

://WOMEN IN
TECH SEO/

INNOVATE



Google Apps Script: Using Google
Sheets Hidden Superpower

Johanna Maier

Senior SEO Consultant
DEPT



Files



No functions

Execution log



Code.gs

```
1 function myFunction() {  
2  
3 }  
4
```



Libraries



Services



Women in Tech SEO Festival

07 June 2024 | Berlin

Google Apps Script Using Google Sheets Hidden Superpower

Agenda

01 Why use Google Apps Script?

02 Environment & Functions

03 Flow, Debugging & Prompting

04 Appendix: Checklists & Resources



- 2019: SEO traineeship
- Lots of data wrangling in Google Sheets.
- 2021: coding bootcamp
- Automation fangirl

Johanna Maier

Senior SEO Consultant at DEPT®



01

Why use Google Apps Script?

I love Google Sheets.

But **formulas** also have their
limitations.



automation tools you might know already



Google Apps Script



Microsoft®
Excel VBA





no-code

vs.

low-code (🚀 ChatGPT)




Google Apps Script



Microsoft®
Excel VBA



Google Apps Script 
unlocks the full potential of
Google's Workspace or any
tools/sources with an API.

You can access Google Apps Script from many places.

The image displays four screenshots illustrating how to access Google Apps Script from various Google Workspace applications:

- Google Sheets:** The 'Extensions' menu is open, and 'Apps Script' is highlighted in the dropdown list.
- Google Docs:** The 'Extensions' menu is open, and 'Apps Script' is highlighted in the dropdown list.
- Google Slides:** The 'Extensions' menu is open, and 'Apps Script' is highlighted in the dropdown list.
- Google Classroom:** The 'Send' button is open, and 'Script editor' is highlighted in the dropdown list.

Google Apps Script is for
'non-traditional programmers'
or 'citizen developers'.

Sources: medium.com/@Nontechpreneur & workspace.google.com

Don't you often already use
complex formulas
in your workflows?



```

=TRIM(
  SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE(
  SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE(
  SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE(
  JOIN( " ",
    ARRAYFORMULA(
      IF(
        RIGHT(SORT(TRANSPOSE(SPLIT(lower(trim(A1)), " "))),1)="s",
        LEFT(
          SORT(TRANSPOSE(SPLIT(LOWER(TRIM(A1)), " "))),
          LEN(SORT(TRANSPOSE(SPLIT(LOWER(TRIM(A1)), " "))))-1
        ),
        SORT(TRANSPOSE(SPLIT(A1, " ")))
      )
    )
  )
)
, " las", " "), "las ", " "), " los", " "), "los ", " "), "á", "a"), "é", "e"), "ó", "o"), "ú", "u"), "í", "i"), "ñ", "n"),
"&", " "), "and ", " "), " and", " "), " de", " "), "de ", " "), " la", " "), "la ", " "), " el", " "), "el ", " "),
" para", " "), "para ", " "), " ", " "), "à", "a"), "è", "e"), "ò", "o"), "ù", "u"), "ì", "i"), " ", " ")
)

```

Difficult to understand or update.

```

=TRIM(
  SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE(
  SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE(
  SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE( SUBSTITUTE(
  JOIN( " ",
    ARRAYFORMULA(
      IF(
        RIGHT(SORT(TRANSPOSE(SPLIT(,TRIM(A1)), " "))),1)="s",
        LEFT(
          SORT(TRANSPOSE(SPLIT(,TRIM(A1)), " "))),
          LEN(SORT(TRANSPOSE(SPLIT(,TRIM(A1)), " ")))-1
        ),
        SORT(TRANSPOSE(SPLIT(A1, " ")))
      )
    )
  )
)
, " las", " "), "las ", " "), " los", " "), "los ", " "), "á", "a"), "é", "e"), "ó", "o"), "ú", "u"), "í", "i"), "ñ", "ñ"),
"&", " "), "and ", " "), " and", " "), " de", " "), "de ", " "), " la", " "), "la ", " "), " el", " "), "el ", " "),
" para", " "), "para ", " "), "i", "i"), "à", "a"), "è", "e"), "ò", "o"), "ù", "u"), "ì", "ì"), " " , " " )
)

```



Formulas work fine in many cases. When does a script work better?

When can a script provide value over formulas?

1) Avoid messy formulas

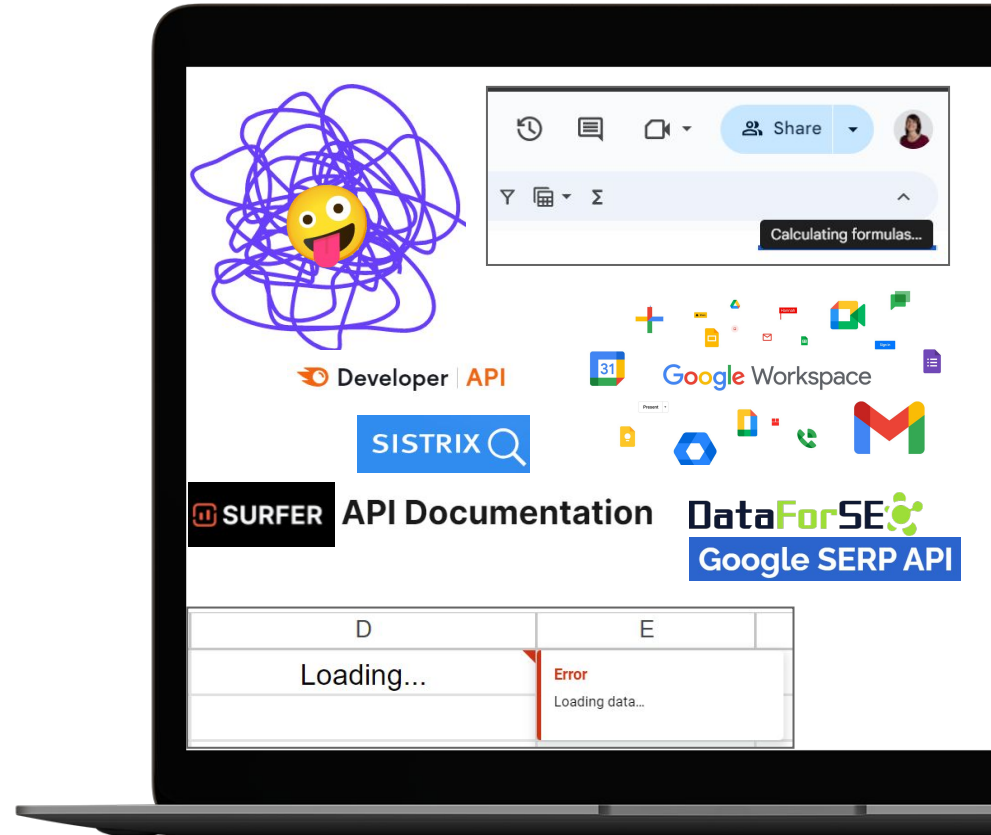
- Scripts are easier to read & debug.
- Updates are more straightforward.

2) Avoid long loading times

- Formulas run instantly / on open.
- Scripts can run selectively and perform better with large data.

3) Integrate Google Workspace & Tool APIs

- Scripts send data between Google files.
- Scripts pull in data from tool APIs.





'Create new **content briefings** from keywords in Google Sheets.'



 SURFER



'Upload **keyword targeting data** to a **BigQuery** project.'

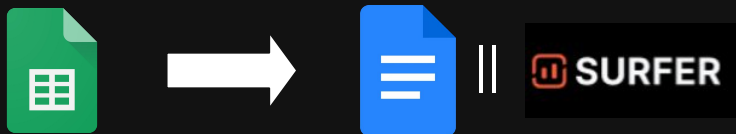


'Pull data from a tool **API** and write it to a Google Sheet.'



USE CASE

Content Briefing Creation at Scale



Situation


You want to transfer keyword targeting info into your preferred content briefing format.

Approach

- Send keywords to a tool API (e.g. SurferSEO) to create briefings in bulk.
- Create briefings from a Google Docs template and fill placeholders with briefing info.

Script Example

bit.ly/dept-wtsfest-berlin

 surfer-seo-briefings.gs

 surfer-seo-briefings

Content Briefings in Bulk with a Tool API

The image shows two screenshots of a Google Sheet titled "Women in Tech SEO Talk".

Top Screenshot: A custom menu is open, showing options like "Get name and append", "Visualise control flow", "Copy formulas", and "Get HTTP status codes". A new item, "SurferSEO: Create Briefings", is being added to the menu. The spreadsheet below has columns for "Content Status", "URL", "SurferSEO Sharable Briefing Link", "SurferSEO Internal Briefing Link", "Focus Keyword", "KW2", "KW3", "KW4", and "KW5". Row 2 contains data for "2 - SurferSEO API | Create Briefings" with the URL "https://www.visitberlin.de/".

Bottom Screenshot: The "SurferSEO: Create Briefings" menu item is now active. A large blue arrow points from the menu item in the top screenshot to the bottom screenshot. In the bottom screenshot, the "SurferSEO Sharable Briefing Link" and "SurferSEO Internal Briefing Link" columns are populated with links like "https://app.surferseo.com/draft" and "https://app.surferseo.com/drafts/". The "Content Status" column now shows "3 - Sent to SurferSEO for Briefing".

Content Briefings in Bulk with Docs Template

The image shows a Google Sheet on the left and a Docs template on the right. The sheet has columns for 'Keyword Group', 'SV', and 'Briefing'. A blue box highlights the first row of data: 'seo agentur', '4900', and a checked checkbox. A blue arrow points from this row to the Docs template. The template has a title 'Briefing "###MAIN_KEYWORD###"' and a table with fields for 'Main Keyword', 'Search Volume', 'Content Format', and 'Words'. The second instance of the template shows the 'seo agentur' row from the sheet populated into the template fields.

| Keyword Group | SV | Briefing |
|---------------------|------|-------------------------------------|
| seo agentur | 4900 | <input checked="" type="checkbox"/> |
| seo agentur berlin | 800 | <input checked="" type="checkbox"/> |
| seo agentur hamburg | 600 | <input type="checkbox"/> |
| seo agentur köln | 600 | <input type="checkbox"/> |
| seo agentur zürich | 100 | <input checked="" type="checkbox"/> |
| seo agentur kosten | 80 | <input checked="" type="checkbox"/> |
| seo agentur schweiz | 40 | <input type="checkbox"/> |

Briefing "###MAIN_KEYWORD###"

| | |
|----------------|---------------------|
| Main Keyword | ###MAIN_KEYWORD### |
| Search Volume | ###SEARCH_VOLUME### |
| Content Format | |
| Words | |

seo agentur Briefing

| | |
|----------------|--------------------|
| Main Keyword | <u>seo agentur</u> |
| Search Volume | 4900 |
| Content Format | |
| Words | |

Source: [Marcus Kästner](#)

USE CASE

Import from Tool APIs: Crawl Data



Situation

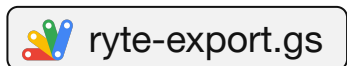
You want to export crawl data from a tool to keep your analysis sheets up-to-date.

Approach

Use a cloud crawling tool with an API connector, schedule your site crawls and write the needed data right into your sheets.

Script Example

bit.ly/dept-wtsfest-berlin



Import Crawl Data via Tool API

The screenshot shows the Ryte Data Explorer interface for 'Deot Berlin GmbH'. The 'Data Explorer' menu is highlighted in blue. The main area displays 'Internal File' with 'Yes (260)' and 'No (133)' filters. A table of 260 results is shown, with columns for Document, OPR, Status Code, Indexability Category, and Clicks. The 'API Call' and 'Data export' options are highlighted in blue in the top right corner of the table.

| Document | OPR | Status Code | Indexability Category | Clicks | |
|------------|-----|-------------|-----------------------|--------|----|
| [Redacted] | 21 | 200 | Indexable | 165 | |
| [Redacted] | 21 | 200 | Indexable | - | OK |
| [Redacted] | 57 | 200 | Indexable | 796 | OK |
| [Redacted] | 21 | 200 | Indexable | 1 | OK |
| [Redacted] | 80 | 200 | Indexable | 11 | OK |

Import Crawl Data via Tool API

The image shows a screenshot of a web interface for configuring an API call. On the left, a window titled "API Call" displays a JSON object with fields for "action", "crawl", "project", "filter", and "pagination". The "project" field is highlighted with a blue box, and a blue arrow points to the right. On the right, a code editor shows the corresponding JSON payload, with the "authentication" block (containing "api_key" and "project") also highlighted with a blue box. The "project" field in the payload is redacted with a brown box.

```
API Call
```

This is the API Call for the selected object

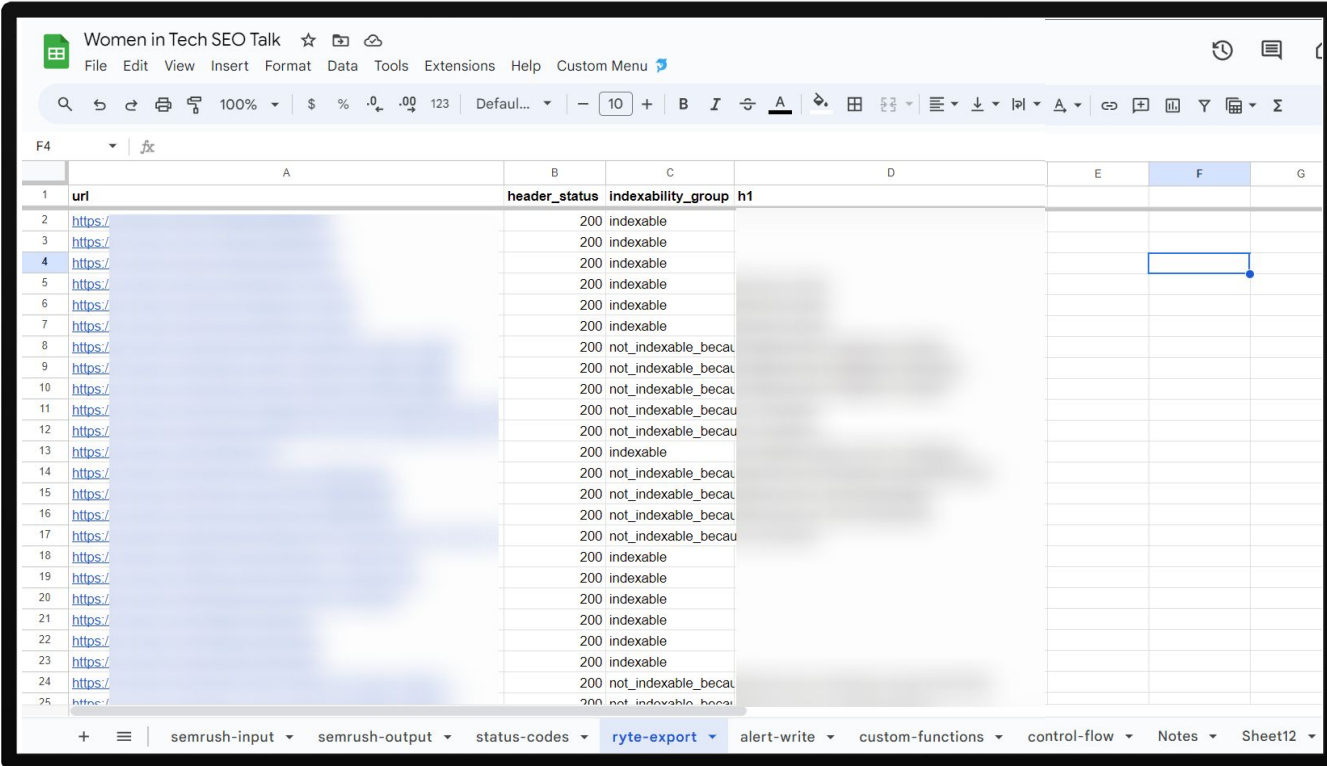
```
{
  "action": "list",
  "crawl": "pull",
  "project": "XXXXXXXXXX",
  "filter": {
    "AND": [
      {
        "field": "is_local",
        "operator": "==",
        "value": 1
      }
    ]
  },
  "pagination": {
```

Copy & Close

```
const endpointRyte = "https://api.ryte.com/zoom/json";

const payloadCrawl =
{
  "action": "list",
  "crawl": "pull",
  "authentication": {
    "api_key": "XXXXXXXXXX",
    "project": "XXXXXXXXXX"
  },
  "filter": {
    "AND": [
      {
        "field": "is_local",
        "operator": "==",
        "value": 1
      },
      {
        "field": "url_type_id",
        "operator": "==",
        "value": 1
      }
    ]
  },
  "pagination": {
```

Import Crawl Data via Tool API



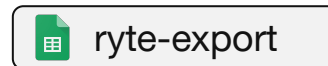
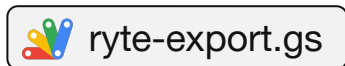
Women in Tech SEO Talk

File Edit View Insert Format Data Tools Extensions Help Custom Menu

100% 123 Default... 10

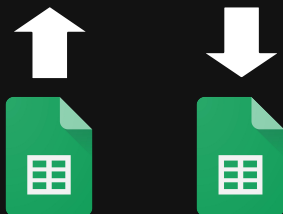
| | A | B | C | D | E | F | G |
|----|---------------------------------|---------------|-----------------------|---|---|---|---|
| 1 | uri | header_status | indexability_group h1 | | | | |
| 2 | https:// | 200 | indexable | | | | |
| 3 | https:// | 200 | indexable | | | | |
| 4 | https:// | 200 | indexable | | | | |
| 5 | https:// | 200 | indexable | | | | |
| 6 | https:// | 200 | indexable | | | | |
| 7 | https:// | 200 | indexable | | | | |
| 8 | https:// | 200 | not_indexable_becau | | | | |
| 9 | https:// | 200 | not_indexable_becau | | | | |
| 10 | https:// | 200 | not_indexable_becau | | | | |
| 11 | https:// | 200 | not_indexable_becau | | | | |
| 12 | https:// | 200 | not_indexable_becau | | | | |
| 13 | https:// | 200 | indexable | | | | |
| 14 | https:// | 200 | not_indexable_becau | | | | |
| 15 | https:// | 200 | not_indexable_becau | | | | |
| 16 | https:// | 200 | not_indexable_becau | | | | |
| 17 | https:// | 200 | not_indexable_becau | | | | |
| 18 | https:// | 200 | indexable | | | | |
| 19 | https:// | 200 | indexable | | | | |
| 20 | https:// | 200 | indexable | | | | |
| 21 | https:// | 200 | indexable | | | | |
| 22 | https:// | 200 | indexable | | | | |
| 23 | https:// | 200 | indexable | | | | |
| 24 | https:// | 200 | not_indexable_becau | | | | |
| 25 | https:// | 200 | not_indexable_becau | | | | |

semrush-input semrush-output status-codes **ryte-export** alert-write custom-functions control-flow Notes Sheet12



USE CASE

Bulk Keyword Reports



Situation


You need 'Related Queries' reports from SEMrush to expand a list of seed keywords.


Approach


Use the SEMrush developer API to append keyword reports for each seed keyword to one output sheet.

Script Example

bit.ly/dept-wtsfest-berlin

 `semrush-keywords.gs`

 `semrush-input`

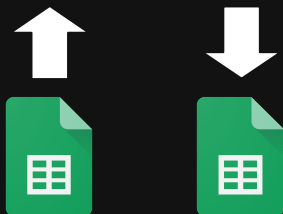
 `semrush-output`

Keyword Expansion in Bulk with Tool API

| SEMrush Related Keywords Output | | | | | | | |
|---------------------------------|-------------------|------|-------------|---------------|----------|----------------|------------------------|
| Keyword | Number of Results | CPC | Competition | Search Volume | Trends | Related Releva | Seed Keyword Input |
| restaurant in berlin | 181000000 | 0.31 | 0.14 | 1000 | 0.37,0.5 | 0.55 | top restaurants berlin |
| restaurant s | 1600000000 | 1.74 | 0.02 | 1000 | 0.44,0.2 | 0.05 | top restaurants berlin |
| restaurant tipps berlin | 16700000 | 0.2 | 0.08 | 1000 | 0.36,0.3 | 0.5 | top restaurants berlin |
| restaurants am see | 918000000 | 0 | 0.02 | 1000 | 0.20,0.2 | 0.05 | top restaurants berlin |
| restaurants berlin am wasser | 9530000 | 0.34 | 0.08 | 1000 | 0.04,0.0 | 0.05 | top restaurants berlin |
| restaurants in | 0 | 1.95 | 0.01 | 1000 | 0.31,0.4 | 0 | top restaurants berlin |
| sehenswürdigkeiten | 84300000 | 0.37 | 0.02 | 301000 | 0.29,0.4 | 0.05 | top 10 berlin |
| berlin sehenswürdigkeiten | 20200000 | 0.21 | 0.72 | 33100 | 0.44,0.5 | 0.25 | top 10 berlin |
| sehenswürdigkeiten berlin | 24000000 | 0.18 | 0.68 | 22200 | 0.54,0.6 | 0.25 | top 10 berlin |
| ausflug mit kindern | 13400000 | 0.45 | 0.16 | 6600 | 0.44,0.3 | 0.05 | top 10 berlin |
| sehenswuerdigkeiten | 654000 | 0.37 | 0.02 | 6600 | 0.13,0.1 | 0.1 | top 10 berlin |
| unternehmungen mit kindern | 6030000 | 0.47 | 0.17 | 6600 | 0.54,0.5 | 0.05 | top 10 berlin |
| berlin mit kindern | 151000000 | 0.4 | 0.73 | 5400 | 0.81,0.8 | 0.05 | top 10 berlin |
| was kann man in berlin machen | 493000000 | 0.21 | 0.94 | 5400 | 0.66,0.8 | 0.05 | top 10 berlin |
| kinderaktivitäten | 924000 | 0.54 | 0.09 | 4400 | 0.44,0.6 | 0.05 | top 10 berlin |
| sehenswürdigkeiten in berlin | 27600000 | 0.25 | 0.66 | 4400 | 0.29,0.3 | 0.25 | top 10 berlin |
| aktivitäten mit kindern | 52700000 | 0.43 | 0.14 | 3600 | 0.65,0.6 | 0.05 | top 10 berlin |
| kinder aktivitäten | 189000000 | 0 | 0 | 3600 | 0.66,0.6 | 0.05 | top 10 berlin |
| mögliche aktivitäten in berlin | 0 | 0.51 | 0.94 | 3600 | 0.07,0.0 | 0.05 | top 10 berlin |
| sehenswürdigkeit | 44000000 | 0.37 | 0.02 | 3600 | 0.16,0.2 | 0.05 | top 10 berlin |
| sightseeing berlin | 20500000 | 0.41 | 0.61 | 2900 | 0.44,0.5 | 0.1 | top 10 berlin |

USE CASE

Custom SERP Analysis

DataForSEO**Situation**

SERP layouts are constantly changing. To help topic prioritisation, you want to scrape SERP data incl. all its layout components.

Approach

Use the DataForSEO SERP API to scrape SERP info & to calculate custom metrics like pixel rank.

Script Example

Google Sheet: bit.ly/dept-pixel-rank

Campixx Talk:

<https://speakerdeck.com/johannamaier/pixel-rank-the-better-metric-for-your-ranking>

Custom SERP Analysis with DataForSEO

The screenshot shows the DataForSEO Pixel Rank & SERP Features Extractor interface. The interface is a spreadsheet with various settings and data columns. Annotations highlight key areas:

- Region, Language & Domain:** Points to the 'DataForSEO Settings' section, which includes fields for Region (Germany), Language (German), and Domain (www.visitberlin.de).
- Keyword List:** Points to the 'Your Keyword List' section, which contains a list of keywords such as 'berlin sehenswürdigkeiten', 'märkte berlin', 'escape room berlin', etc.
- Script Menu:** Points to the 'DataForSEO' menu item in the top right corner.
- Device & API Credentials:** Points to the 'Mobile or Desktop?' and 'Username/Password' fields, which are used to configure the device and API access.

The spreadsheet displays the following data:

| 1 | DataForSEO Settings | | Selection | Code | Is a trigger running? | Max | Calculation for N° of Keywords |
|----|--|--------------------------|---------------------------|---------------------------|--|---------------------|--------------------------------|
| 2 | Region | Germany | | 2276 | OK | No triggers running | Will you also query CPC data? |
| 3 | Language | German | | | | | |
| 4 | Domain (incl. subdomain if present) | www.visitberlin.de | | | | | |
| 5 | Force Filter = 0? | <input type="checkbox"/> | | | | | |
| 6 | Force Filter = 0? | <input type="checkbox"/> | | | | | |
| 7 | Your Keyword List (Please remove any duplicates, empty cells or whitespace around keywords before or multiple whitespaces.) | Checked? | Checked Keywords | Remaining Keywords | List all "not relevant" domains here including subdomains. (These will be skipped when calculating the "Pixel To First Relevant Ranking" metric.) | | |
| 8 | berlin sehenswürdigkeiten | <input type="checkbox"/> | berlin sehenswürdigkeiten | berlin sehenswürdigkeiten | www.wikipedia.org | | |
| 9 | märkte berlin | <input type="checkbox"/> | märkte berlin | märkte berlin | de.wikipedia.org | | |
| 10 | escape room berlin | <input type="checkbox"/> | escape room berlin | escape room berlin | en.wikipedia.org | | |
| 11 | corona regeln berlin | <input type="checkbox"/> | corona regeln berlin | corona regeln berlin | www.berlin.de | | |
| 12 | alexanderplatz | <input type="checkbox"/> | alexanderplatz | alexanderplatz | www.facebook.com | | |
| 13 | berlin döner | <input type="checkbox"/> | berlin döner | berlin döner | www.instagram.com | | |
| 14 | tierpark berlin | <input type="checkbox"/> | tierpark berlin | tierpark berlin | www.twitter.com | | |
| 15 | aktivitäten berlin | <input type="checkbox"/> | aktivitäten berlin | aktivitäten berlin | www.pinterest.com | | |
| 16 | berlin mit kindern | <input type="checkbox"/> | berlin mit kindern | berlin mit kindern | www.dict.cc | | |
| 17 | spandau bei berlin | <input type="checkbox"/> | spandau bei berlin | spandau bei berlin | dict.leo.org | | |
| 18 | radtouren berlin | <input type="checkbox"/> | radtouren berlin | radtouren berlin | www.merriam-webster.com | | |
| 19 | berliner seen | <input type="checkbox"/> | berliner seen | berliner seen | duden.de | | |
| 20 | top restaurants berlin | <input type="checkbox"/> | top restaurants berlin | top restaurants berlin | wiktionary.org | | |
| 21 | pizza berlin | <input type="checkbox"/> | pizza berlin | pizza berlin | | | |
| 22 | staatsoper berlin | <input type="checkbox"/> | staatsoper berlin | staatsoper berlin | | | |
| 23 | berliner stadtteile | <input type="checkbox"/> | berliner stadtteile | berliner stadtteile | | | |

Custom SERP Analysis with DataForSEO

Analysis Examples (Copy of DEPT || Pixel Rank & SERP Features Extractor || DataForSEO) ☆ 🔒 ☁

File Edit View Insert Format Data Tools Extensions Help

100% Default...

Keyword List **SERP Items**

| | Language Used | Location Used | Keyword Used | Device Used | SERP Item Types | Pixel Top 10 Organic |
|----|---------------|---------------|---------------------------|-------------|--|--|
| 1 | German | Germany | berlin sehenswürdigkeiten | desktop | top_sights,people_also_ask,organic,knowledge_graph | 790,958,1104,1272,1418,1586,1754,1900,20 |
| 2 | German | Germany | wittenau | desktop | organic,people_also_ask,knowledge_graph | 182,350,804,950,1208,1354,1500,1646,1792 |
| 3 | German | Germany | buddy bear | desktop | organic,knowledge_graph | 182,596,742,1032,1178,1324,1470,1616,176 |
| 4 | German | Germany | berliner bär | desktop | organic,find_results_on,local_pack,people_also_ask,knowledge_graph | 243,1334,1592,2006,2277,2548,2738,2928,3 |
| 5 | German | Germany | alba berlin | desktop | organic,top_stories,find_results_on,local_pack,twitter,people_also_ask,knowledge_graph | 937,3021,3167,3367,3795,3941,4087,4233,4 |
| 6 | German | Germany | union berlin | desktop | top_stories,organic,twitter,people_also_ask,knowledge_graph | 1380,2516,2942,3086,3284,3428,3572,3716 |
| 7 | German | Germany | hertha bsc | desktop | top_stories,organic,twitter,knowledge_graph | 1292,1796,1920,2066,2234,2380,2526,2726 |
| 8 | German | Germany | wir fahren nach berlin | desktop | organic,knowledge_graph | 243,984,1198,1478,1624,1770,1938,2084,22 |
| 9 | German | Germany | berlin berlin | desktop | organic,people_also_ask,top_stories,knowledge_graph | 182,404,854,1047,1240,1386,1532,1678,182 |
| 10 | German | Germany | berlin wahl | desktop | top_stories,organic,people_also_ask | 984,1412,1558,1704,1962,2108,2254,2400,2 |
| 11 | German | Germany | rotes rathaus | desktop | organic,people_also_ask,images,knowledge_graph,google_reviews | 194,452,956,1124,1598,1766,1934,2102,224 |
| 12 | German | Germany | bvg | desktop | organic,top_stories,twitter,knowledge_graph | 182,1444,1644,1790,1936,2082,2250,2418,2 |
| 13 | German | Germany | mauerweg | desktop | organic,people_also_ask,knowledge_graph | 182,744,890,1058,1204,1350,1496,1642,178 |
| 14 | German | Germany | tegeler see | desktop | organic,people_also_ask,images,top_sights,knowledge_graph,google_reviews | 182,610,756,1609,1755,1901,2069,2215,236 |
| 15 | German | Germany | dept agency berlin | desktop | organic,knowledge_graph,google_reviews | 182,328,496,642,788,934,1080,1226,1394,1 |
| 16 | German | Germany | taxi berlin | desktop | find_results_on,local_pack,organic,people_also_ask | 1089,1785,1953,2099,2245,2391,2537,2683 |
| 17 | German | Germany | teufelsberg | desktop | organic,images,people_also_ask,knowledge_graph,google_reviews | 182,596,1382,1662,1808,1954,2122,2268,24 |
| 18 | German | Germany | konzerte berlin 2023 | desktop | events,organic,people_also_ask | 734,978,1124,1270,1492,1714,2142,2400,25 |
| 19 | German | Germany | u8 berlin | desktop | organic,people_also_ask,knowledge_graph | 182,328,778,924,1070,1216,1438,1631,1777 |
| 20 | German | Germany | schrippe | desktop | organic,people_also_ask,recipes,knowledge_graph | 677,1105,1251,1772,1918,2086,2232,2378,2 |
| 21 | German | Germany | berliner luft | desktop | organic,people_also_ask,images | 182,610,778,946,1092,1350,1630,1776,2272 |
| 22 | German | Germany | museen berlin | desktop | carousel,organic,people_also_ask,top_stories | 507,957,1147,1293,1956,2102,2369,2515,26 |
| 23 | German | Germany | berlin friedrichstraße | desktop | organic,video,people_also_ask,knowledge_graph | 182,328,586,1255,1683,1829,1975,2121,226 |

+ ≡ SERP Layout Details ▾ SERP Layout Overview ▾ Pivot | Tracking over Time ▾ Pivot | Keyword Analysis ▾ Pivot | Above the Fold ▾ Pivot | DE vs. US ▾

Ungefähr 54.600.000 Ergebnisse (0,78 Sekunden)

Ergebnisse für Berlin Region auswählen

Anzeige Pizza in Berlin - Einfach Online Bestellen

Ergebnisse finden auf

Orte



Ristorante Piazza Rossa

L'Osteria Berlin Memharstraße

Trattoria Peretti

Top10 Liste: Pizza | top10berlin

Platz 1: Papa Pane di Sorrento ... Das Papa Pane gilt als bester Italiener in Berlin

11 Berliner Restaurants, in denen ihr richtig gute Pizza ...

1203 pixels

Ungefähr 4.030.000 Ergebnisse (0,61 Sekunden)

Die Top 11 der schönsten Radtouren im Sommer in Berlin

- Top 1: Mauernadweg 1 im Norden,
Top 2: Mauernadweg 2 im Süden,
Top 3: Nikolaasse Route,
Top 4: Fünf-Ödler-Tour im Osten Berlins,
Top 5: Fußball-Route Berlin,
Top 6: Wannsee-Babelsberg Route,
Top 7: Fahrradtour durch Reinickendorf,
Top 8: Fahrradtour am Tiergarten.

Die Top 11 der schönsten Radtouren im Sommer in Berlin

Die 20 schönsten Fahrradtouren rund um Berlin - Komoot

11 schöne Radtouren durch und um Berlin

11 schöne Radtouren am Wasser | Mit Vergnügen Berlin

Rund um Berlin - Regionalparks

Radtouren in Berlin: Die 10 besten Touren der Stadt - GEO

Top10 Liste: Fahrradtouren durch Brandenburg | top10berlin

Reisereporter.de - Inspiration | Radtouren

Radtouren durch Brandenburg: Seen, Wälder, Sternenglück

182 pixels

Ungefähr 17.900.000 Ergebnisse (0,77 Sekunden)

Top-Sehenswürdigkeiten in Berlin



Brandenburger Tor, Berliner Fernsehturm, Berliner Dom

Top 10 Sehenswürdigkeiten in Berlin | visitBerlin.de

- Top 1: Reichstag, Top 2: Brandenburger Tor, Top 3: Fernsehturm, Top 4: Humboldt Forum, Top 5: Gendarmenmarkt, Top 6: Kurfürstendamm, Top 7: Schloss ...

Ähnliche Fragen

- Was muss man gesehen haben in Berlin?, Was sollte man in Berlin nicht vergessen?, Was lohnt sich in Berlin anzusehen?, Was kann man in Berlin machen Tagesausflug?

Sehenswürdigkeiten in Berlin



Die Top-Sehenswürdigkeiten Berlins

Berlin Sehenswürdigkeiten: Top-24-Liste mit Tipps & Bezirken

- 1. Brandenburger Tor, 2. Reichstag, 3. Fernsehturm, 4. Berliner Dom, 5. Checkpoint Charlie, 6. Museumsinsel, 7. East Side Gallery, 8. Oberbaumbrücke.

595 pixels



Berlin Hauptstadt von Deutschland

Berlin ist die Hauptstadt und ein Land der Bundesrepublik Deutschland.

Fläche: 891,8 km², Wetter: 8 °C, Wind aus SW mit 27 km/h, 69 % Luftfeuchtigkeit

Hochschulen und Universitäten Über 45 weitere ansehen

Kommende Veranstaltungen

World Heritage Museumsinsel (in English), One on One: Sandra Mujigas, BM5WR, ...

Check your tool stack:
Do you have **access to APIs**
to speed up any workflows?

USE CASE

On-Demand Execution



Situation

Formulas (or custom functions) are triggered every time a spreadsheet loads.


Long loading times and unwanted credit consumption from API calls.


Approach

- (1) Use a custom menu or a time-based trigger to control when a function is run.
- (2) Use script to turn formulas into text and vice versa. Run via custom menu.

Script Examples

bit.ly/dept-wtsfest-berlin

 status-codes.gs

 status-codes

Execute Functions & Formulas On-Demand

The screenshot shows a Google Sheets spreadsheet with the following data:

| | A | B | C | D | E |
|----|---|---------------------|--------------------------------|---|---|
| 1 | URLs for Status Code Check | Status Codes | Formula: URL Duplicate? | | |
| 2 | https://www.otto.de/random-not-existing-url/ | =HTTPSTATUSCODE(A2) | 1 | | |
| 3 | https://www.otto.de/damen/mode/sportmode | Loading... | 1 | | |
| 4 | https://www.otto.de/damen/mode/sportmode/ | Loading... | 1 | | |
| 5 | https://www.otto.de/damen/mode/sportmode/funktionsblusen/ | Loading... | 2 | | |
| 6 | https://www.otto.de/damen/mode/sportmode/funktionsblusen/ | Loading... | 2 | | |
| 7 | https://www.otto.de/damen/mode/sportmode/funktionswaesche/ | Loading... | 1 | | |
| 8 | https://www.otto.de/damen/mode/hosen/outdoorhosen/ | Loading... | 1 | | |
| 9 | https://www.otto.de/damen/mode/jacken/?anwendungsgebiet=outdoor | Loading... | 1 | | |
| 10 | https://www.otto.de/damen/mode/sportmode/sportanzuege/ | Loading... | 1 | | |
| 11 | https://www.otto.de/herren/mode/hosen/outdoorhosen/ | Loading... | 1 | | |
| 12 | https://www.otto.de/herren/mode/jacken/?anwendungsgebiet=outdoor | Loading... | 1 | | |
| 13 | https://www.otto.de/herren/mode/sportmode/sportanzuege/ | Loading... | 1 | | |
| 14 | https://www.otto.de/herren/mode/sportmode/sportbademode/ | Loading... | 1 | | |
| 15 | https://www.otto.de/herren/mode/hemden/sporthemden/ | Loading... | 1 | | |
| 16 | https://www.otto.de/herren/mode/sportmode/sporthosen/ | Loading... | 1 | | |
| 17 | https://www.otto.de/herren/mode/sportmode/sportpullover/ | Loading... | 1 | | |
| 18 | https://www.otto.de/herren/mode/sportmode/sportshirts/ | 200 | 1 | | |

Execute Functions & Formulas On-Demand

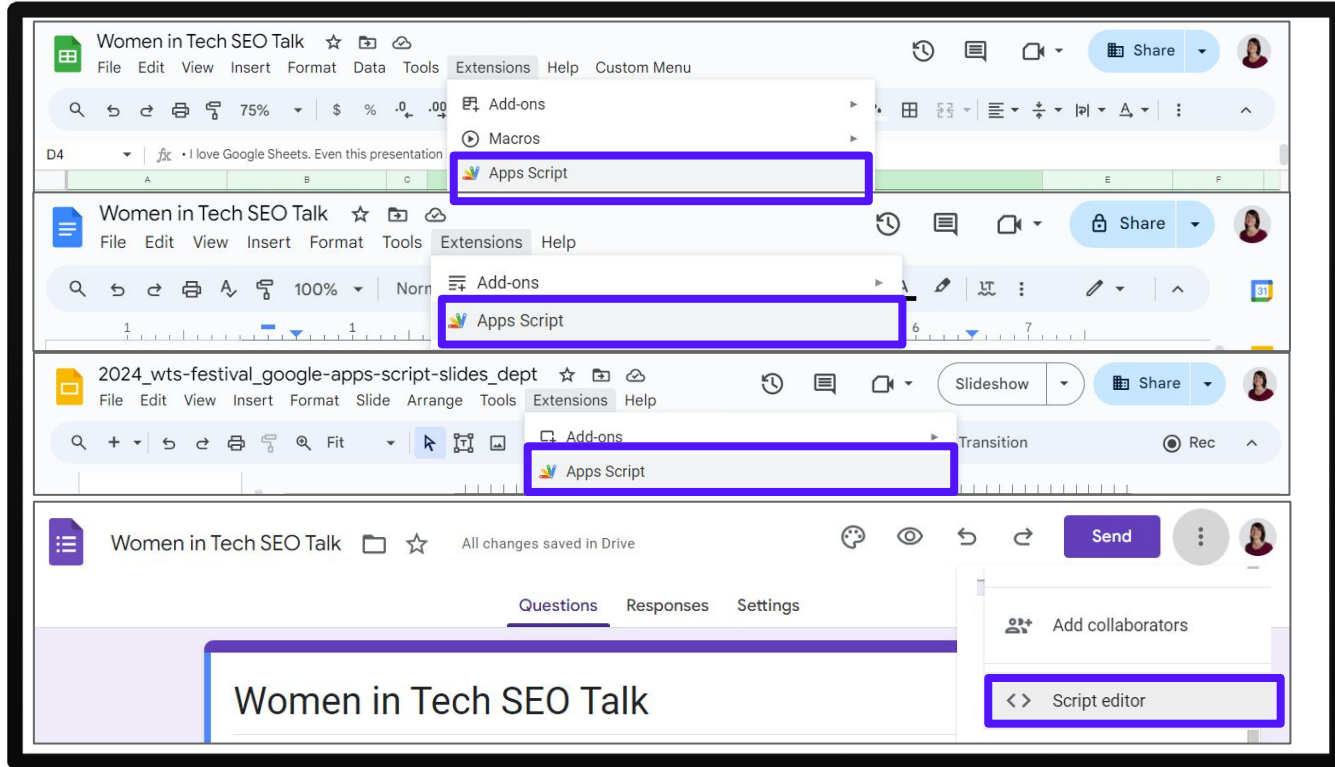
The screenshot shows a Google Sheets spreadsheet titled "Women in Tech SEO Talk". A custom menu is open, displaying several options. Two options are highlighted with blue boxes: "Get HTTP status codes" and "Drag formula down". The spreadsheet contains a table with columns for URLs, status codes, and other data.

| 1 | URLs for Status Code Check | | |
|----|--|-----|---|
| 2 | https://www.otto.de/random-not-existing-url/ | | |
| 3 | https://www.otto.de/damen/mode/sportmode | 301 | 1 |
| 4 | https://www.otto.de/damen/mode/sportmode/ | 200 | 1 |
| 5 | https://www.otto.de/damen/mode/sportmode/funktionsblusen/ | 200 | 2 |
| 6 | https://www.otto.de/damen/mode/sportmode/funktionsblusen/ | 200 | 2 |
| 7 | https://www.otto.de/damen/mode/sportmode/funktionswaesche/ | 200 | 1 |
| 8 | https://www.otto.de/damen/mode/hosen/outdoorhosen/ | 301 | 1 |
| 9 | https://www.otto.de/damen/mode/jacken/?anwendungsgebiet=outdoor | 200 | 1 |
| 10 | https://www.otto.de/damen/mode/sportmode/sportanzuege/ | 200 | 1 |
| 11 | https://www.otto.de/herren/mode/hosen/outdoorhosen/ | 301 | 1 |
| 12 | https://www.otto.de/herren/mode/jacken/?anwendungsgebiet=outdoor | 200 | 1 |
| 13 | https://www.otto.de/herren/mode/sportmode/sportanzuege/ | 200 | 1 |
| 14 | https://www.otto.de/herren/mode/sportmode/sportbademode/ | 200 | 1 |
| 15 | https://www.otto.de/herren/mode/hemden/sporthemden/ | 200 | 1 |
| 16 | https://www.otto.de/herren/mode/sportmode/sporthosen/ | 200 | 1 |
| 17 | https://www.otto.de/herren/mode/sportmode/sportpullover/ | 200 | 1 |
| 18 | https://www.otto.de/herren/mode/sportmode/sportshirts/ | 200 | 1 |
| 19 | | | |

02

Apps Script Environment & Functions

Access to the Apps Script environment

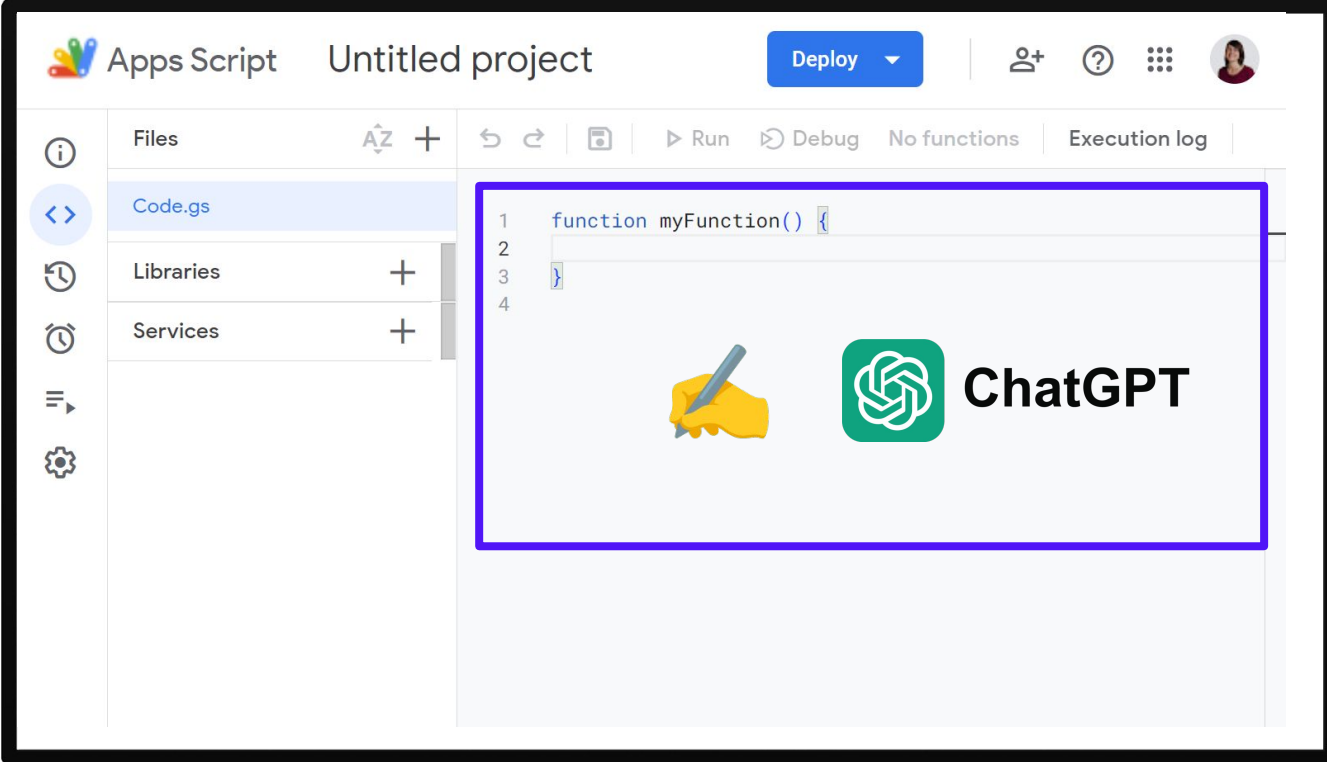


Access to the [Apps Script](#) environment

script.google.com > New project

script.new

All options open up the integrated code editor.



The screenshot displays the Google Apps Script integrated code editor interface. At the top, it shows "Apps Script" and "Untitled project" with a "Deploy" button. The left sidebar contains a "Files" section with "Code.gs" selected, and "Libraries" and "Services" sections with plus signs. The main editor area shows a code editor with a blue border. The code editor contains the following code:

```
1 function myFunction() {  
2  
3 }  
4
```

Overlaid on the code editor is a graphic featuring a yellow hand holding a pencil, the OpenAI logo (a green square with a white knot), and the text "ChatGPT".



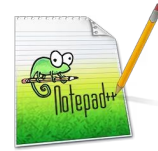
CONCEPT

Integrated Development Environment

'An IDE is a robust tool for writing, editing, debugging, and running your code.'

A text editor only lets you write and edit your code. You might have to step out of a text editor to run your code or download plugins to help it do the running for you.'

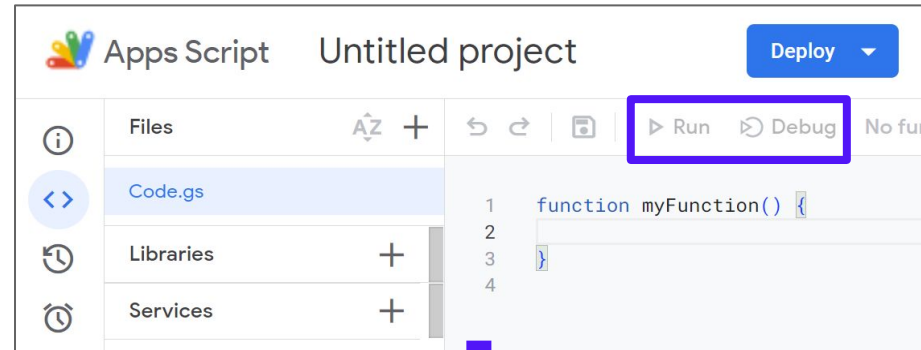
Source: [freeCodeCamp](https://www.freecodecamp.org/)



Visual Studio Code

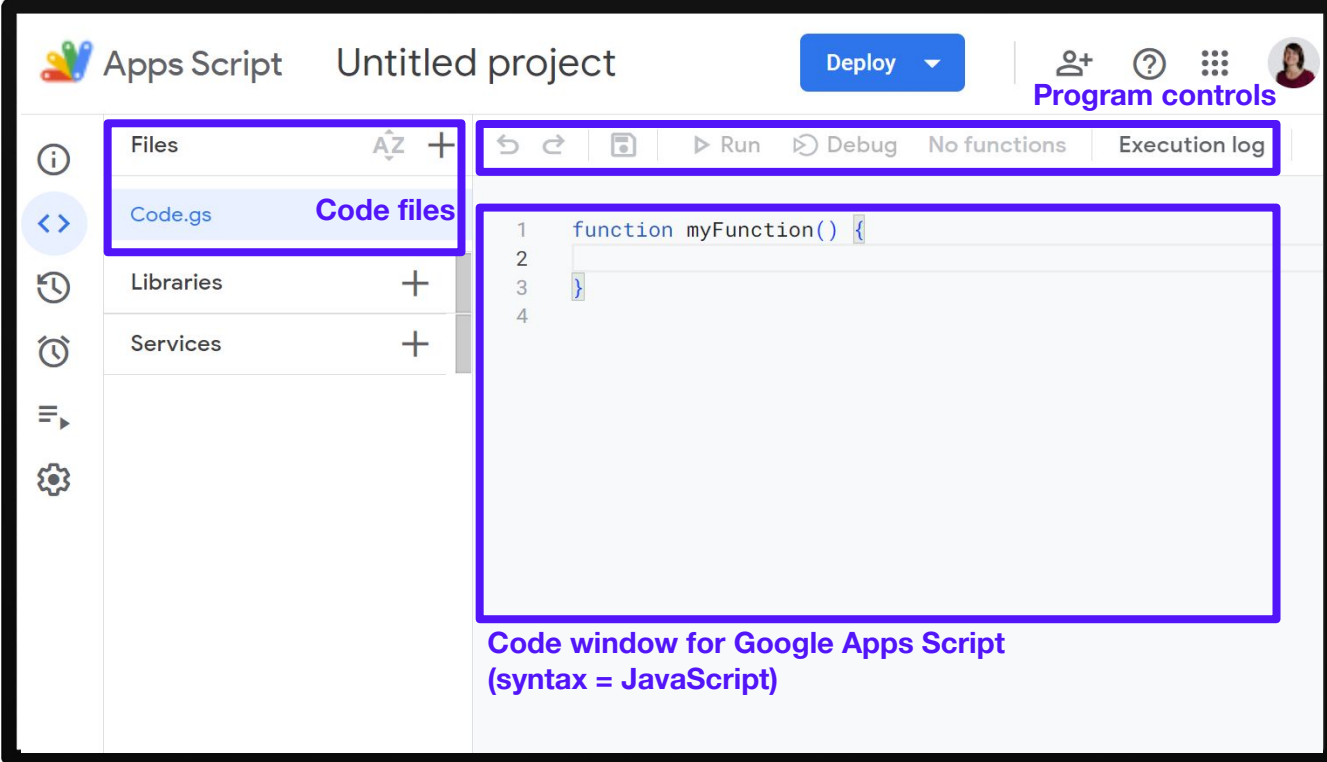


Both are 'just' code editors.



IDE: You can actually run your code & get a result.

Sections that you need to get started.



Apps Script Untitled project

Deploy

Program controls

Files **Code files**

Code.gs

Libraries +

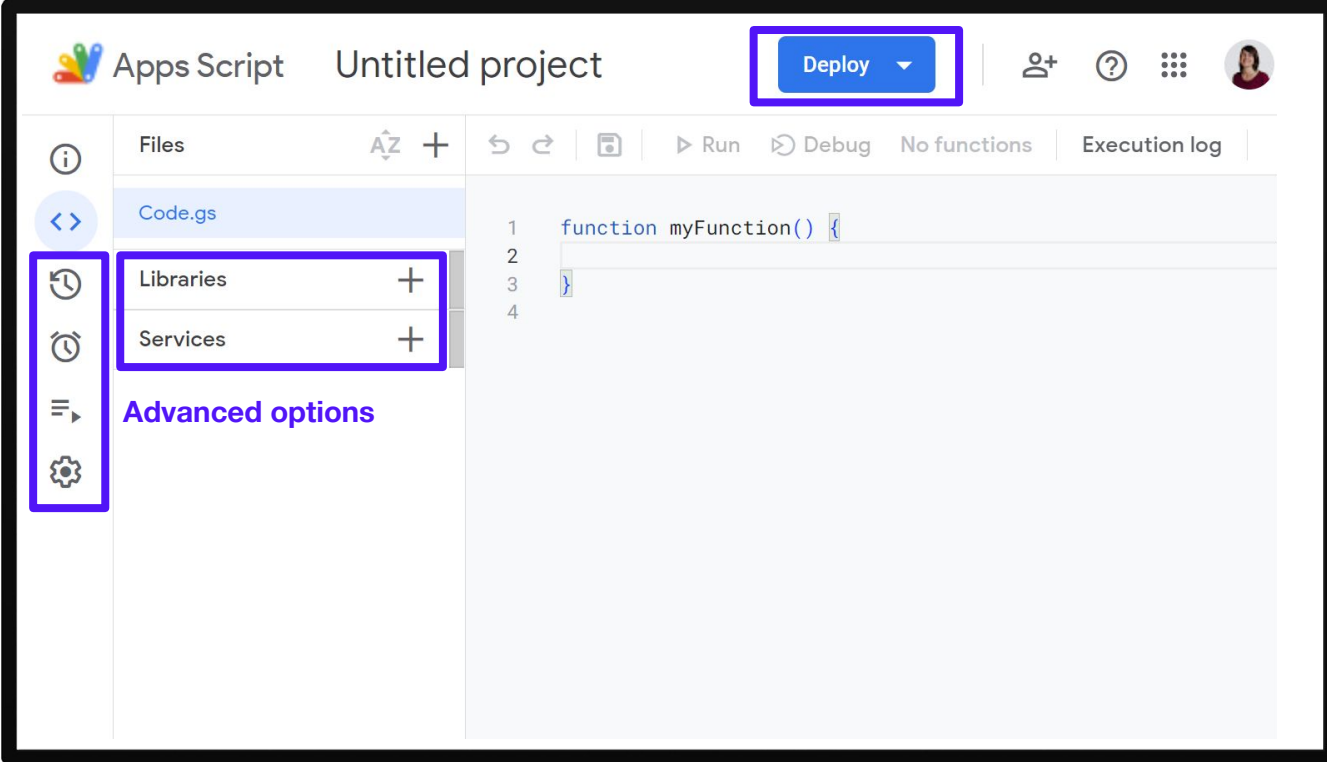
Services +

Run Debug No functions Execution log

```
1 function myFunction() {  
2  
3  
4
```

Code window for Google Apps Script
(syntax = JavaScript)

But there are also advanced options.

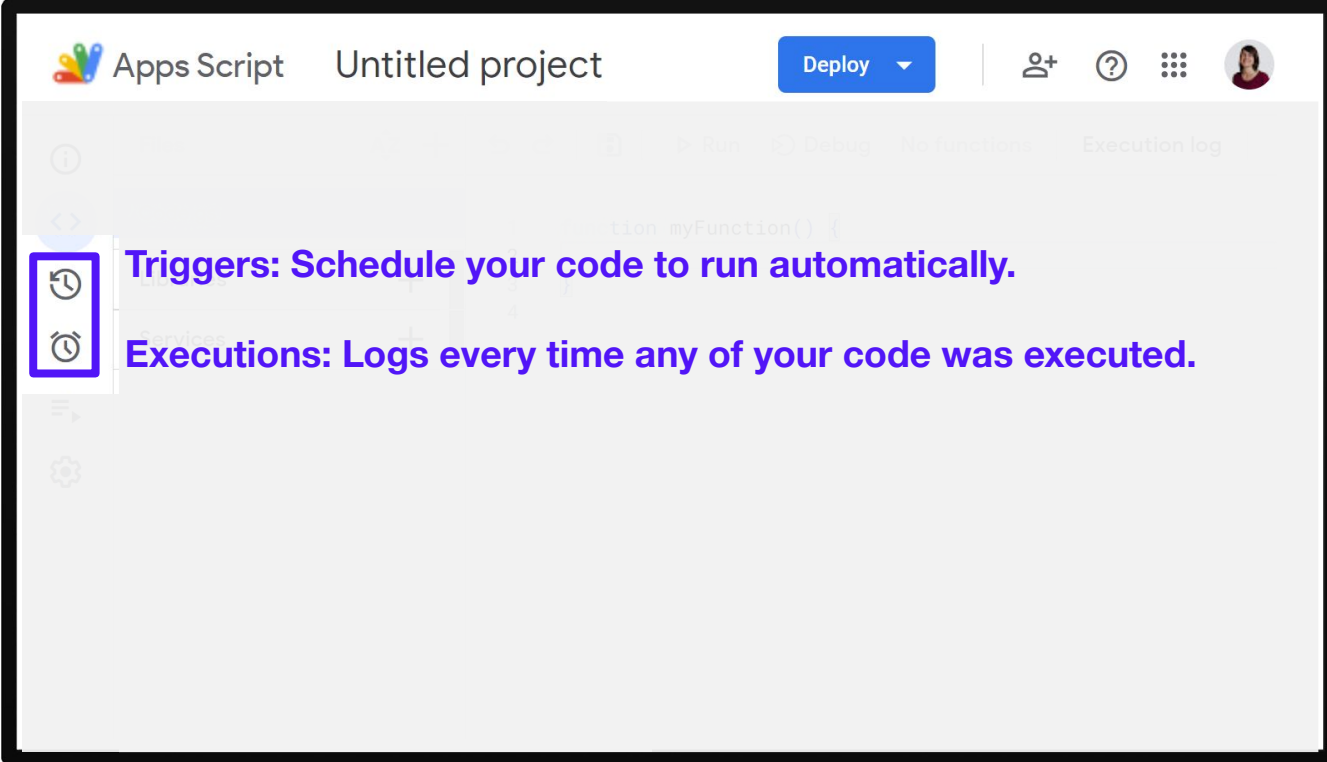


The screenshot shows the Google Apps Script editor interface. At the top, the text "Apps Script" and "Untitled project" are visible. A blue "Deploy" button with a dropdown arrow is highlighted with a purple box. Below the header, there are icons for help, user, and settings. The main editor area shows a code editor with a function definition:

```
1 function myFunction() {  
2  
3 }  
4
```

 On the left sidebar, the "Advanced options" section is highlighted with a purple box, showing "Libraries" and "Services" with plus signs, and a gear icon for settings.

But there are also advanced options.



The screenshot shows the Google Apps Script interface for an "Untitled project". The top navigation bar includes the "Apps Script" logo, the project name, a "Deploy" button, and user profile icons. Below the navigation bar, there are buttons for "Run", "Debug", "No functions", and "Execution log". The main area displays a code editor with a function definition: `function myFunction() {`. On the left side, a vertical menu contains several icons, with two highlighted by a blue box: a clock icon representing "Triggers" and an alarm clock icon representing "Executions".

Triggers: Schedule your code to run automatically.

Executions: Logs every time any of your code was executed.

Let's start with the
core element of each
script: a function.

Think of it like a formula
where you get to look inside
how the formula magic
works.

Like Google Sheets formulas,
Apps Script **functions** are
used to **contain logic** and to
make it reusable.

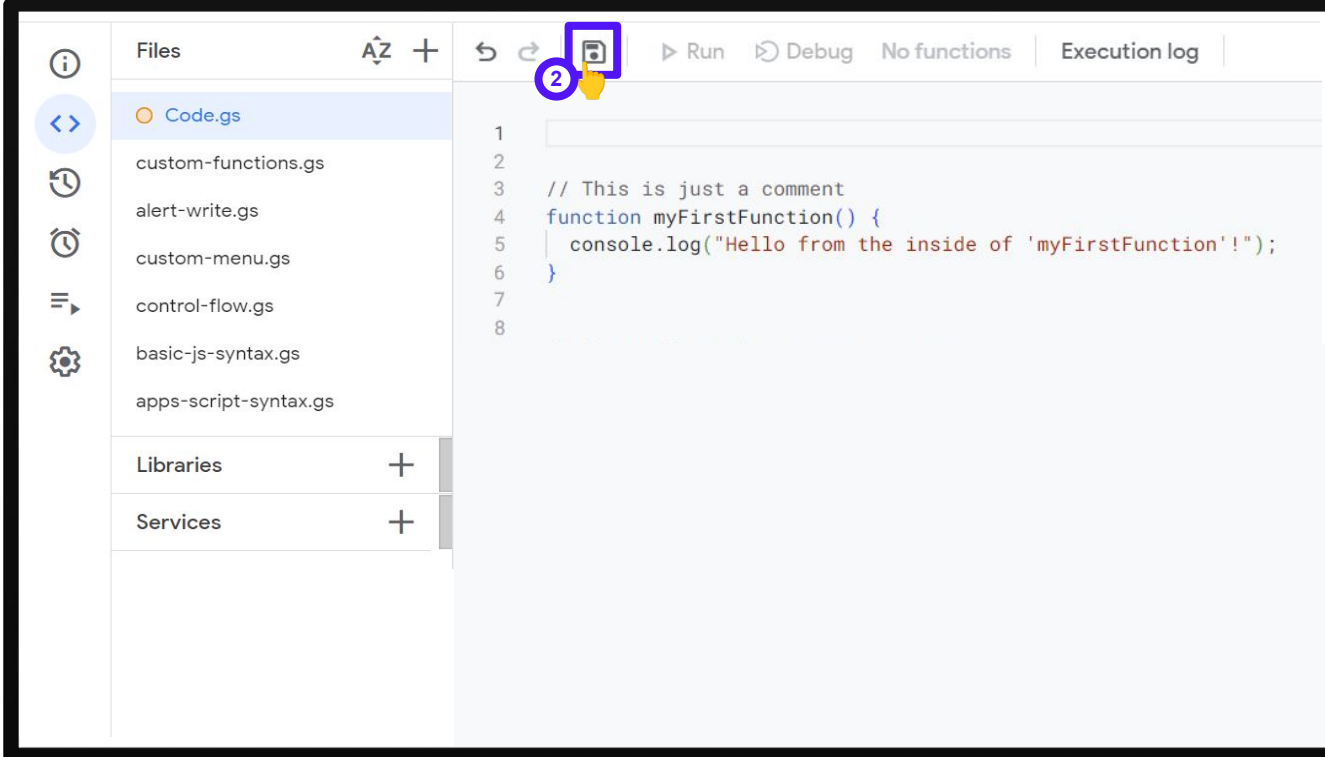
How to run your first function in Apps Script?

The screenshot shows the Google Apps Script editor interface. On the left, the 'Files' sidebar is visible, listing several files including 'Code.gs', 'custom-functions.gs', 'alert-write.gs', 'custom-menu.gs', 'control-flow.gs', 'basic-js-syntax.gs', and 'apps-script-syntax.gs'. The main editor area displays the following code:

```
1  
2  
3 // This is just a comment  
4 function myFirstFunction() {  
5   console.log("Hello from the inside of 'myFirstFunction!'");  
6 }  
7  
8
```

A blue rectangular box highlights the function definition on lines 4 and 5. A yellow arrow points to the right side of this box, and a blue circle with the number '1' is positioned next to the arrow.

How to run your first function in Apps Script?

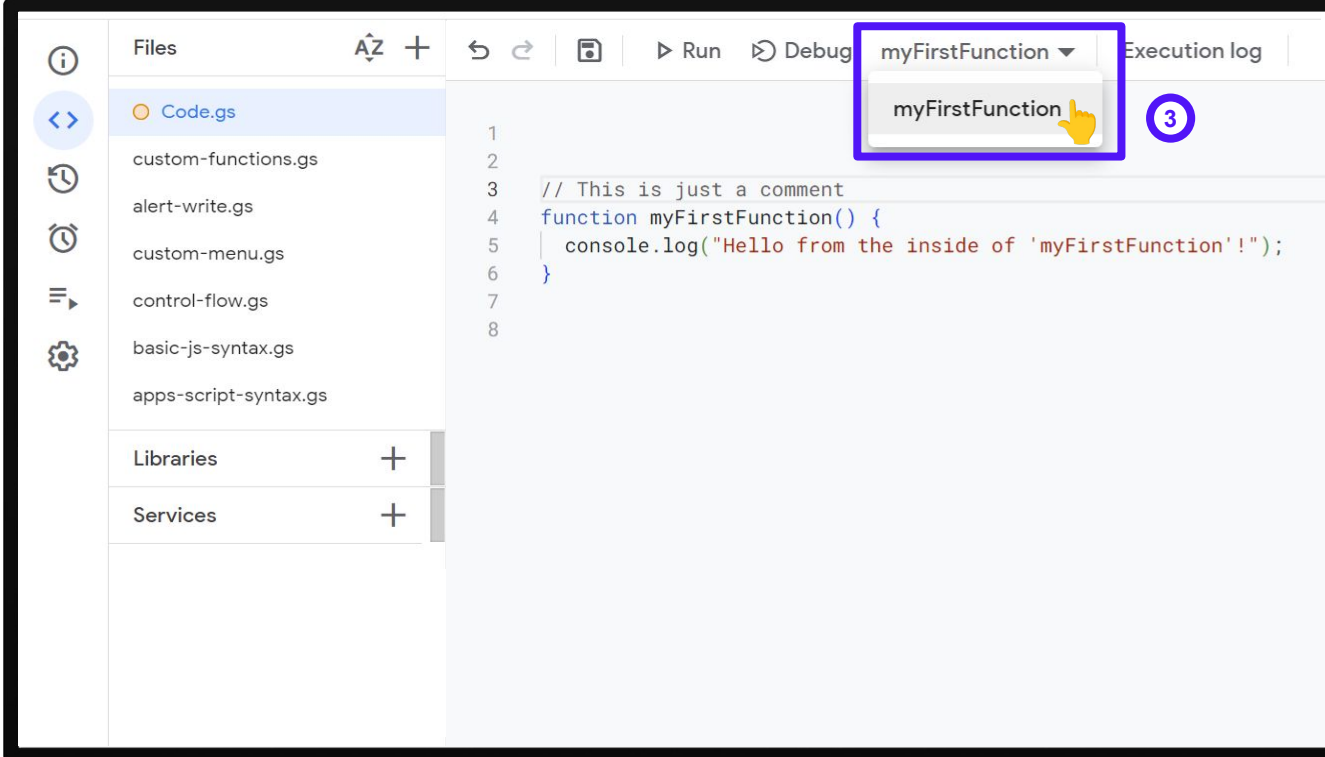


The screenshot shows the Google Apps Script editor interface. The left sidebar displays a file list with 'Code.gs' selected. The main editor area contains the following JavaScript code:

```
1  
2  
3 // This is just a comment  
4 function myFirstFunction() {  
5   console.log("Hello from the inside of 'myFirstFunction!'");  
6 }  
7  
8
```

A blue box highlights the 'Run' button in the top toolbar, with a yellow arrow pointing to it and a blue circle containing the number '2'.

How to run your first function in Apps Script?

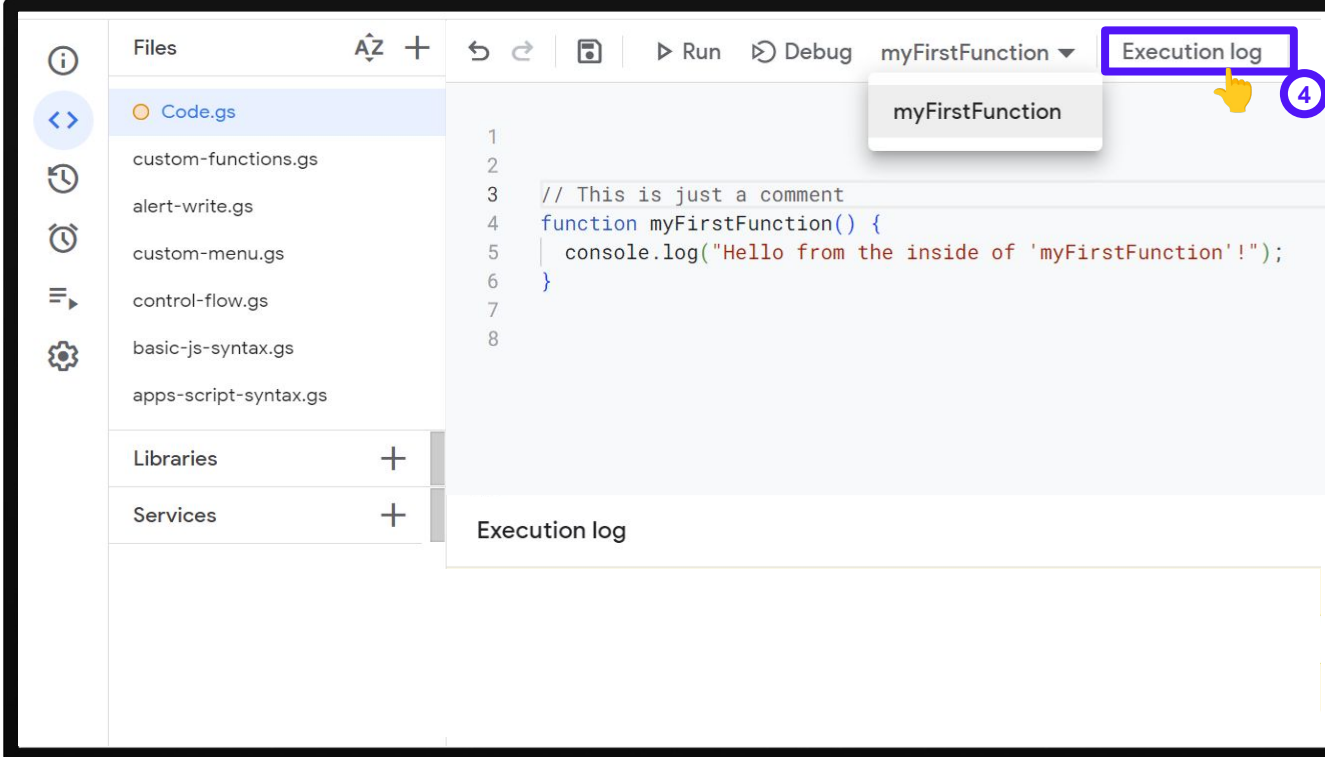


The screenshot displays the Google Apps Script editor interface. On the left, a file explorer shows a list of files including 'Code.gs', 'custom-functions.gs', 'alert-write.gs', 'custom-menu.gs', 'control-flow.gs', 'basic-js-syntax.gs', and 'apps-script-syntax.gs'. The main editor area shows the following code:

```
1  
2  
3 // This is just a comment  
4 function myFirstFunction() {  
5   console.log("Hello from the inside of 'myFirstFunction!'");  
6 }  
7  
8
```

At the top right, a dropdown menu is open, showing 'myFirstFunction' selected. A yellow hand cursor is pointing at the 'myFirstFunction' option in the dropdown. A blue circle with the number '3' is positioned to the right of the dropdown. The 'Run' button is visible in the top toolbar.

How to run your first function in Apps Script?



The screenshot displays the Google Apps Script editor interface. On the left, a sidebar contains a 'Files' panel with a list of files: 'Code.gs', 'custom-functions.gs', 'alert-write.gs', 'custom-menu.gs', 'control-flow.gs', 'basic-js-syntax.gs', and 'apps-script-syntax.gs'. Below the files are sections for 'Libraries' and 'Services', both with plus signs. The main editor area shows a code editor with the following code:

```
1  
2  
3 // This is just a comment  
4 function myFirstFunction() {  
5   console.log("Hello from the inside of 'myFirstFunction!'");  
6 }  
7  
8
```

At the top right of the editor, there is a dropdown menu currently set to 'myFirstFunction'. To its right, the text 'Execution log' is highlighted with a purple box. A yellow hand cursor is pointing at this 'Execution log' text. A blue circle with the number '4' is positioned next to the cursor. Below the code editor, there is an 'Execution log' section which is currently empty.

How to run your first function in Apps Script?

The screenshot shows the Google Apps Script editor interface. The top toolbar includes a 'Run' button, which is highlighted with a yellow hand cursor and a blue circle containing the number 5. The code editor displays the following JavaScript code:

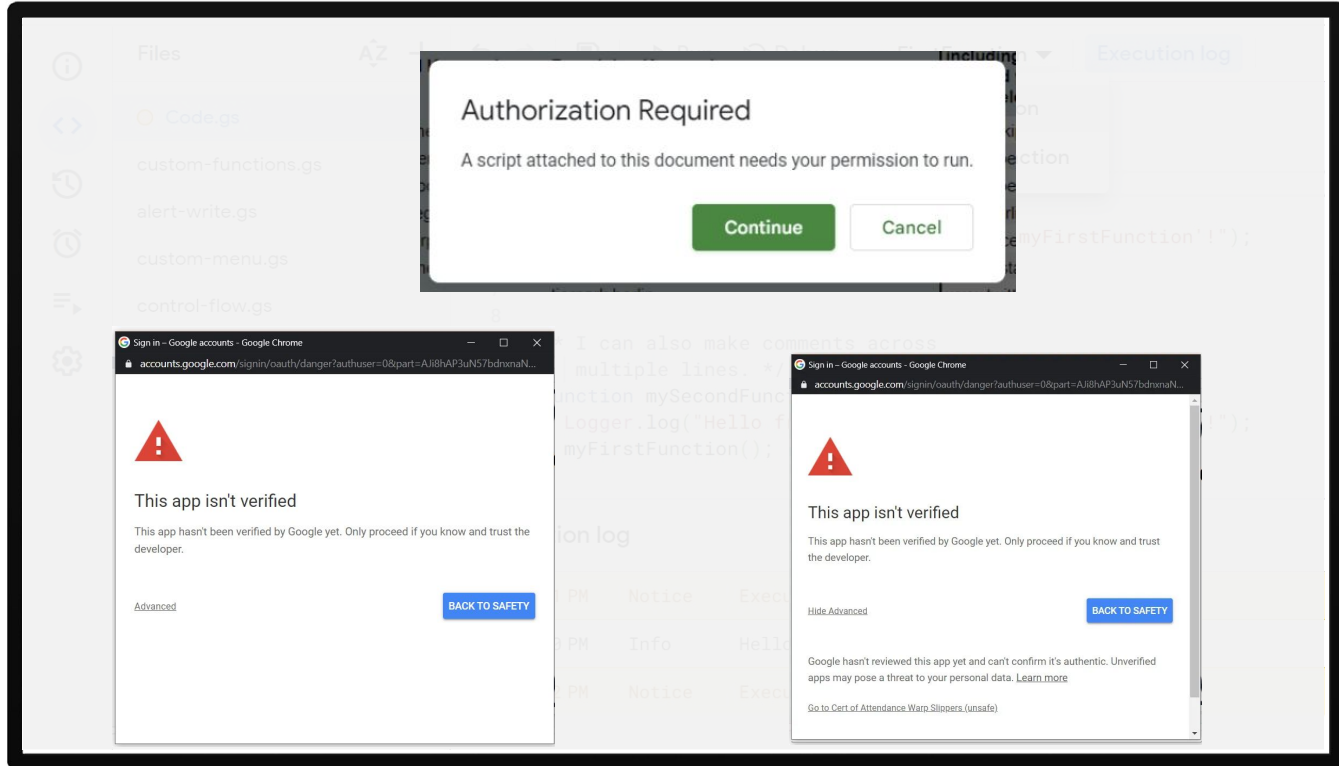
```
1  
2  
3 // This is just a comment  
4 function myFirstFunction() {  
5   console.log("Hello from the inside of 'myFirstFunction!'");  
6 }  
7  
8
```

The left sidebar shows a file explorer with the following files:

- Code.gs
- custom-functions.gs
- alert-write.gs
- custom-menu.gs
- control-flow.gs
- basic-js-syntax.gs
- apps-script-syntax.gs

Below the file explorer, there are sections for 'Libraries' and 'Services', both with a plus sign (+) next to them. The bottom of the editor shows an 'Execution log' section.

Sidenote: You will be asked for authorization.



[more info](#)



[more info](#)

How to run your first function in Apps Script?

The screenshot shows the Google Apps Script editor interface. The left sidebar contains a file explorer with 'Code.gs' selected. The main editor area shows the following code:

```
1  
2  
3 // This is just a comment  
4 function myFirstFunction() {  
5   console.log("Hello from the inside of 'myFirstFunction!'");  
6 }  
7
```

The 'Run' button in the top toolbar is highlighted with a blue box and a yellow hand cursor. A blue circle with the number '5' is next to it. A dropdown menu for 'myFirstFunction' is visible below the button. The line of code `console.log("Hello from the inside of 'myFirstFunction!'");` is also highlighted with a blue box.

Below the code editor is the 'Execution log' section, which contains the following entries:

| Time | Level | Message |
|------------|--------|---|
| 2:16:41 PM | Notice | Execution started |
| 2:16:40 PM | Info | Hello from the inside of 'myFirstFunction!' |
| 2:16:42 PM | Notice | Execution completed |

A blue box highlights the 'Info' entry in the log. A callout box on the right says "Thanks to → console.log() 🥰". A blue circle with the number '5' is next to the log entry.

How to run your first function in Apps Script?

The screenshot displays the Google Apps Script editor. The left sidebar shows a file list with 'Code.gs' selected. The main editor area contains the following code:

```
1
2
3 // This is just a comment
4 function myFirstFunction() {
5   console.log("Hello from the inside of 'myFirstFunction!'");
6 }
7
8
9 /* I can also make comments across
10  multiple lines. */
11 function mySecondFunction() {
12   Logger.log("Hello from the inside of 'mySecondFunction!'");
13   myFirstFunction();
14 }
```

The 'Run' button in the top toolbar is highlighted with a blue box and a yellow arrow, with a circled '5' next to it. A dropdown menu is open, showing 'mySecondFunction' selected, also highlighted with a blue box. The 'mySecondFunction' code block is highlighted with a blue box. The 'Execution log' at the bottom shows the following entries:

| Time | Level | Message |
|------------|--------|--|
| 2:22:36 PM | Notice | Execution started |
| 2:22:36 PM | Info | Hello from the inside of 'mySecondFunction'! |
| 2:22:36 PM | Info | Hello from the inside of 'myFirstFunction'! |
| 2:22:37 PM | Notice | Execution completed |

A circled '5' is next to the first log entry. A callout box points to the log entries with the text: "Logger.log() → does the same".

CONCEPT

JavaScript Functions

'A function is a block of code that performs a specific task.'

JavaScript functions are basically used to encapsulate logic, making that code more reusable and easier to understand.

Functions can take input in the form of parameters and can return a value or output.'

Source: [freeCodeCamp](#)

The screenshot shows a code editor with the following JavaScript code:

```

1
2
3 // This is just a comment
4 function myFirstFunction() {
5   console.log("Hello from the inside of 'myFirstFunction!'");
6 }
7
8
9 /* I can also make comments across
10  | multiple lines. */
11 function mySecondFunction() {
12   logger.log("Hello from the inside of 'mySecondFunction!'");
13   myFirstFunction();
14 }
15

```

The execution log below the code shows the following entries:

| Time | Level | Message |
|------------|--------|--|
| 2:22:36 PM | Notice | Execution started |
| 2:22:36 PM | Info | Hello from the inside of 'mySecondFunction'! |
| 2:22:36 PM | Info | Hello from the inside of 'myFirstFunction'! |
| 2:22:37 PM | Notice | Execution completed |

We reused *myFirstFunction* inside *mySecondFunction* 

We can do
more than just logging
with our function outputs.

CONCEPT

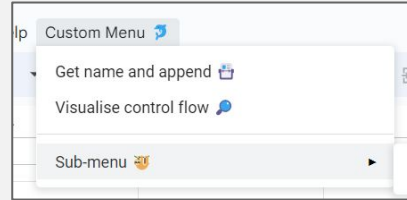
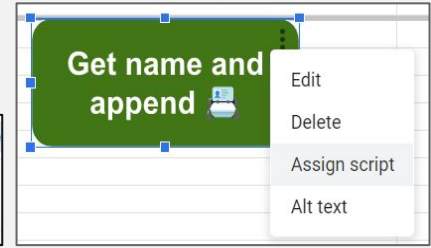
Run & Display Options

There are different ways to run the logic in a function and to display its output.

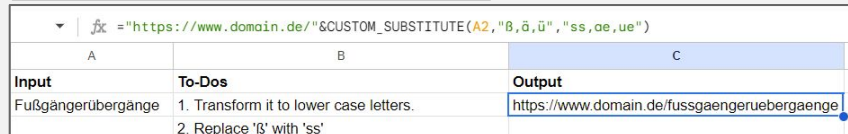
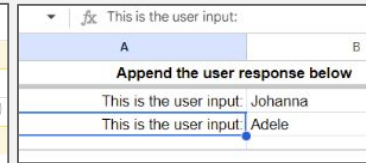
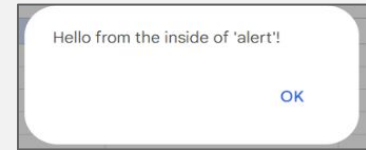
Run it: via (1) the editor, (2) a button, (3) a custom menu or (4) triggers.

Display it: with (1) the execution log, (2) UI alerts, (3) by writing to a sheet or (4) returning a value in a cell.

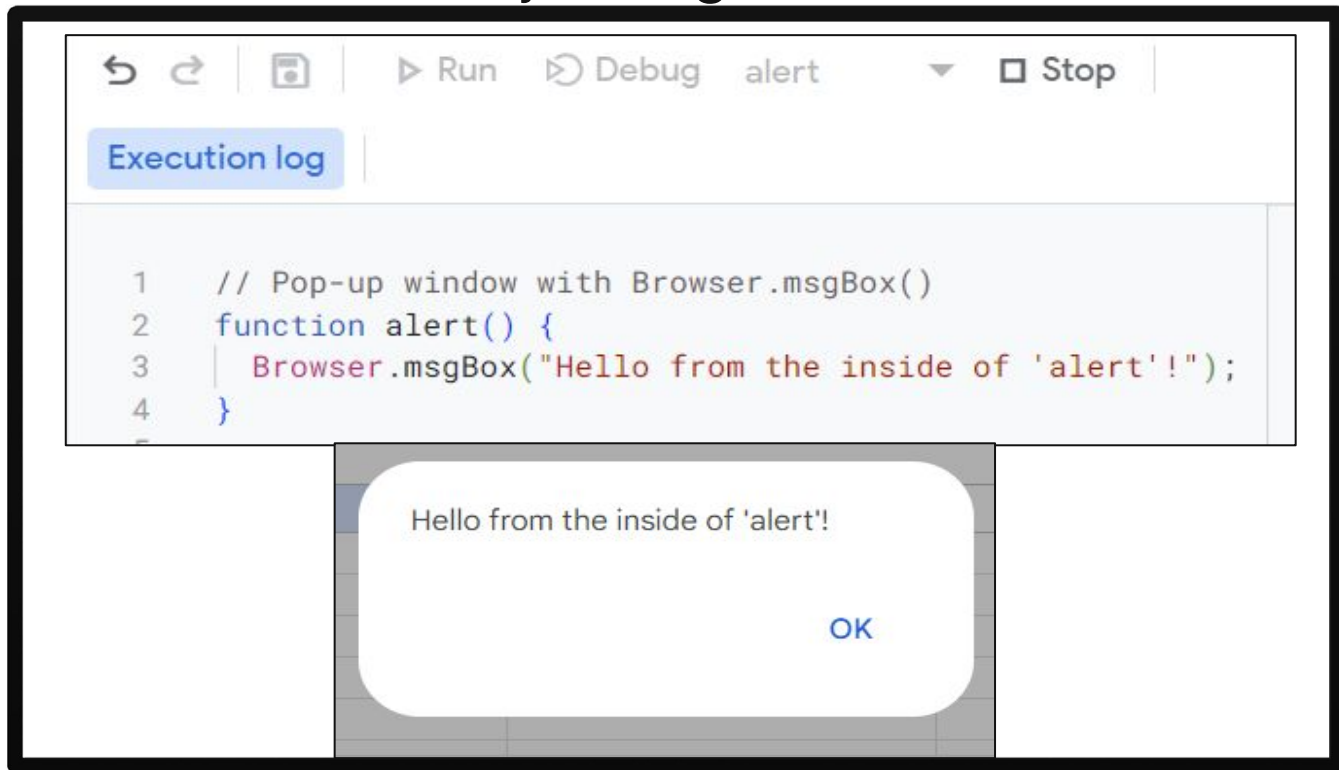
RUN IT



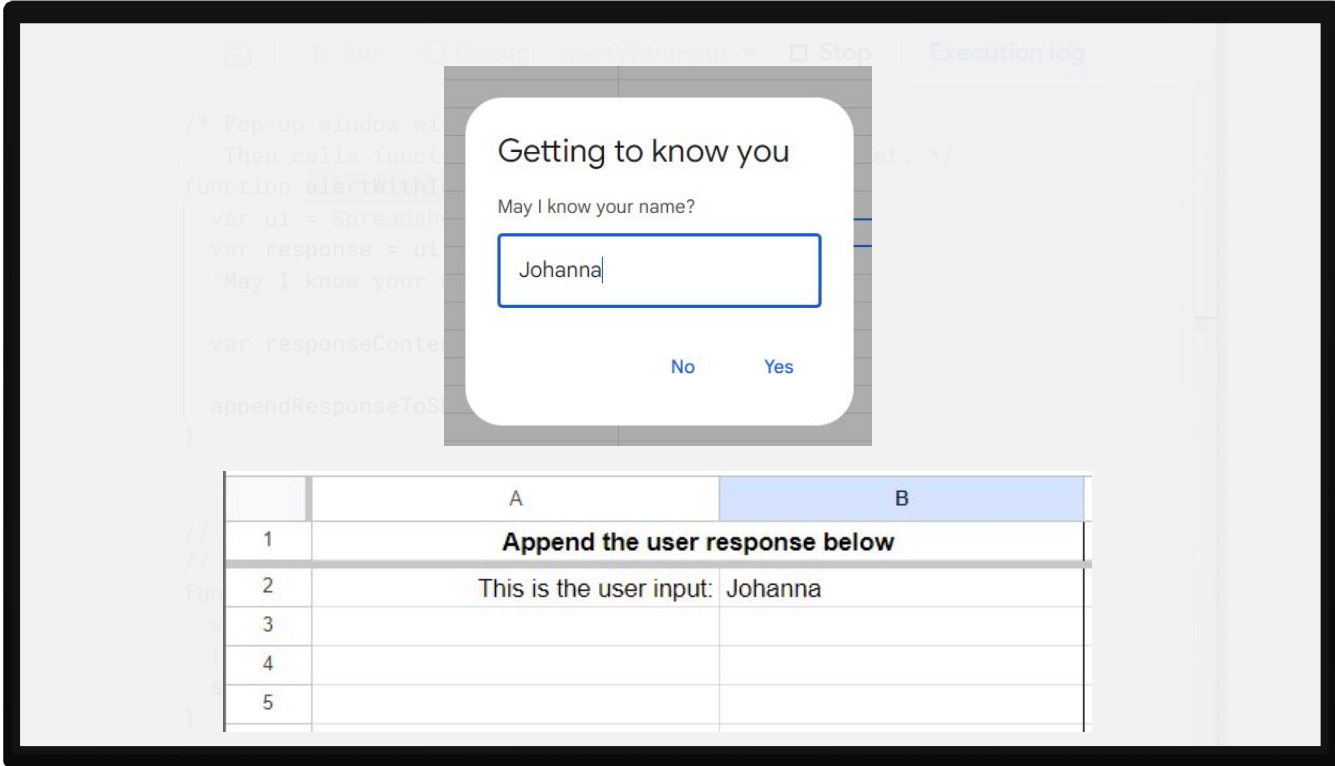
DISPLAY IT



Don't just log - ALERT!



Don't just alert - WRITE!



The screenshot shows a web browser interface with a JavaScript alert dialog box. The dialog box has a title "Getting to know you" and a question "May I know your name?". Below the question is a text input field containing the name "Johanna". At the bottom of the dialog are two buttons: "No" and "Yes".

In the background, a spreadsheet is visible with the following content:

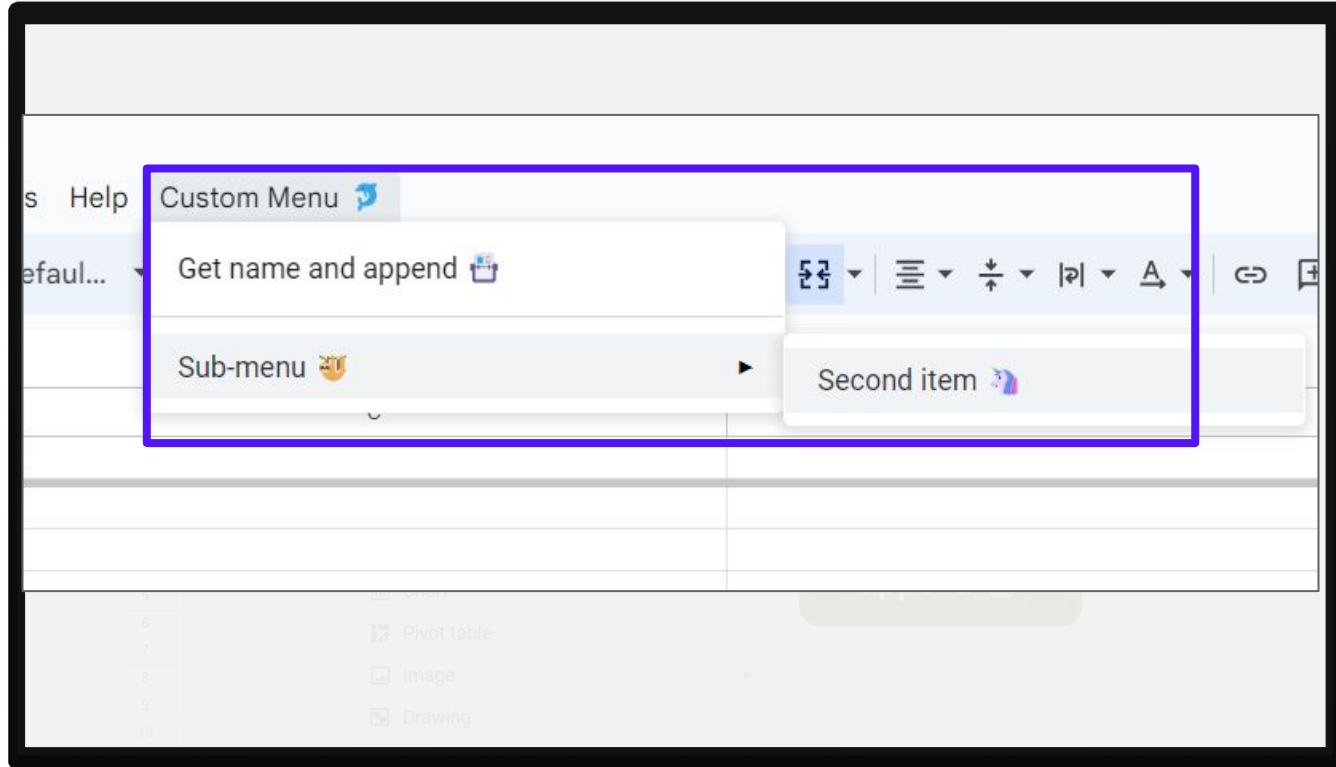
| | A | B |
|---|---------------------------------------|---------|
| 1 | Append the user response below | |
| 2 | This is the user input: | Johanna |
| 3 | | |
| 4 | | |
| 5 | | |

Create **MENUS** to start it all!

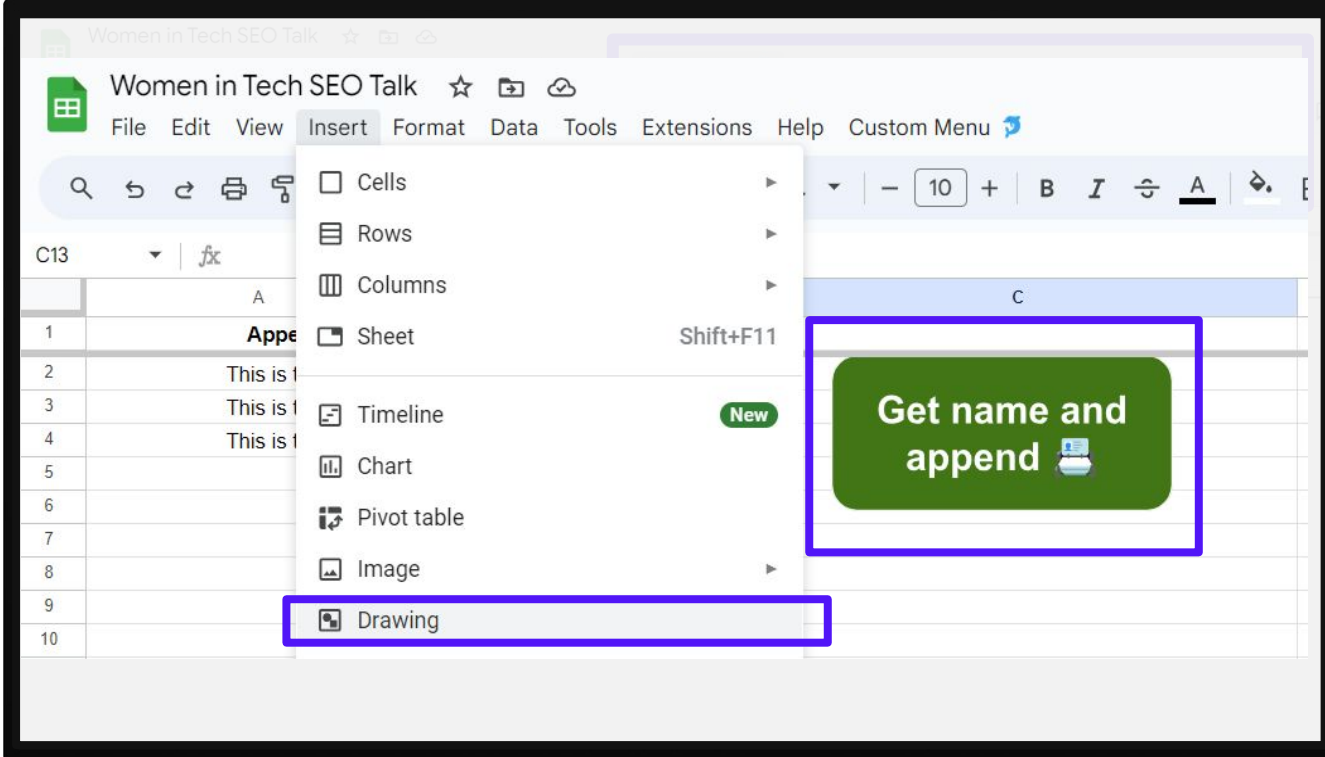
```
1 // Documentation: https://developers.google.com/apps-script/guides/menus
2
3 function onOpen() {
4   var ui = SpreadsheetApp.getUi();
5
6   ui.createMenu('Custom Menu 🗨️')
7     .addItem('Get name and append 📄', 'alertWithInput')
8     .addSeparator()
9     .addSubMenu(ui.createMenu('Sub-menu 🍷')
10      | | .addItem('Second item 🗨️', 'menuItem2'))
11     .addToUi();
12 }
13
14
15 // Function to alert you second menu item is clicked.
16 function menuItem2() {
17   // SpreadsheetApp or DocumentApp, SlidesApp or FormApp.
18   SpreadsheetApp.getUi()
19   | | .alert('You clicked the second menu item!');
20 }
```



Run everything from a MENU in the Sheets UI!



Besides your own menus, you can also use **BUTTONS**.



The screenshot shows a Google Sheets interface with a custom menu open. The menu items are: Cells, Rows, Columns, Sheet, Timeline, Chart, Pivot table, Image, and Drawing. A green button with the text "Get name and append" and a printer icon is highlighted with a blue border. The button is located in the center of the screen, overlapping the menu. The background shows a spreadsheet with columns A, B, and C, and rows 1 through 10. The title bar of the spreadsheet is "Women in Tech SEO Talk".



custom-menu.gs



alert-write

Schedule your functions with TRIGGERS!

The screenshot displays the Google App Script interface for a project named "Women in Tech SEO Festival". The interface is divided into two panels. The top panel shows the "Triggers" tab selected in the left-hand navigation menu. The main content area displays a table with one trigger listed. The table has columns for Deployment, Event, Function, and Error rate. The trigger listed is "Head" (Deployment), "Time-based" (Event), "myFirstFunction" (Function), and "-" (Error rate). The bottom panel shows the "Triggers" tab selected in the left-hand navigation menu. The main content area displays a table with one trigger listed. The table has columns for Owned by, Last run, Deployment, Event, Function, and Error rate. The trigger listed is "Me" (Owned by), "-" (Last run), "Head" (Deployment), "Time-based" (Event), "myFirstFunction" (Function), and "-" (Error rate). There is also a "+ Add a filter" button above the table.

Apps Script Women in Tech SEO Festival Deploy ? [User Profile]

Overview Showing 1 trigger

Editor

Project History

Triggers

Executions

| Deployment | Event | Function | Error rate |
|------------|------------|-----------------|------------|
| Head | Time-based | myFirstFunction | - |

Apps Script Women in Tech SEO Festival Deploy ? [User Profile]

Triggers Showing 1 trigger

+ Add a filter

| Owned by | Last run | Deployment | Event | Function | Error rate |
|----------|----------|------------|------------|-----------------|------------|
| Me | - | Head | Time-based | myFirstFunction | - |

Schedule your functions with TRIGGERS!

The screenshot shows the 'Add Trigger for Women in Tech SEO Festival' dialog box in the Google Cloud console. The dialog is divided into two main sections: configuration and failure notification settings.

Configuration Section:

- Choose which function to run:** myFirstFunction
- Choose which deployment should run:** Head
- Select event source:** Time-driven
- Select type of time based trigger:** Hour timer
- Select hour interval:** Every hour

Failure notification settings: Notify me daily

At the bottom of the dialog, there are 'Cancel' and 'Save' buttons.

The background shows the Google Cloud Functions interface with a list of triggers. The first trigger is visible, showing 'Showing 1 trigger' and 'Error rate'.

JavaScript functions are like
Google Sheets formulas.

Make your own formulas: custom functions

```

=TRIM(
SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(
SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(
SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(
SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(
JOIN(" ",ARRAYFORMULA(IF(RIGHT(SORT(TRANPOSE(SPLIT(lower(trim(A1))," ")),1)="s",
LEFT(SORT(TRANPOSE(SPLIT(LOWER(TRIM(A1))," ")),LEN(SORT(TRANPOSE(SPLIT(LOWER(TRIM(A1))," "))-1),
SORT(TRANPOSE(SPLIT(A1," "))))))
," las",""),"las ","")," los",""),"los ",""),"á","a"),"é","e"),"ó","o"),"ú","u"),"í","i"),"ñ","n")," &","
"),"and ",""),"and ","")," de",""),"de ","")," la",""),"la ","")," el",""),"el ","")," para",""),
"para ","")," ','"),"à","a"),"è","e"),"ò","o"),"ü","u"),"ì","i")," ", " ")
)
  
```



Anonymous

Write a Google Apps Script custom function that works like a nested SUBSTITUTE() formula.

It should take 3 parameters: cell with text, characters to replace, characters to replace them with. It should be possible to specify multiple, comma-separated characters to replace.

Before any substitution steps, it should also transform the characters to lowercase. It should return the transformed value. Include JSDoc annotations.

Example Input: fußgängerübergänge

Example Output: fussgaengeruebergaenge



ChatGPT

Make your own formulas with custom functions.

```
26 // Custom function to substitute multiple input values
27 // https://chat.openai.com/share/aa82ad85-bbae-47e3-91f1-b25d2c246ca7
28
29 /**
30  * Replaces specified characters in a given text with the provided replacement characters.
31  * Before substitution, transforms the characters to lowercase.
32  *
33  * @param {string} text - The cell value or text to be transformed.
34  * @param {string} charactersToReplace - The characters to be replaced (comma-separated).
35  * @param {string} replacementCharacters - The characters to replace with (comma-separated).
36  * @return {string} The transformed text after substitution.
37  * @customfunction
38  */
39 function CUSTOM_SUBSTITUTE(text, charactersToReplace, replacementCharacters) {
40     // Convert characters to arrays
41     var fromArray = charactersToReplace.split(",");
42     var toArray = replacementCharacters.split(",");
43
44     // Ensure equal length of fromArray and toArray
45     if (fromArray.length !== toArray.length) {
46         throw new Error("Number of characters to replace and replacement characters must be equal.");
47     }
48
49     // Convert text to lowercase
50     text = text.toLowerCase();
51
52     // Perform substitution
53     for (var i = 0; i < fromArray.length; i++) {
54         text = text.replace(new RegExp(fromArray[i], "g"), toArray[i]);
55     }
56
57     return text;
58 }
```

No logging, no alerting, no appending – just **return**



Make your own formulas: custom functions

The screenshot shows a Google Sheets spreadsheet with a custom function being used. The formula bar displays the following formula:

```
=https://www.domain.de/&CUSTOM_SUBSTITUTE(A2, "ß,ä,ü", "ss,ae,ue")
```

The spreadsheet is organized into three columns: A (Input), B (To-Dos), and C (Output). The data is as follows:

| Input | To-Dos | Output |
|--------------------|---|--|
| Fußgängerübergänge | <ol style="list-style-type: none"> 1. Transform it to lower case letters. 2. Replace 'ß' with 'ss' 3. Replace 'ä' with 'ae' 4. Replace 'ü' with 'ue' 5. Add result to "https://www.domain.de/" | https://www.domain.de/fussgaengeruebergaenge |

A tooltip for the `CUSTOM_SUBSTITUTE` function is shown, providing the following information:

```
CUSTOM_SUBSTITUTE(text,
charactersToReplace, replacementCharacters
)
CUSTOM_SUBSTITUTE(string, string, string)
ABOUT
Replaces specified characters in a given text with the
provided replacement characters. Before substitution,
transforms the characters to lowercase.
text
- The cell value or text to be transformed.
charactersToReplace
- The characters to be replaced (comma-separated).
replacementCharacters
- The characters to replace with (comma-separated).
```



JavaScript & Google Sheets
syntax has similar logic.

03

Control Flow, Debugging & Prompting

ChatGPT will speed up your coding.

But to fact-check it, it helps to think a bit like a machine.

How does your computer
try to run the code that
ChatGPT gives you?

CONCEPT

Control Flow in JavaScript

'Control flow in JavaScript is how your computer runs code from top to bottom.'

It starts from the first line and ends at the last line, unless it hits any statement that changes the control flow of the program such as loops, conditionals, or functions.'

Source: medium.com

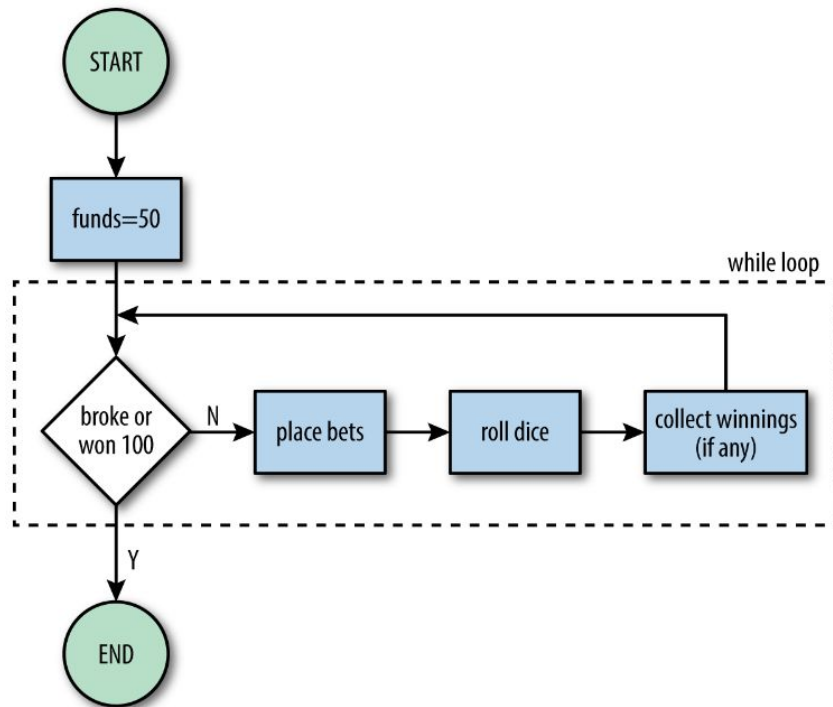


Figure 4-2. Crown and Anchor simulation flowchart (refined)

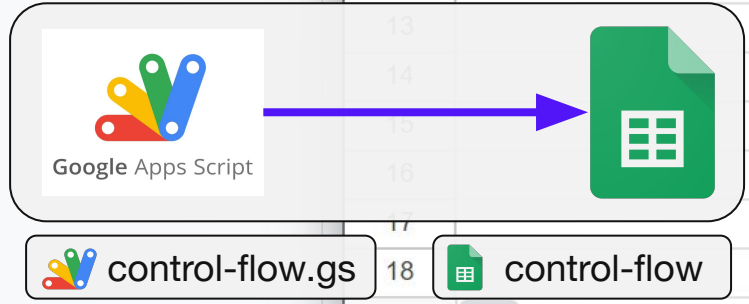
Source: [Learning JavaScript, 3rd Edition](#)

```

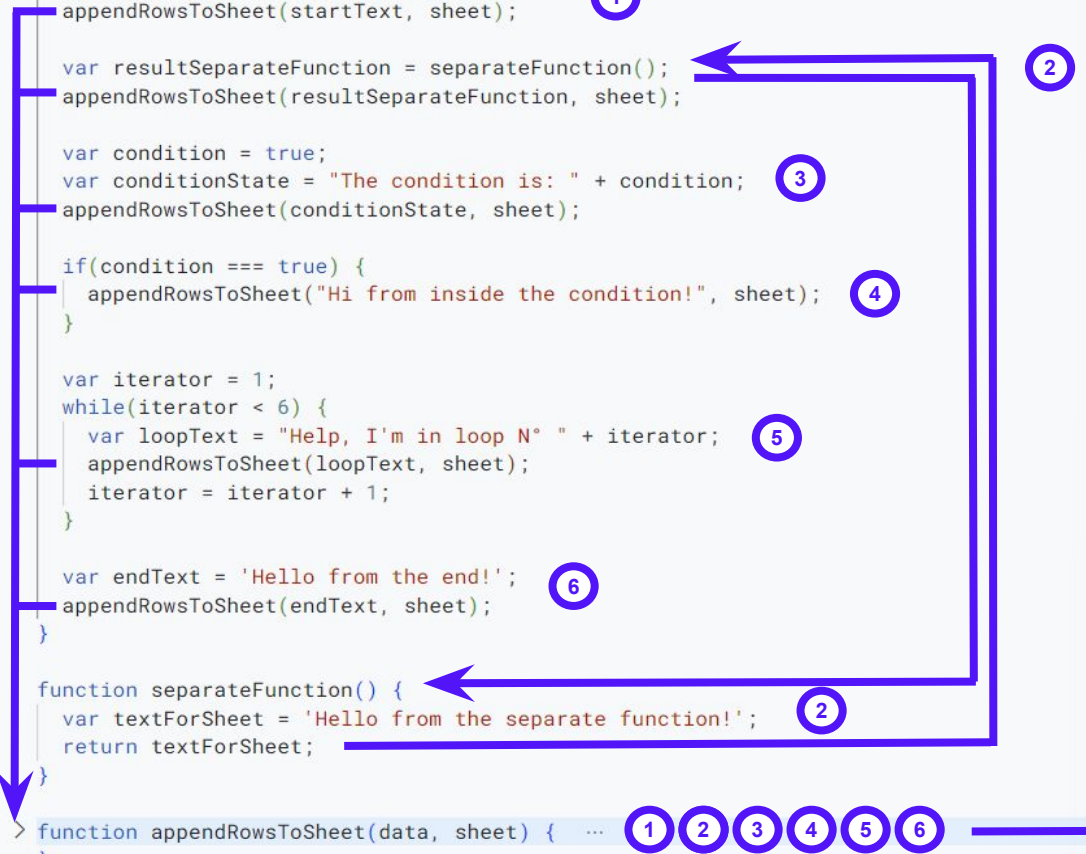
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: " + condition;
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) { ...
44 \

```

| | A |
|----|-----------------------------------|
| 1 | Hello from the start! |
| 2 | Hello from the separate function! |
| 3 | The condition is: true |
| 4 | Hi from inside the condition! |
| 5 | Help, I'm in loop N° 1 |
| 6 | Help, I'm in loop N° 2 |
| 7 | Help, I'm in loop N° 3 |
| 8 | Help, I'm in loop N° 4 |
| 9 | Help, I'm in loop N° 5 |
| 10 | Hello from the end! |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |
| 16 | |
| 17 | |
| 18 | |

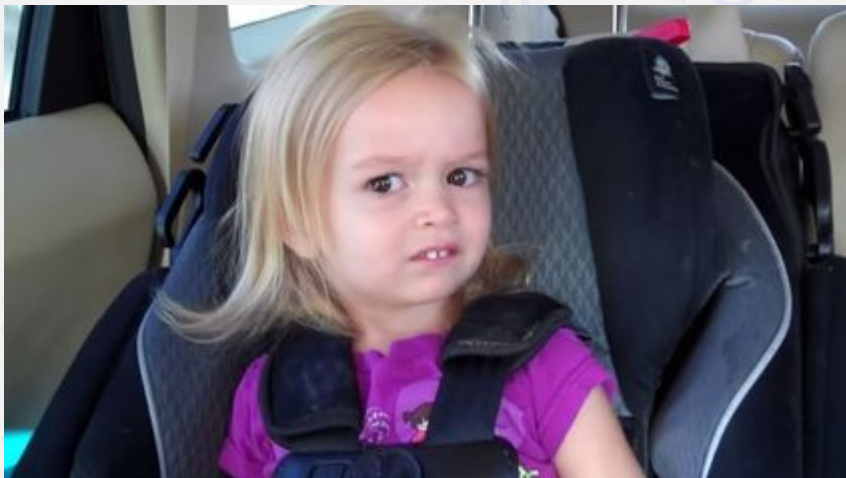


```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: " + condition;
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) { ...
44 }
```



| | A |
|----|-----------------------------------|
| 1 | Hello from the start! |
| 2 | Hello from the separate function! |
| 3 | The condition is: true |
| 4 | Hi from inside the condition! |
| 5 | Help, I'm in loop N° 1 |
| 5 | Help, I'm in loop N° 2 |
| 5 | Help, I'm in loop N° 3 |
| 5 | Help, I'm in loop N° 4 |
| 5 | Help, I'm in loop N° 5 |
| 6 | Hello from the end! |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |
| 16 | |
| 17 | |
| 18 | |
| 19 | |
| 20 | |
| 21 | |
| 22 | |
| 23 | |
| 24 | |
| 25 | |
| 26 | |
| 27 | |
| 28 | |
| 29 | |
| 30 | |
| 31 | |
| 32 | |
| 33 | |
| 34 | |
| 35 | |
| 36 | |
| 37 | |
| 38 | |
| 39 | |
| 40 | |
| 41 | |
| 42 | |
| 43 | |
| 44 | |

```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: true";
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) {
```



| | |
|----|-----------------------------------|
| 1 | Hello from the start! |
| 2 | Hello from the separate function! |
| 3 | The condition is: true |
| 4 | Hi from inside the condition! |
| 5 | Help, I'm in loop N° 1 |
| 6 | Help, I'm in loop N° 2 |
| 7 | Help, I'm in loop N° 3 |
| 8 | Help, I'm in loop N° 4 |
| 9 | Help, I'm in loop N° 5 |
| 10 | Hello from the end! |

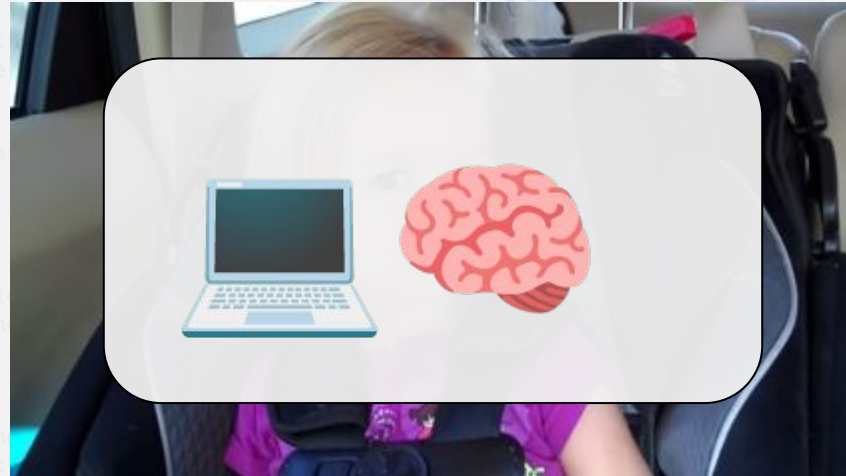
```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: true";
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) {
```

1

2

2

1 2 3 4 5 6



- 1
- 2
- 3
- 4
- 5

Hello from the start!
Hello from the separate function!
The condition is: true
Hi from inside the condition!
Help, I'm in loop N° 1
p, I'm in loop N° 2
p, I'm in loop N° 3
p, I'm in loop N° 4
p, I'm in loop N° 5
Hello from the end!

```

1  var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3  function visualiseControlFlow() {
4    var startText = 'Hello from the start!';
5    appendRowsToSheet(startText, sheet);
6
7    var resultSeparateFunction = separateFunction();
8    appendRowsToSheet(resultSeparateFunction, sheet);
9
10   var condition = true;
11   var conditionState = "The condition is: " + condition;
12   appendRowsToSheet(conditionState, sheet);
13
14   if(condition === true) {
15     appendRowsToSheet("Hi from inside the condition!", sheet);
16   }
17
18   var iterator = 1;
19   while(iterator < 6) {
20     var loopText = "Help, I'm in loop N° " + iterator;
21     appendRowsToSheet(loopText, sheet);
22     iterator = iterator + 1;
23   }
24
25   var endText = 'Hello from the end!';
26   appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) { ...
44 \

```



sheet: data object of tab 'control-flow'


```

1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4     var startText = 'Hello from the start!';
5     appendRowsToSheet(startText, sheet);
6
7     var resultSeparateFunction = separateFunction();
8     appendRowsToSheet(resultSeparateFunction, sheet);
9
10    var condition = true;
11    var conditionState = "The condition is: " + condition;
12    appendRowsToSheet(conditionState, sheet);
13
14    if(condition === true) {
15        appendRowsToSheet("Hi from inside the condition!", sheet);
16    }
17
18    var iterator = 1;
19    while(iterator < 6) {
20        var loopText = "Help, I'm in loop N° " + iterator;
21        appendRowsToSheet(loopText, sheet);
22        iterator = iterator + 1;
23    }
24
25    var endText = 'Hello from the end!';
26    appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30     var textForSheet = 'Hello from the separate function!';
31     return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) { ...
44 \

```



sheet: data object of tab 'control-flow'
 startText: 'Hello from the start!'

```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: " + condition;
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 function appendRowsToSheet(data, sheet) {
```

| | A |
|----|-----------------------|
| 1 | Hello from the start! |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |
| 16 | |
| 17 | |
| 18 | |

1

1

1



```
var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");

function visualiseControlFlow() {
  var startText = 'Hello from the start!';
  appendRowsToSheet(startText, sheet);

  var resultSeparateFunction = separateFunction();
  appendRowsToSheet(resultSeparateFunction, sheet);

  var condition = true;
  var conditionState = "The condition is: " + condition;
  appendRowsToSheet(conditionState, sheet);

  if(condition === true) {
    appendRowsToSheet("Hi from inside the condition!", sheet);
  }

  var iterator = 1;
  while(iterator < 6) {
    var loopText = "Help, I'm in loop N° " + iterator;
    appendRowsToSheet(loopText, sheet);
    iterator = iterator + 1;
  }

  var endText = 'Hello from the end!';
  appendRowsToSheet(endText, sheet);
}

function separateFunction() {
  var textForSheet = 'Hello from the separate function!';
  return textForSheet;
}

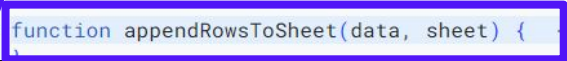
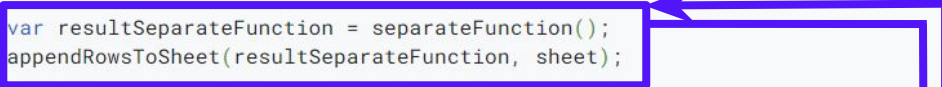
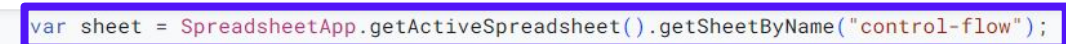
function appendRowsToSheet(data, sheet) { ...
}
```



sheet: data object of tab 'control-flow'
startText: 'Hello from the start!'
resultSeparateFunction: 'Hello from the separate function!'

```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: " + condition;
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 function appendRowsToSheet(data, sheet) {
35 }
```

| | A |
|----|-----------------------------------|
| 1 | Hello from the start! |
| 2 | Hello from the separate function! |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |
| 16 | |
| 17 | |
| 18 | |

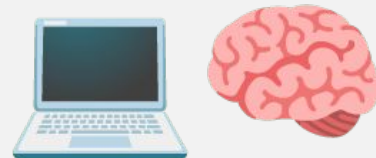


```

1  var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3  function visualiseControlFlow() {
4    var startText = 'Hello from the start!';
5    appendRowsToSheet(startText, sheet);
6
7    var resultSeparateFunction = separateFunction();
8    appendRowsToSheet(resultSeparateFunction, sheet);
9
10   var condition = true;
11   var conditionState = "The condition is: " + condition;
12   appendRowsToSheet(conditionState, sheet);
13
14   if(condition === true) {
15     appendRowsToSheet("Hi from inside the condition!", sheet);
16   }
17
18   var iterator = 1;
19   while(iterator < 6) {
20     var loopText = "Help, I'm in loop N° " + iterator;
21     appendRowsToSheet(loopText, sheet);
22     iterator = iterator + 1;
23   }
24
25   var endText = 'Hello from the end!';
26   appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) { ...
44 \

```

3



sheet: data object of tab 'control-flow'
startText: 'Hello from the start!'
resultSeparateFunction: 'Hello from the separate function!'
condition: true
conditionState: 'The condition is: true'

```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: " + condition;
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 function appendRowsToSheet(data, sheet) { ...
35 }
```

B20 | fx | A

| | |
|----|-----------------------------------|
| 1 | Hello from the start! |
| 2 | Hello from the separate function! |
| 3 | The condition is: true |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |
| 16 | |
| 17 | |
| 18 | |
| 19 | |
| 20 | |
| 21 | |
| 22 | |
| 23 | |
| 24 | |
| 25 | |
| 26 | |
| 27 | |
| 28 | |
| 29 | |
| 30 | |
| 31 | |
| 32 | |
| 33 | |
| 34 | |
| 35 | |

+ | ≡ | control-flow | Notes | < | >

3

3

3

```

1  var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3  function visualiseControlFlow() {
4    var startText = 'Hello from the start!';
5    appendRowsToSheet(startText, sheet);
6
7    var resultSeparateFunction = separateFunction();
8    appendRowsToSheet(resultSeparateFunction, sheet);
9
10   var condition = true;
11   var conditionState = "The condition is: " + condition;
12   appendRowsToSheet(conditionState, sheet);
13
14   if(condition === true) {
15     appendRowsToSheet("Hi from inside the condition!", sheet);
16   }
17
18   var iterator = 1;
19   while(iterator < 6) {
20     var loopText = "Help, I'm in loop N° " + iterator;
21     appendRowsToSheet(loopText, sheet);
22     iterator = iterator + 1;
23   }
24
25   var endText = 'Hello from the end!';
26   appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) { ...
44 \

```

4 'if statement'



sheet: data object of tab 'control-flow'
 startText: 'Hello from the start!'
 resultSeparateFunction: 'Hello from the separate function!'
 condition: true
 conditionState: 'The condition is: true'

```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: " + condition;
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 function appendRowsToSheet(data, sheet) { ...
35 }
```

4 'if statement'

| | A |
|----|-----------------------------------|
| 1 | Hello from the start! |
| 2 | Hello from the separate function! |
| 3 | The condition is: true |
| 4 | Hi from inside the condition! |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |
| 16 | |
| 17 | |
| 18 | |
| 19 | |
| 20 | |
| 21 | |
| 22 | |
| 23 | |
| 24 | |
| 25 | |
| 26 | |
| 27 | |
| 28 | |
| 29 | |
| 30 | |
| 31 | |
| 32 | |
| 33 | |
| 34 | |
| 35 | |

4

4


```

1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: " + condition;
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) { ...
44 \

```

5 'while loop'



sheet: data object of tab 'control-flow'
startText: 'Hello from the start!'
resultSeparateFunction: 'Hello from the separate function!'
condition: true
conditionState: 'The condition is: true'
iterator: 1
loopText: 'Help, I'm in loop N° 1'

```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: " + condition;
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) { ...
44
```

5 'while loop'

| | A |
|----|-----------------------------------|
| 1 | Hello from the start! |
| 2 | Hello from the separate function! |
| 3 | The condition is: true |
| 4 | Hi from inside the condition! |
| 5 | Help, I'm in loop N° 1 |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |
| 16 | |
| 17 | |
| 18 | |
| 19 | |
| 20 | |
| 21 | |
| 22 | |
| 23 | |
| 24 | |
| 25 | |
| 26 | |
| 27 | |
| 28 | |
| 29 | |
| 30 | |
| 31 | |
| 32 | |
| 33 | |
| 34 | |
| 35 | |
| 36 | |
| 37 | |
| 38 | |
| 39 | |
| 40 | |
| 41 | |
| 42 | |
| 43 | |
| 44 | |

5

5

```

1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: " + condition;
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) { ...
44 \

```

5 'while loop'



sheet: data object of tab 'control-flow'
 startText: 'Hello from the start!'
 resultSeparateFunction: 'Hello from the separate function!'
 condition: true
 conditionState: 'The condition is: true'
 iterator: 1
 loopText: 'Help, I'm in loop N° 1'
 iterator: 2
 loopText: 'Help, I'm in loop N° 2'
 (...)
 loopText: 'Help, I'm in loop N° 5'
 iterator: 6

```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: " + condition;
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) { ...
44
```

5

'while loop'

5

| | A |
|----|-----------------------------------|
| 1 | Hello from the start! |
| 2 | Hello from the separate function! |
| 3 | The condition is: true |
| 4 | Hi from inside the condition! |
| 5 | Help, I'm in loop N° 1 |
| 6 | Help, I'm in loop N° 2 |
| 7 | Help, I'm in loop N° 3 |
| 8 | Help, I'm in loop N° 4 |
| 9 | Help, I'm in loop N° 5 |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |
| 16 | |
| 17 | |
| 18 | |
| 19 | |
| 20 | |
| 21 | |
| 22 | |
| 23 | |
| 24 | |
| 25 | |
| 26 | |
| 27 | |
| 28 | |
| 29 | |
| 30 | |
| 31 | |
| 32 | |
| 33 | |
| 34 | |
| 35 | |
| 36 | |
| 37 | |
| 38 | |
| 39 | |
| 40 | |
| 41 | |
| 42 | |
| 43 | |
| 44 | |

5

5

5

0

1

2

3

4

5

6

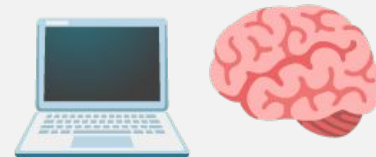
7

8

```

1  var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3  function visualiseControlFlow() {
4    var startText = 'Hello from the start!';
5    appendRowsToSheet(startText, sheet);
6
7    var resultSeparateFunction = separateFunction();
8    appendRowsToSheet(resultSeparateFunction, sheet);
9
10   var condition = true;
11   var conditionState = "The condition is: " + condition;
12   appendRowsToSheet(conditionState, sheet);
13
14   if(condition === true) {
15     appendRowsToSheet("Hi from inside the condition!", sheet);
16   }
17
18   var iterator = 1;
19   while(iterator < 6) {
20     var loopText = "Help, I'm in loop N° " + iterator;
21     appendRowsToSheet(loopText, sheet);
22     iterator = iterator + 1;
23   }
24
25   var endText = 'Hello from the end!';
26   appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) { ...
44 \

```



sheet: data object of tab 'control-flow'
startText: 'Hello from the start!'
resultSeparateFunction: 'Hello from the separate function!'
condition: true
conditionState: 'The condition is: true'
~~iterator: 1~~
~~loopText: 'Help, I'm in loop N° 1~~
~~iterator: 2~~
~~loopText: 'Help, I'm in loop N° 2~~
 (...)

loopText: 'Help, I'm in loop N° 5'
iterator: 6
endText: 'Hello from the end!'

```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: " + condition;
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 function appendRowsToSheet(data, sheet) { ...
35 }
```

B20 | fx | A

| | |
|----|-----------------------------------|
| 1 | Hello from the start! |
| 2 | Hello from the separate function! |
| 3 | The condition is: true |
| 4 | Hi from inside the condition! |
| 5 | Help, I'm in loop N° 1 |
| 6 | Help, I'm in loop N° 2 |
| 7 | Help, I'm in loop N° 3 |
| 8 | Help, I'm in loop N° 4 |
| 9 | Help, I'm in loop N° 5 |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |
| 16 | |
| 17 | |
| 18 | |
| 19 | |
| 20 | |
| 21 | |
| 22 | |
| 23 | |
| 24 | |
| 25 | |
| 26 | |
| 27 | |
| 28 | |
| 29 | |
| 30 | |
| 31 | |
| 32 | |
| 33 | |
| 34 | |
| 35 | |

+ | ≡ | control-flow | Notes | < | >

6

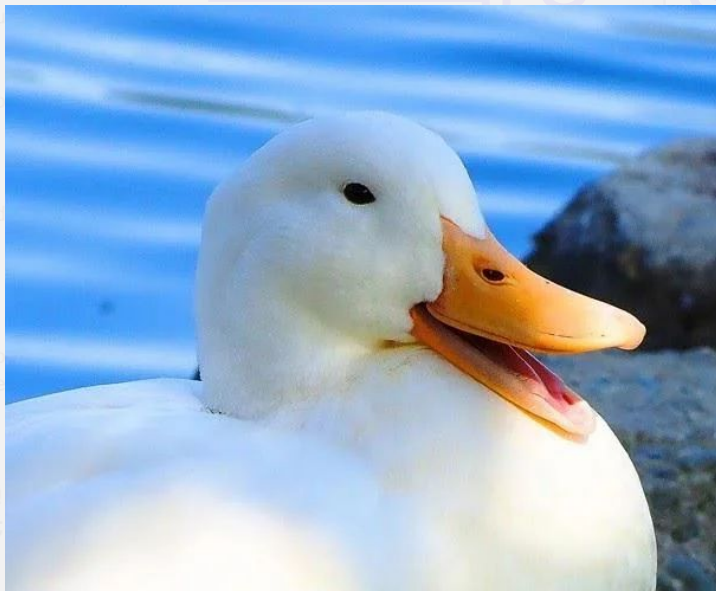
6

6

```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!'; ①
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction(); ②
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: " + condition; ③
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet); ④
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator; ⑤
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!'; ⑥
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!'; ②
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) { ... ① ② ③ ④ ⑤ ⑥
35 }
```

| | A |
|----|-----------------------------------|
| 1 | Hello from the start! |
| 2 | Hello from the separate function! |
| 3 | The condition is: true |
| 4 | Hi from inside the condition! |
| 5 | Help, I'm in loop N° 1 |
| 5 | Help, I'm in loop N° 2 |
| 5 | Help, I'm in loop N° 3 |
| 5 | Help, I'm in loop N° 4 |
| 5 | Help, I'm in loop N° 5 |
| 6 | Hello from the end! |
| 11 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |

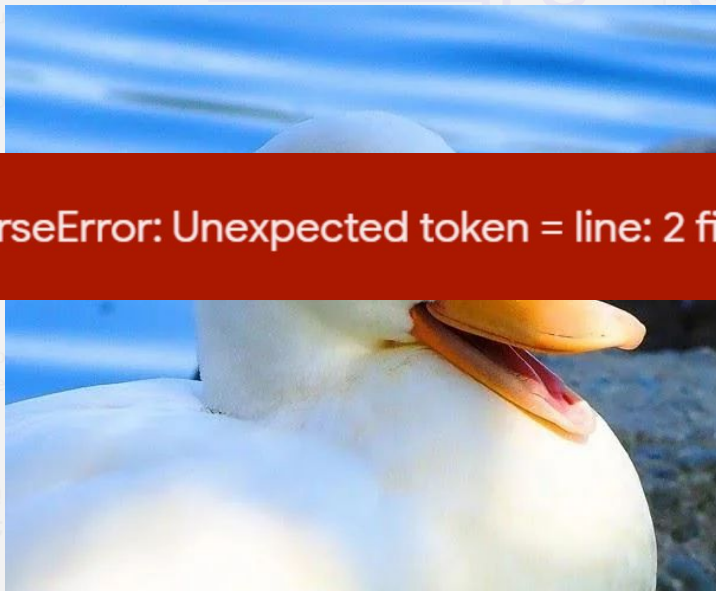
```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: true";
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  var endText = 'Hello from the end!';
26  appendRowsToSheet(endText, sheet);
27 }
28
29 function separateFunction() {
30   var textForSheet = 'Hello from the separate function!';
31   return textForSheet;
32 }
33
34 > function appendRowsToSheet(data, sheet) {
```



| | A |
|----|-----------------------------------|
| 1 | Hello from the start! |
| 2 | Hello from the separate function! |
| 3 | The condition is: true |
| 4 | Hi from inside the condition! |
| 5 | Help, I'm in loop N° 1 |
| 6 | Help, I'm in loop N° 2 |
| 7 | Help, I'm in loop N° 3 |
| 8 | Help, I'm in loop N° 4 |
| 9 | Help, I'm in loop N° 5 |
| 10 | Hello from the end! |

Syntax error: ParseError: Unexpected token = line: 2 file: test.gs

Copy



```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("control-flow");
2 function visualiseControlFlow() {
3   var startText = 'Hello from the start!';
4   appendRowsToSheet(startText, sheet);
5   var resultSeparateFunction = separateFunction();
6   appendRowsToSheet(resultSeparateFunction, sheet);
7   var condition = true;
8   var conditionState = "The condition is: " + condition;
9   appendRowsToSheet(conditionState, sheet);
10  if (condition) {
11    var iterator = 1;
12    while(iterator < 6) {
13      var loopText = "Help, I'm in loop N° " + iterator;
14      appendRowsToSheet(loopText, sheet);
15      iterator = iterator + 1;
16    }
17  }
18  var endText = 'Hello from the end!';
19  appendRowsToSheet(endText, sheet);
20 }
21 function separateFunction() {
22   var textForSheet = 'Hello from the separate function!';
23   return textForSheet;
24 }
25 function appendRowsToSheet(data, sheet) {
26   sheet.appendRow(data);
27 }
```

- 1 Hello from the start!
- 2 Hello from the separate function!
- 3 The condition is: true
- 4 Hi from inside the condition!
- 5 Help, I'm in loop N° 1
- 6 Help, I'm in loop N° 2
- 7 Help, I'm in loop N° 3

Execution log

10:02:29 AM Notice Execution started

10:02:29 AM Error TypeError: Cannot read property 'ID' of undefined
addNewRow @ [funcs.gs:5](#)

Syntax error: ParseError: Unexpected token = line: 2 file: test.gs

Copy



2

1 2 3 4 5 6

Execution log

10:02:29 AM Notice Execution started

10:02:29 AM Error TypeError: Cannot read property 'ID' of undefined
addNewRow @ [funcs.gs:5](#)

Syntax error: ParseError: Unexpected token = line: 2 file: test.gs

Copy

Execution log

12:48:13 PM Notice Execution started

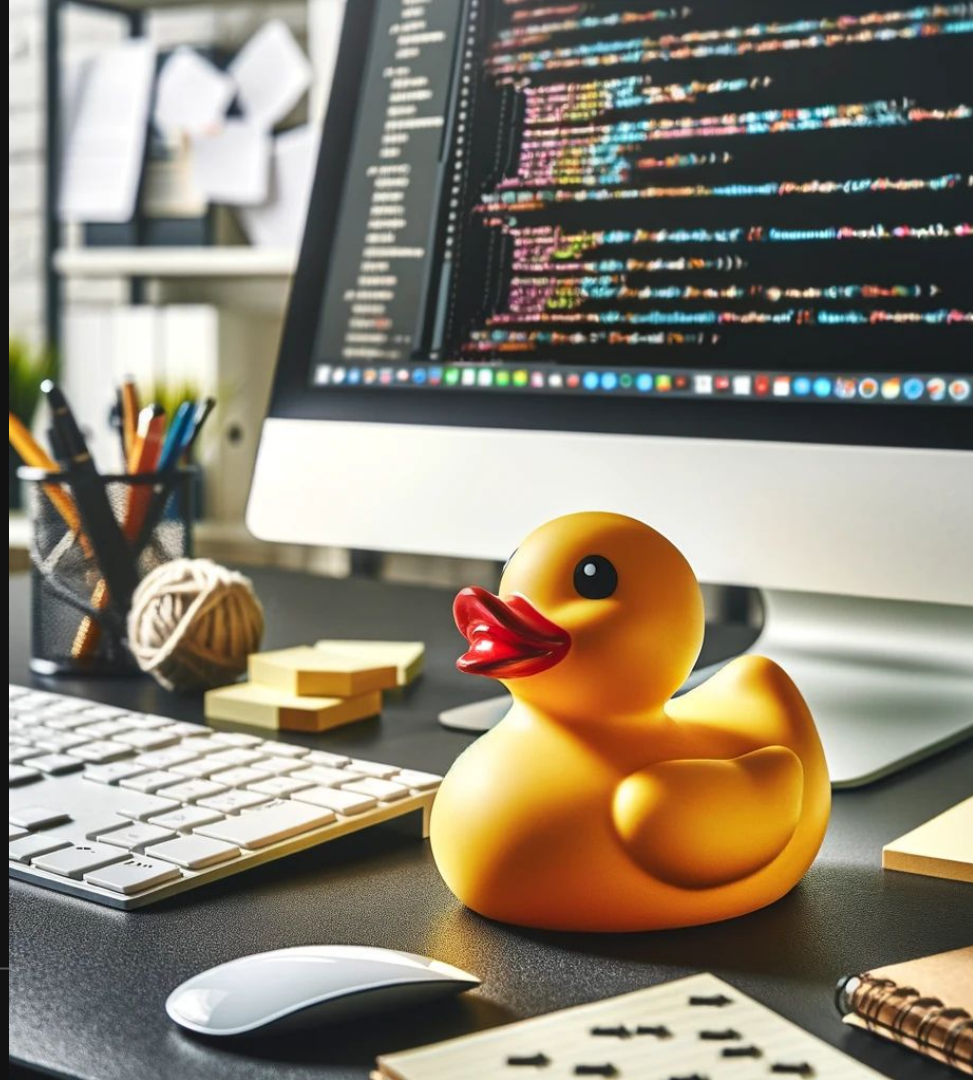
12:48:19 PM Error ReferenceError: sheet is not defined
visualiseControlFlow @ [control-flow.gs:5](#)

CONCEPT

Rubber Duck Debugging

'method of debugging code by articulating a problem in spoken or written natural language'

Source: [Wikipedia Rubber Duck Debugging](#)



CONCEPT

Debugging in the Google Apps Script IDE

Logging & execution log:

Use `console.log()` to verify that your variables have the expected values & data types at specific steps in your control flow.

Debugger & breakpoints:

Use the built-in debugger with breakpoints to look inside the machine's brain at specific moments.

The screenshot displays two panels from the Google Apps Script IDE. The top panel, titled "Execution log", shows a table of execution events:

| Time | Level | Message |
|------------|--------|----------------------------|
| 4:22:05 PM | Notice | Execution started |
| 4:22:04 PM | Info | Women in Tech SEO Festival |
| 4:22:06 PM | Notice | Execution completed |

The bottom panel, titled "Debugger", shows the current state of the script. It includes a toolbar with play, refresh, and step-through icons. The "Call stack" section shows the current function call: `visualiseControlFlow @ control-flow:21`. The "Variables" section shows the local variables:

```
▼ Local  
startText: "Hello from the start!"  
resultSeparateFunction: "Hello from the  
separate function!"  
condition: true  
conditionState: "The condition is: true"  
iterator: 1  
loopText: "Help, I'm in loop N° 1"  
endText: undefined
```

```
1 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName
  ("control-flow");
2
3 function visualiseControlFlow() {
4   var startText = 'Hello from the start!';
5   appendRowsToSheet(startText, sheet);
6
7   var resultSeparateFunction = separateFunction();
8   appendRowsToSheet(resultSeparateFunction, sheet);
9
10  var condition = true;
11  var conditionState = "The condition is: " + condition;
12  appendRowsToSheet(conditionState, sheet);
13
14  if(condition === true) {
15    appendRowsToSheet("Hi from inside the condition!", sheet);
16  }
17
18  var iterator = 1;
19  while(iterator < 6) {
20    var loopText = "Help, I'm in loop N° " + iterator;
21    appendRowsToSheet(loopText, sheet);
22    iterator = iterator + 1;
23  }
24
25  appendRowsToSheet("End of the script", sheet);
26
27 }
```

1 Set 'breakpoints' at the code line where you want to stop for inspection.



4 Resume code execution.

Call stack

visualiseControlFlow @ control-flow:21

Variables

Local

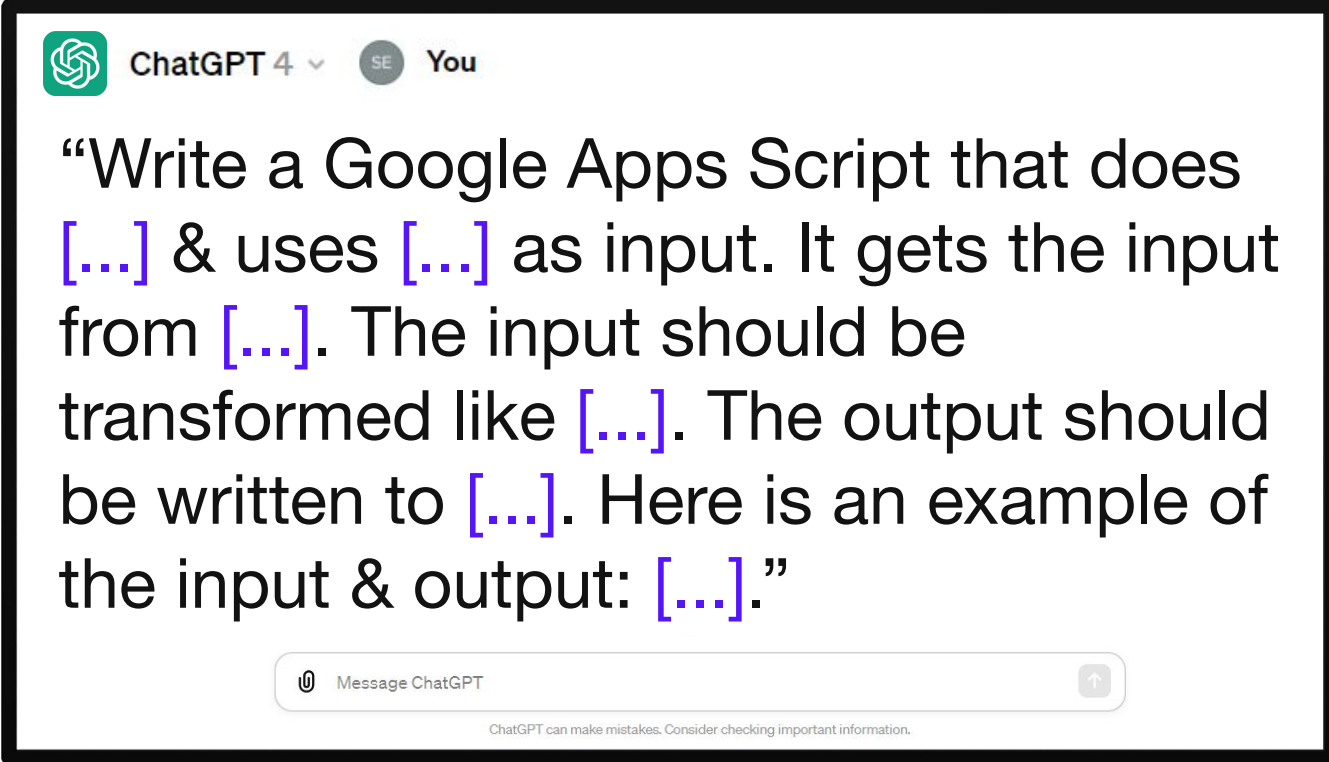
startText: "Hello from the start!"
resultSeparateFunction: "Hello from the separate function!"
condition: true
conditionState: "The condition is: true"
iterator: 1
loopText: "Help, I'm in loop N° 1"
endText: undefined

Global

3 Inspect the current state of variables at this point in the code.

Always have the
control flow in mind when
working with ChatGPT.

Example **prompt structure** in ChatGPT



The screenshot shows a ChatGPT chat window. At the top left is the OpenAI logo, followed by "ChatGPT 4" and a dropdown arrow. To the right is a "SE" icon and the text "You". The main content is a large text prompt: "Write a Google Apps Script that does [...] & uses [...] as input. It gets the input from [...]. The input should be transformed like [...]. The output should be written to [...]. Here is an example of the input & output: [...].". Below the text is a text input field containing "Message ChatGPT" and an upward arrow icon. At the bottom of the chat window, there is a small disclaimer: "ChatGPT can make mistakes. Consider checking important information."

ChatGPT 4 ▾ SE You

“Write a Google Apps Script that does [...] & uses [...] as input. It gets the input from [...]. The input should be transformed like [...]. The output should be written to [...]. Here is an example of the input & output: [...].”

Message ChatGPT

ChatGPT can make mistakes. Consider checking important information.

Example prompt for the SurferSEO API script.



surfer-seo-briefings.gs



surfer-seo-briefings

I want to write a **Google Apps Script** that **sends specific keyword data to an API** and **returns a URL** where we can look at the result. I need to get the keyword data from within a sheet.

The **keywords** for one request are in the **same row but different columns: E, F, G, H, I** - It's possible that the cells are empty and contain no keywords. The rows that need to be sent to the API are **marked in column A with "2 - SurferSEO API | Create Content Briefing"**.

For each row, I want to send the following request format to the specified API endpoint.

```
curl -X POST 'https://app.surferseo.com/api/v1/content_editors' -H 'API-KEY: <your key>' -H 'Content-Type: application/json' --data-raw '{
"keywords": ["seo research","seo article optimization"],
"location": "United States"
}'
```

The response looks like this:

```
{
  "state": "scheduled",
  "permalink_hash": "kKi7n3pkRk7Gw5cxKDiBAbCAybnDTt2z",
  "id": 5632898
}
```

examples from
[SurferSEO docs](#)



Save the **“permalink_hash”** value of the response in **column C** and the **“id”** in **column D** of the processed row.

Knowing basic JavaScript
syntax helps you to
accelerate with ChatGPT.

SEO & Google Sheets are a
great bridge into the
world of coding.

Thank you!

Appendix: Checklists & Resources

Google Sheets script examples: bit.ly/dept-wtsfest-berlin

References to sheet:



Apps Script article:



04

Appendix: Checklists & Resources

CHECKLIST

When to use Apps Script to automate your task?



- Is it a repetitive task?
- Can you write down every step of it?
- Do you have programmatic access to the data sources? (e.g. API credentials)
- Is there a reason why a simple formula cannot solve the issue?

Some examples:

- Long loading times of formulas
- Complex nesting of formulas
- Advanced logic like API calls
- Need to control execution times (e.g. scheduling via triggers)

CHECKLIST

How to get started with Google Apps Script?



- ❑ Get familiar with advanced Google Sheets formulas as much as possible.
- ❑ Learn the basic Apps Script syntax:
 - ❑ Check the resources on the next slide.
 - ❑ But don't get lost in too many tutorials - try to solve your own problems right away!
- ❑ Find a use case where automation makes sense.
- ❑ Ask ChatGPT to write simple scripts and test them in the code editor (see prompting checklist).
- ❑ Try to understand how each step of the syntax works.
- ❑ Ask ChatGPT to clarify all coding concepts that are new to you in simple terms.
- ❑ It's not working? (see debugging checklist)

Master Google Sheets Options & Formulas

[Hannah Rampton @ WTSFest 20 & Advanced Formulas 30 Day Challenge \(Ben Collins\)](#)

Official Google Resources & Code Labs

[Apps Script fundamentals codelabs overview & Hands-on with Google Apps Script](#)

Great SEO Use Cases Built with Google Apps Script

<https://keywordsinsheets.com/free-sheets-scripts/>

Learn Coding using Google Sheets and Apps Script

<https://spreadsheet.dev/learn-coding-google-sheets-apps-script>

Intro to JavaScript on FreeCodeCamp

[Basic JavaScript & Debugging & Basic Data Structures](#) (everything else not needed at first)

Agency Automators with Advanced SEO Use Cases

<https://www.youtube.com/watch?v=aSOxOTG44Vo&t=2328s>

Apps Script Beginner Guides by Ben Collins

[Google Apps Script: A Beginner's Guide & Introduction To Google Apps Script](#)

Apps Script for SEOs by David Sottimano

[TechSEO Boost - Apps script for SEOs](#) (note: examples in old Apps Script IDE before 2021)

Latest News & Guides on Google Apps Script

<https://pulse.appsscript.info/>

CONCEPT

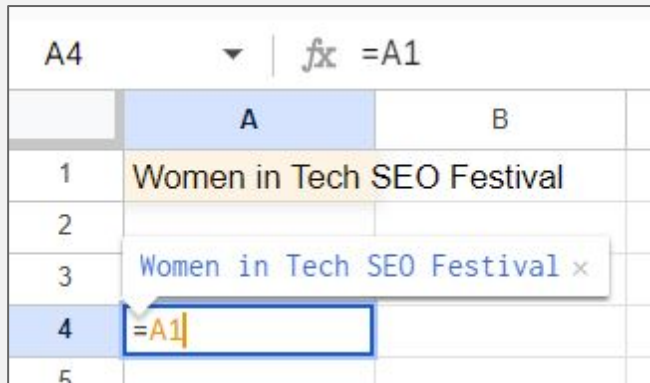
JavaScript Variables

JavaScript variables are like cells in Sheets. Both store a value of a specific data type and make it referenceable.

In JS, you create and name a new variable with `'var [name]'` and assign it a value with `'='`. To reference it later, you can just use its name.

In Google Sheets you simply write the value into a cell and reference it.

Google Sheets Cell



| | A | B |
|---|------------------------------|---|
| 1 | Women in Tech SEO Festival | |
| 2 | | |
| 3 | Women in Tech SEO Festival × | |
| 4 | =A1 | |
| 5 | | |

JavaScript Variable

```

9  var event = "Women in Tech SEO Festival"; // data type 'string'
10
11 function logEventName() {
12     console.log(event);
13 }
14

```

Execution log

| | | |
|------------|--------|----------------------------|
| 4:22:05 PM | Notice | Execution started |
| 4:22:04 PM | Info | Women in Tech SEO Festival |
| 4:22:06 PM | Notice | Execution completed |

CONCEPT

JavaScript Data Types

Data types define the nature of a value in a JS variable or a Google Sheets cell.

They influence the operations that you can do with a value, e.g. you can 'sum up' numbers but not strings.

Use the 'typeof' operator to check the data type of a value in JavaScript, e.g.

`console.log(typeof event);` outputs 'string'.

In Google Sheets, you can use formulas like `TYPE()`, `ISNUMBER()` or `ISTEXT()`.

Google Sheets

| A4 | A | B | C |
|----|---|---|---|
| 1 | TRUE | | |
| 2 | | | |
| 3 | 4 × | | |
| 4 | =TYPE(A1) | | |
| 5 | | | |
| 6 | TYPE(value) | | |
| 7 | EXAMPLE | | |
| 8 | TYPE(C4) | | |
| 9 | | | |
| 10 | ABOUT | | |
| 11 | Returns a number associated with the type of data passed into the function. | | |
| 12 | | | |

Syntax

`TYPE(value)`

- `value` - The data whose type is to be determined.

Notes

- This function returns the following numbers:
 - 1: if `value` is a number
 - 2: if `value` is text
 - 4: if `value` is boolean
 - 16: if `value` is an error
 - 64: if `value` is an array

JavaScript

```
var event = "Women in Tech SEO Festival"; // data type 'string'
var year = 2024; // data type 'number'
var awesome = true; // data type 'boolean'
var cities = ["Berlin", "London", "Philadelphia"]; // data type 'array object'

var countryCities = // data type 'object'
{
  "Germany": "Berlin",
  "United Kingdom": "London",
  "United States": "Philadelphia"
}

var nothing = null; // data type 'null'
var notDefined = undefined; // data type 'undefined'
```

| | | | |
|----|---|-------------------------------------|--------------|
| 9 | <code>var event = "Women in Tech SEO Festival"; // data type 'string'</code> | Women in Tech SEO Festival | |
| 10 | <code>var year = 2024; // data type 'number'</code> | | 2024 |
| 11 | <code>var awesome = true; // data type 'boolean'</code> | <input checked="" type="checkbox"/> | TRUE |
| 12 | | | |
| 13 | | | |
| 14 | <code>var cities = ["Berlin", "London", "Philadelphia"]; // data type 'array object'</code> | Berlin | |
| 15 | | London | |
| 16 | <code>var countryCities = // data type 'object'</code> | Philadelphia | |
| 17 | <code>{</code> | | |
| 18 | <code> "Germany": "Berlin",</code> | Germany | Berlin |
| 19 | <code> "United Kingdom": "London",</code> | United Kingdom | London |
| 20 | <code> "United States": "Philadelphia"</code> | United States | Philadelphia |
| 21 | <code>}</code> | | |

JavaScript data types compared to Google Sheets.

- Strings & numbers are single values written in cell.
- Booleans are true or false - like checkboxes.
- Arrays are like a list of values, like in a column.
- Objects contain a list of key-value pairs. The structure is also used in JSON-LD syntax that we know from structured data.

CONCEPT

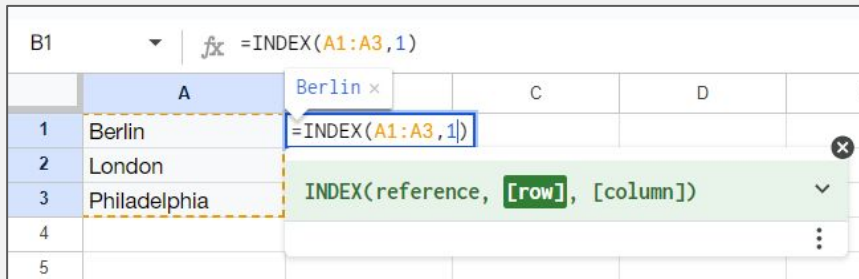
JavaScript Arrays

Arrays are like a list of values, comparable to a column or a row in Google Sheets.

A typical App Script use case: Grabbing & storing data from your sheet to transform it. To do so, you have to access the array values.

How to access arrays? You can use an index, similar to the INDEX() formula in Sheets. Each value in a JS array has an index - starting at '0'. To access a specific array value use `arrayName[index]`.

Google Sheets



| | A | B | C | D |
|---|--------------|-----------------|---|---|
| 1 | Berlin | =INDEX(A1:A3,1) | | |
| 2 | London | | | |
| 3 | Philadelphia | | | |
| 4 | | | | |
| 5 | | | | |

JavaScript

```
var cities = ["Berlin", "London", "Philadelphia"];
```

```
var cityFromArrayOne = cities[0];
var cityFromArrayTwo = cities[1];
var cityFromArrayThree = cities[2];
```

```
function logCityArray() {
  console.log(cityFromArrayOne); // Berlin
  console.log(cityFromArrayTwo); // London
  console.log(cityFromArrayThree); // Philadelphia
}
```

CONCEPT

JavaScript Objects

Objects contain a list of key-value pairs, similar to two connected columns in Google Sheets. The JavaScript object structure is also used in the JSON-LD syntax like for structured data.

A typical App Script use case: Tool APIs send us data in JSON syntax and we want to grab the information relevant to use.

How to access objects? Comparable to the VLOOKUP() formula, we can use the key name, e.g. *objectName[keyName]*.

Google Sheets

| | A | B | D | E |
|---|----------------|--------------|----------------------------|---|
| 1 | Germany | Berlin | =VLOOKUP(A1,A1:B3,2,FALSE) | |
| 2 | United Kingdom | London | | |
| 3 | United States | Philadelphia | | |
| 4 | | | | |
| 5 | | | | |

VLOOKUP(search_key, range, index, [is_sorted])

JavaScript

```
var countryCities = // data type 'object'
{
  "Germany": "Berlin",
  "United Kingdom": "London",
  "United States": "Philadelphia"
}
```

```
var cityFromObjectOne = countryCities["Germany"];
var cityFromObjectTwo = countryCities["United Kingdom"];
var cityFromObjectThree = countryCities["United States"];
```

```
function logCityObject() {
  console.log(cityFromObjectOne); // Berlin
  console.log(cityFromObjectTwo); // London
  console.log(cityFromObjectThree); // Philadelphia
}
```

CONCEPT

JavaScript Functions

'A function is a block of code that performs a specific task.'

JavaScript functions are basically used to encapsulate logic, making that code more reusable and easier to understand.

Functions can take input in the form of parameters and can return a value or output.'

Source: [freeCodeCamp](#)

The screenshot shows a code editor with the following JavaScript code:

```
1
2
3 // This is just a comment
4 function myFirstFunction() {
5   console.log("Hello from the inside of 'myFirstFunction!'");
6 }
7
8
9 /* I can also make comments across
10  | multiple lines. */
11 function mySecondFunction() {
12   logger.log("Hello from the inside of 'mySecondFunction!'");
13   myFirstFunction();
14 }
15
```

The execution log below the code shows the following entries:

| Time | Level | Message |
|------------|--------|--|
| 2:22:36 PM | Notice | Execution started |
| 2:22:36 PM | Info | Hello from the inside of 'mySecondFunction'! |
| 2:22:36 PM | Info | Hello from the inside of 'myFirstFunction'! |
| 2:22:37 PM | Notice | Execution completed |

We reused *myFirstFunction* inside *mySecondFunction* 

CONCEPT

JavaScript Methods

'Methods are functions that are associated with a particular object. They are used to perform a specific task on the object.'

The key: All variables that we define are some kind of object - so you can use methods on them like this:
objectName.methodName()

Sources: playcode.io & dev.to

```
var event = "Women in Tech SEO Festival";  
var bigEventName = event.toUpperCase();  
  
function logBigEventName() {  
  console.log(bigEventName);  
}
```

Execution log

| | | |
|-------------|--------|----------------------------|
| 10:14:27 AM | Notice | Execution started |
| 10:14:28 AM | Info | WOMEN IN TECH SEO FESTIVAL |
| 10:14:29 AM | Notice | Execution completed |

`toUpperCase()` is a JavaScript method that you can use on a [String object](#) like the variable 'event' in the example above (see [docs](#)).

The method's specific task is to take every character and replace it with the capitalised version. It can be compared to the UPPER() formula in Sheets.

Resource Overview: Advanced Features

JavaScript Methods

Pre-defined functions that you can use on specific objects (e.g. a string or an array).

[JavaScript docs](#) & [Apps Script docs](#)

Executions

In the IDE, check the 'Executions' menu icon to see the code that was executed in this specific project. The [Apps Script Dashboard](#) collects this info from all projects that belong to your account.

HTML for Front-ends & Emails

You can write HTML files in the IDE to prepare email templates or even to host proper application frontends.

[Web App Example](#) - [Email Example](#)

Debugger

Use the built-in debugger with breakpoints to assess the state of each variable at specific moments. [Apps Script docs](#)

Calling External APIs

You can use the `UrlFetchApp.fetch()`, `HttpResponse.getContentText()` & `JSON.parse()` to make API requests & parse the JSON output to a data object.

[Apps Script Docs](#) & [tutorial](#)

Version Control in Deployments

Deploy your changes to use the built-in version history. Use deploy as 'library' to save your script progress. [Example](#)

You can also deploy your code as official, public add-on or library.

Triggers

Execute your code automatically based on an event in the spreadsheet ('on open') or at a specific time ('daily between 9-10 am').

[Apps Script docs](#) & [tutorial](#)

Version Control in Github

If you want to develop locally and store your code in Github, use the open-source tool 'clasp'. [Clasp Docs](#) & [Version Docs](#)

Libraries & Services

Use open source libraries written by other users ([library database](#)) and built-in Google services (docs) to speed up creating your logic.

CHECKLIST

How to prompt ChatGPT for scripting?

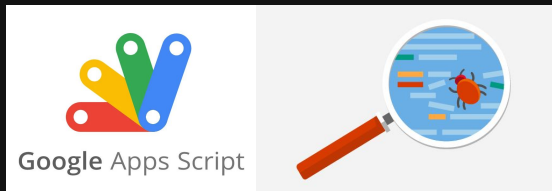


ChatGPT

- ❑ Remember the control flow (🐣) & list the main steps the script has to do in plain language.
- ❑ Specifically list the functional requirements:
 - ❑ Language (Google Apps Script)
 - ❑ Input/output parameters (e.g. “data from tab X, row Y, column Z”)
 - ❑ If possible, give concrete examples of input/output (e.g. from API docs).
 - ❑ Do you want to turn a formula into a script? Give the formula as input.
 - ❑ No sensitive data like API keys.
- ❑ For large scripts: Split the prompt up, test and debug right away.
- ❑ If you know them, use syntax terms like ‘function’, ‘variable’, ‘return’, ‘condition’, ...

CHECKLIST

How to debug in Google Apps Script?



- ❑ Read the error: often it already gives you hints like code lines or missing credentials.
- ❑ Give the error & script to ChatGPT (no sensitive data!).
- ❑ Use the debugger (break points) & execution log (console.log).
- ❑ Use the rubber duck approach and think through the steps of the control flow.
- ❑ Use [typeof](#) to check variable data types.

Are you writing a script from scratch?

- ❑ Debug during writing - console.log() intermediate values at every step.

Does your script come from ChatGPT?

- ❑ Check if you still need to fill in placeholder info like API keys or sheet names/IDs.
- ❑ Split the prompt up, test single parts and debug right away, before you proceed.