlService.createHtmlOutputFromFile('index');
   //return HtmlService.createHtmlOutput(html);
   Logger.log(newName);
}

/* function doGetOLD() {
 const str = 'GoodBye World 2';
 return ContentService.createTextOutput(str);
}
*/
function test1() {
   const strName = 'New Doc';
   strName.toLocaleUpperCase();
   const doc = DocumentApp.create(strName);
   doc.getBody().appendParagraph(strName);
   myFunction();
}

function myFunction() {
   Logger.log(newName);
   let a = 6;
   Logger.log(a);
   a++;
   Logger.log(a);
   a++;
   Logger.log(a);
   for (let i = 0; i < 10; i++) {
       a++;
       //a=10;
       console.log(a);
   }
}

function myFunction2() {
   let a = 6;
   Logger.log(a);
}

function myFunction3() {
   let a = 8;
   Logger.log(a);
   Logger.log(newName);
}

function myFunction24() {
   let a = 6;
   Logger.log(newName);
   Logger.log(a);
}

function findVidz(q) {
   Logger.log(TOTAL);
   let results = YouTube.Search.list('id,snippet', {
       q: q,
       maxResults: 25
   });
   results.items.forEach(function (item) {
       Logger.log('[%s] Title: %s', item.id.videoId, item.snippet.title);
   });
   return results.items;
}

function myFunction() {
   const results = vid.findVidz('cats');
   //Logger.log(results);
   const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName('youtube');
   results.forEach((el, index) => {
       const jsObj = JSON.parse(el);

       sheet.appendRow([jsObj.snippet.title]);
       Logger.log(el);
   })

}
```