Janet Fransen, Megan Kocher, and Jody Kempf

Google forms for staff self-assessment

Creating customization

Like many institutions, University of Minnesota recently adopted the Google Apps for Education suite of tools for all students, staff, and faculty. Since the University Libraries moved to Google Apps, we've been discovering new ways to use the tools for productivity and collaboration. One of the most intriguing tools in the suite is Google forms. In essence, Google forms provide a user interface for entering data into a Google spreadsheet: They're an ideal tool for applications such as surveys and evaluations. Recently, we discovered a way to extend Google forms by e-mailing customized information to each survey-taker based on how they answered questions on a form.

Our need seemed straightforward: We were creating a self-assessment tool for library staff to allow them to gauge their level of expertise with various technologies. If the staff member indicated that they needed to know more about a topic, we wanted the tool to tell them how to gain more expertise. We mapped Yes/No statements ("I am aware of, and can explain the key differences between citation managers.") to content (the Moodle site for an *Introduction to Citation Managers* workshop); now we needed a way to give the right content to each person.

Our ideal solution would:

- Provide a way to analyze results in the aggregate.
- Appear uncomplicated and attractive.
- Have a "not-for-grade" feel.
- Allow multiple assessments in different areas.
- Work with existing authentication methods, rather than requiring a separate user name or password.
- Provide context (a paragraph of text) rather than just links.
- Give each staff member a way to return to their custom content without retaking the assessment.

We explored several solutions, but only Google forms supported almost all of the requirements natively.

Scripts, which are supported by Google Docs spreadsheets and forms, are the key to fulfilling our requirement for customized feedback the user can see (almost) immediately, or save for later. Scripts are written in JavaScript and allow the designer to change the way a form or spreadsheet works. We wrote a script that would build an HTML fragment based on the user's responses, and sent that fragment as the body of an e-mail when the user submitted the form.

Working with scripts

In this section, we'll teach you how to create a solution like ours by building an example step by step. We assume that you know how to create a Google form and add questions to it. After setting up a sample form, we'll show you how to:

1) Open the script editor for a Google form with Yes/No questions.

2) Write a procedure that runs when the user submits the form.

3) Add code to send the user an e-mail when they submit the form.

Janet Fransen is engineering librarian, e-mail fransen@umn.edu, Megan Kocher is library assistant III, e-mail: mkocher@umn.edu, and Jody Kempf is instruction and outreach coordinator, e-mail: j-kemp@umn.edu, at the Science and Engineering Library at the University of Minnesota

4) Add a second sheet with responses for each No answer.

5) Add code to generate the body of the e-mail based on the user's responses.

Open the script editor

You can add scripts to any Google spreadsheet. To see for yourself, follow these steps:

1) Create a new form in Google Docs.

2) Give the form a title, check the box next to Automatically collect respondent's username and change the sample questions to multiple choice questions with Yes and No answers, as shown in Figure 1.

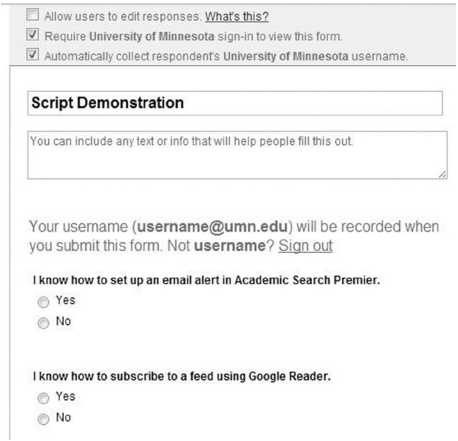

Figure 1. To follow the example, add Yes/No questions to the form. View this article online for detailed images.

If you are working with the public version of Google Docs, you won't see the option to collect the respondent's username. Instead, add a question with a text box for the respondent to enter an e-mail address.

3) Click the Save button to save the form.

4) Go to the spreadsheet for this form by clicking the See Responses button and choosing Spreadsheet.

5) From the Spreadsheet menu, choose Tools | Scripts | Script Editor. The script editor opens, as shown in Figure 2.

Sending an e-mail from code

Once in the editor, you can add code to the

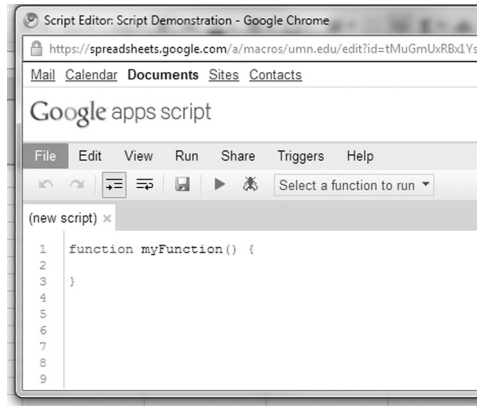

Figure 2. Use the script editor to add JavaScript code.

default myFunction procedure, or write procedures of your own. You can run the procedures you write from the spreadsheet's Tools menu, but more likely you'll want your procedures to run when something happens in the spreadsheet itself. In our solution, the code runs when the user submits an instance of the form.

Much of the code you write will require you to interact with some object: the spreadsheet, the browser, or an e-mail message. The Google Apps Script documentation (http://code.google.com/googleapps/appsscript) describes all of the supported objects.

You can write scripts that read or edit Google Apps objects such as calendar events or contacts, or sends e-mail messages. For our solution, we want to send an e-mail to the person submitting the form.

In a general Google Apps form, you could collect the submitter's e-mail address as part of the form. Because we work in a Google Apps for Education environment, we set up the form to automatically collect the submitter's user name. In either case, the code will need to determine the e-mail address based on the current instance of the form. The information is part of the event, which can be passed to the event handler as an argument.

For a Form Submit event, the event argument contains an array of the values entered in the form's spreadsheet. The array elements are numbered starting at 0 for

the submission's time stamp. In our case, the array element numbered 1 contains the user name.

To make sure the event is passed to the event handler, and change the code to send an e-mail, follow these steps:

1) In the script editor, add a variable, e, to the myFunction declaration:

```
function myFunction(e) {
```

2) Add a line of code to call the MailApp object's sendEmail method:

```
function myFunction(e) {

  MailApp.sendEmail(e.values[1],
"Suggested resources for you","Message
body");

}
```

Note that in this and subsequent code snippets, the code you need to add is in bold. Surrounding lines of code are included for context.

3) Save the script.

4) To re-save the trigger choose Triggers |Current Script's Triggers and then click Save.

5) Click Authorize on the popup window that appears to allow your script to send e-mail.

6) Move to the spreadsheet window.

7) Choose Form | Go to live form to open a new instance of the form.

8) Choose responses for the questions and uncheck the Send me a copy box.

9) Click Submit.

10) Close the acknowledgement window and check your e-mail. Shortly, you should receive the Suggested Resources message.

Adding responses to the spreadsheet

As the solution stands now, the submitter receives the same message every time. To customize the message, we need to build the body of the message based on the submitter's responses. To make our solution easy to maintain, we chose to put the responses into the spreadsheet itself. The code looks for No answers in the set of answers (indicating the submitter wants more information) and adds text from the corresponding spreadsheet cell to the body of the message. By using this method, the code is written once and never touched again. Everything subject to change—the number and text of questions, the responses for each—can be changed from the spreadsheet or form itself rather than in the code.

To make the required changes to the spreadsheet, follow these steps:

1) Add a second sheet to the spreadsheet and name it Response.

2) Select the first column (A) and choose Edit | Named Ranges | Define new range from the menu.

3) Name the range Resources, as shown in Figure 3.

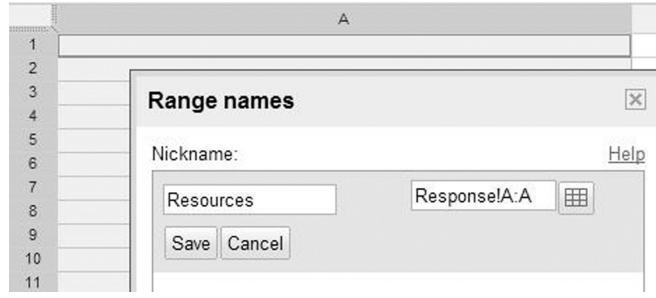

Figure 3. Naming cells or ranges of cells makes it easier to refer to them in the script.

4) Click Save, then click Done to close the Range Names dialog box.

The Resources column will hold the response text. Note that the example only includes text, but you can include HTML if you like. You can use column B to help you keep track of which response matches to which question or column on Sheet1.

5) Add text to the Response sheet, as shown in Figure 4.

	A	B
1	Set up an email alert.	C
2	Subscribe to a feed.	D
3		

Figure 4. The user receives the text in column A if they answer "No" to the corresponding question.

Generating the body of the e-mail

Now that the responses are in place, you can write the code to send appropriate resources. This requires looping through the items in the e.values array, starting with item 2. (Remember that items 0 and 1 contain the time stamp and the user name.) If looping in JavaScript is new to you, see the resource list at the end of the article for some suggested resources.

To get to spreadsheet content not entered in the form, you need to use the Google Spreadsheet object model. The object at the top of this hierarchical model is SpreadsheetApp. You can use the SpreadsheetApp object to find the active spreadsheet, and then refer to sheets, cells, or ranges within that spreadsheet as needed.

To add code for the custom response, follow these steps:

1) Use the SpreadsheetApp object to get the Resources range with the active spreadsheet. Now that the procedure contains more code, the code snippets include only enough for context rather than the entire procedure. In the script editor, add the following code:

```
function myFunction(e) {

  var resourceRange = SpreadsheetApp.
getActiveSpreadsheet().getRangeByName
("Resources");
```

2) Add a variable for the body of the message:

```
var range = SpreadsheetApp.
getActive.Spreadsheet().
getRangeByName("Resources");

var resourceLines = "";
```

3) Figure out how many questions there were on the form:

```
var resourceLines = "";

var len=e.values.length;
```

4) Loop through the responses, looking for No answers:

```
Var len=e.values.length;

for(var i=2; i<len; i++) {

  var response = e.values[i];

  if (response == "No") {
}

}
```

5) For No responses, use the Range object's getCell method to get the response using row,column coordinates, and append its value (as well as an HTML line break) to the resourceLines string:

```
if (response == "No") {

  var newLine = resourceRange.
getCell(i-1,1).getValue();

  resourceLines += newLine;

  resourceLines += "<br/><br/>";

}
```

6) Change the sendMail arguments to send the resourceLines value as the body of the e-mail. Note that if the output includes HTML tags, the method call must be adjusted to send HTML:

Figure 5. The script editor uses colors to make the code easier to read.

```
    resourceLines += "<br/><br/>";

  }

}
```

`MailApp.sendEmail(e.values[1], "Suggested resources for you", resourceLines, {"htmlBody" : resourceLines});`

```
}
```

7) Save the script.

8) At this point, your script should look like Figure 5.

9) Return to the spreadsheet and choose Form | Go to live form.

10) Choose No for the question responses, and uncheck the Send me a copy box.

11) Click Submit.

12) Close the acknowledgement window, and check your e-mail. The response should be similar to Figure 6.

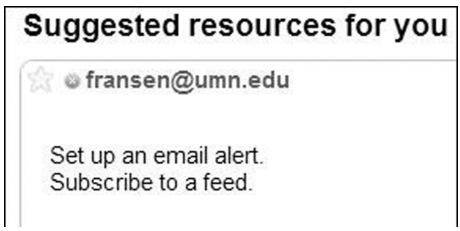

Figure 6. The e-mail message contains basic text, but could include HTML formatting.

Conclusion

In this article we've introduced you to Google's scripting platform by showing you how we put together a solution to our particular problem. You can use Google scripts for much more, both within spreadsheets and in a Google Site. Check out the references at the end of the article for tutorials and many more examples.[2]

Notes

1. To see a working example of the solution presented here, go to http://z.umn.edu/googlescript. You can view the Google spreadsheet and make a copy of your own. To take a sample assessment and receive the response e-mail yourself, go to http://z.umn.edu/umncapim and click the Library Staff Assessment and Training link in the navigation menu on the left.

2. There are extensive resources available on the Web, which will give you samples of code that does something similar to what you need. Try searching for JavaScript and an appropriate keyword(s) to search for helpful examples. There are many JavaScript books available. One popular example is David Flanagan, *JavaScript: The Definitive Guide: Activate Your Web Pages (*Beijing; Farnham: O'Reilly, 2011).

Google offers extensive script documentation on its Web site. Start with these pages:

- http://code.google.com/googleapps/appsscript/.
- http://code.google.com/googleapps/appsscript/guide.html.
- http://code.google.com/googleapps/appsscript/allservices.html.