

Google Apps Script

Automate everything

Ido Ben Haim



```
filterByOrg = filterByOrg ? study.lead_organization === filterByOrg : true  
filterByStatus = filterByStatus ? study.status === filterByStatus : true  
return (filterByOrg || filterByStatus) ? study : null  
} catch (e) {  
  console.log(e)  
}
```

```
function filterStudies({ studies, filterByOrg, filterByStatus }) {  
  return studies.filter(study => {  
    filterByOrg = filterByOrg ? study.lead_organization === filterByOrg : true  
    filterByStatus = filterByStatus ? study.status === filterByStatus : true  
    return (filterByOrg || filterByStatus) ? study : null  
  })  
}
```


The Big Picture

- Google Apps Script is a platform that lets developers automate their workflow within the google suite
- It works out of the box with most commonly used google apps
- Has a wide range of applications from workflow automation to data collection and Rest API hosting



Some use cases

- Send emails at scheduled times
- Custom processing of form submissions (eg. send emails, parse resumes, auto assign to groups...)
- Custom GUI elements in Google apps
- It is even possible to add GPT to Gmail message writing
- The possibilities are limitless

The screenshot shows the Google Apps Script editor interface. The code defines two functions: `installTrigger()` and `onFormSubmit(e)`. The `onFormSubmit` function processes form data, including email addresses, names, and topics. It uses `SpreadsheetApp` to write data to a spreadsheet and `MailApp` to send an email. Annotations with icons indicate the integration with Google Docs, Google Sheets, and Gmail.

```
1 function installTrigger() {
2   ScriptApp.newTrigger('onFormSubmit')
3     .forSpreadsheet(SpreadsheetApp.getActiveSpreadsheet())
4     .onFormSubmit()
5     .create();
6 }
7
8 function onFormSubmit(e) {
9   let responses = e.namedValues;
10  let email = responses['Email Address'][0].trim();
11  let name = responses.Name[0].trim();
12  let topicsString = responses.Topics[0].toLowerCase();
13  let topics = Object.keys(topicUrls).filter(topicUrl =>
14    | topicsString.indexOf(topicUrl.toLowerCase()) > -1);
15  });
16  let status = '';
17  if (topics.length > 0) {
18    MailApp.sendEmail({
19      to: email,
20      subject: EMAIL_SUBJECT,
21      htmlBody: createEmailBody(name, topics),
22    });
23    status = 'Sent';
24  }
25  else {
26    | status = 'No topics selected';
27  }
28  let sheet = SpreadsheetApp.getActiveSheet();
29  let row = sheet.getActiveRange().getRow();
30  let column = e.values.length + 1;
31  sheet.getRange(row, column).setValue(status);
32 }
33
```


Today's activity

Custom Google Sheets data processing

- Open the [Google Sheet template](#)
- Make a copy of the sheet so you can follow along
- If you ever require additional reference refer to the [Apps Script docs](#)

