AUTOMATING METADATA PULLS VIA DOI & GOOGLE APPS SCRIPTS

HIMAJA KAILASWAR & AAJAY MURPHY
KENNESAW STATE UNIVERSITY

DC SEUG 2020 – GEORGIA SOUTHERN UNIVERSITY

HOW’D WE LAND ON THIS PROJECT?

- The dreaded /facpubs workflow
- Haven’t found comprehensive tools that do this as we’d like
- Himaja was hired on as our Graduate Research Assistant

CONSIDERATIONS BEFORE WE BEGAN

- Which source should we use?
- Plug in DOI and automagically pull metadata
- Ability to customize the metadata that's included to reflect our /facpubs requirements

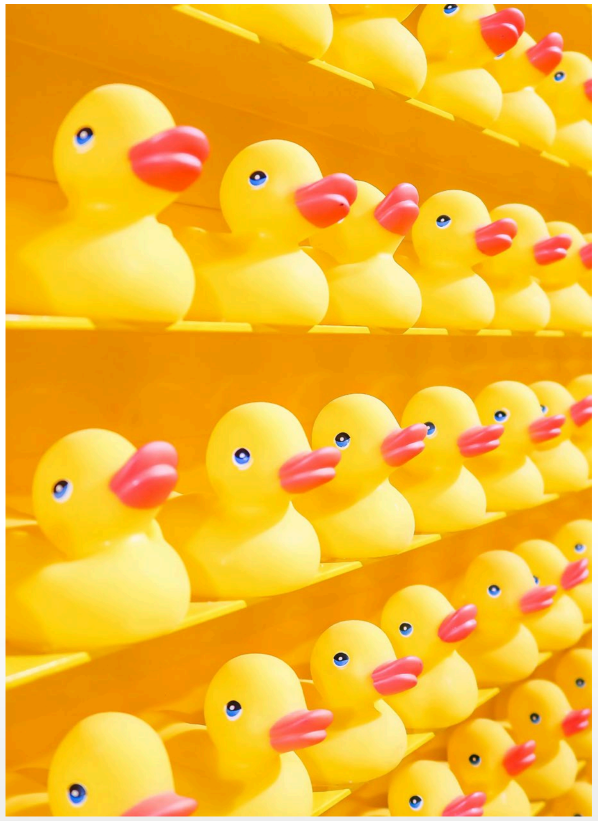
Photo by JOSHUA COLEMAN on Unsplash

INTRODUCTION

- **Google Apps Script** lets you programmatically create and to add custom menus, dialogs, and sidebars to **Google Slides**. You can also integrate **Slides** with other **Google** services like Calendar, Drive, and **Gmail**.
- We can write code in modern JavaScript

CUSTOM MENUS

- Scripts can extend certain Google products by adding user-interface elements that, when clicked, execute an Apps Script function. The most common example is running a script from a custom menu item in Google Docs, Sheets, Slides, or Forms, but script functions can also be triggered by clicking on images and drawings in Google Sheets.

CONTD

- Apps Script can add new menus in Google Docs, Sheets, Slides, or Forms, with each menu item tied to a function in a script. (In Google Forms, custom menus are visible only to an editor who opens the form to modify it, not to a user who opens the form to respond.)
- A script can only create a menu if it is bound to the document, spreadsheet, or form. To display the menu when the user opens a file, write the menu code within an onOpen() function.

CONTD

- The example below shows how to add a menu with one item, followed by a visual separator, then a sub-menu that contains another item. (Note that in Google Sheets, unless you're using the new version, you must use the addMenu() syntax instead, and sub-menus are not possible.) When the user selects either menu item, a corresponding function opens an alert dialog.

EXAMPLE:

```
function onOpen() {
  var ui = SpreadsheetApp.getUi();
  // Or DocumentApp or FormApp.
  ui.createMenu('Custom Menu')
      .addItem('First item', 'menuItem1')
      .addSeparator()
      .addSubMenu(ui.createMenu('Sub-menu')
          .addItem('Second item', 'menuItem2'))
      .addToUi();
}

function menuItem1() {
  SpreadsheetApp.getUi() // Or DocumentApp or FormApp.
     .alert('You clicked the first menu item!');
}

function menuItem2() {
  SpreadsheetApp.getUi() // Or DocumentApp or FormApp.
     .alert('You clicked the second menu item!');
}
```

CREATION:

- Create a new spreadsheet.
- From within your new spreadsheet, select the menu item **Tools > Script editor**. If you are presented with a welcome screen, click **Blank Project**.
- Delete any code in the script editor and paste in the code below.
- Select the menu item **File > Save**. Name your new script and click **OK**.

- Select the menu item **File > Save**. Name your new script and click **OK**.

EXECUTION

- With only a small amount of code, you've created custom functions, added a menu item, and automatically generated a new sheet of information.

FUTURE PLANS AND NEEDS

- Community support and further development
- Integration with existing rights checking Sheets app scripts
- Batch DOI checking
- Tweaking to look and function exactly like the bepress batch upload spreadsheet

CONCLUSION

- Whenever we enter an DOI in the spreadsheet space provided, then after running we can see that entire article information what are required like what all fields we provided in our code, the data will be populated on the spreadsheet.
- By using app scripts, we can get rid of manually searching the article fields information by using Google plugins like Dissemin etc.
- There is a lot more to work on it if we want to make any changes as per the requirements we need.