Google Sites Blog- Comments, RSS, and Automatic Publishing Guide

The purpose of this document is to be a guide for users who want to host a blog on Google Sites and still have access to features such as comments and RSS as well as a guide on how to use Forms and Apps Script to automate the publishing of posts to RSS since Google Sites has no native functionality of this sort.

Creating your site

Creating a site with Google Sites is outside the scope of this document. However you can easily create a new site by visiting sites.new and choosing a template. Create a page named “Blog” to use as your index, new blog posts can be created as subpages. I recommend hiding new pages from navigation, otherwise they will be added as a link under the Blog menu item and will clutter the site navigation. I created a Google Sheet that can be embedded in a page and can act as an automatically updating index by pasting the link to an RSS feed in the Feed Setting worksheet. Here is the customized embed code to make it look like part of your website natively.

```
<iframe src="https://docs.google.com/spreadsheets/d/[docId of the google sheet]/pubhtml?gid=0&widget=false&headers=false&chrome=false" width="100%" height="100%" frameBorder="0"</iframe>
```

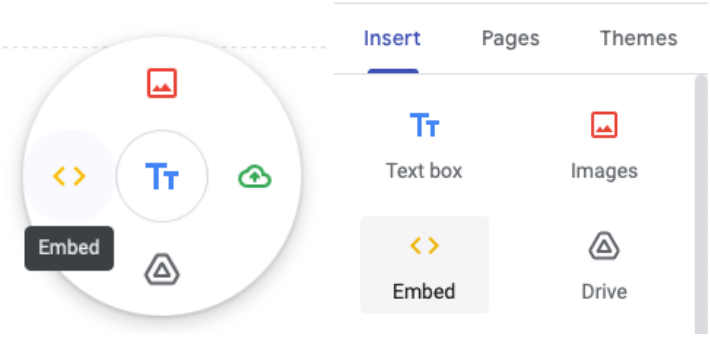
Embedded Google Sheets are not responsive to screen size and do not look great on mobile so I decided to just manually update the index page on my blog instead.

Adding a comment section

Using Disqus

Create a publishing account for disqus and follow the instructions for the Universal Embed Code

At the bottom of your blog post, create a new embed. This can be done by double-clicking a blank section of the page and choosing embed, or by choosing embed from the Insert menu.



Copy and paste the universal embed code into the Embed Code section of the Embed from the web pop-up that opens when you insert a new embed element. Click next to see a preview and click Insert

Replace the PAGE_URL with the full URL of the published page (that is, the URL the page will have when visited by a user, not the URL to the edited page, eg. https://www.kmstrube.net/blog/11) and PAGE_IDENTIFIER (the portion of the URL following the domain (the portion after https://sites.google.com for a default site, or after your custom domain if your site is set up with a custom domain, eg. blog/11)

Remove the /* above the var disqus_config = function () { line. And remove the */ below the }; line. An example of my embed code is below:

```
<script>
    /**
    *  RECOMMENDED CONFIGURATION VARIABLES: EDIT AND UNCOMMENT THE SECTION BELOW TO INSERT DYNAMIC VALUES FROM YOUR PLATFORM OR CMS.
    *  LEARN WHY DEFINING THESE VARIABLES IS IMPORTANT: https://disqus.com/admin/universalcode/#configuration-variables    */

    var disqus_config = function () {
    this.page.url = 'https://www.kmstrube.net/blog/11';  // Replace PAGE_URL with your page's canonical URL variable
    this.page.identifier = 'blog/11'; // Replace PAGE_IDENTIFIER with your page's unique identifier variable
    };

    (function() { // DON'T EDIT BELOW THIS LINE
    var d = document, s = d.createElement('script');
    s.src = 'https://[redacted].disqus.com/embed.js';
    s.setAttribute('data-timestamp', +new Date());
    (d.head || d.body).appendChild(s);
    })();
</script>
```

Using Mastodon

Daniel Pecos, has a good post about using mastodon posts as a comment section. For Google sites embeds, including the CSS in a style tag isn't respected when the site is rendered. The style needs to be included in the style attribute of every element. Some time I may attempt to refactor Dan's code so that the styling works with Google Sites but be aware that that isn't working for now.

Publishing with RSS

RSS requires a XML file that is filled out according to the RSS or Atom spec. Create a new Google Doc and fill out the feed settings according to the RSS 2.0 spec. The RSS feed will always start like this

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
   <channel>
```

The most essential elements in the channel section are <title>, <link>, and <description>. Hopefully it's somewhat self explanatory but <title> is the Title of your Blog, <link> is the URL to the homepage, and <description> is a short description of your blog as seen in feed readers. There are other optional elements for the channel section, check out the spec linked above for a description of each optional element. See the example below to get started.

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
   <channel>
      <title>[Title of Blog]</title>
      <link>[Link to Home Page]</link>
      <description>[short description]</description>
      <category>[arbitrary category, can have multiple]</category>
      <copyright>[copyright info]</copyright>
  </channel>
</rss>
```

To add posts to the feed you need to add an <item> element within the <channel> element. An item just needs either a <title> or a <description>. <link>, <guid>, and <pubDate> are essential for properly organizing content in the feed reader though. <title> is your post title. <description> can be either the full content encoded in ![CDATA] tags or just a teaser to get readers to click the full content posted on your blog. <link> is the URL to your post on your blog. <guid> is a unique identifier. Typically this is just the link with the isPermaLink attribute set to true. <pubDate> is the date the post was published, formatted according to the RFC 3339 spec. See the example of an item below.

```
<item>
   <title>[Blog Post Title]</title>
    <link>[link to post]</link>
    <description>
     [A short text description or entire HTML of blog post wrapped in ![CDATA] tag]
    </description>
    <category>[arbitrary category, can have multiple]</category>
    <guid isPermaLink="true">[unique ID, link to post if isPermaLink is set to true]</guid>
    <pubDate>[RFC 3339 Formatted Date String (DDD, DD MMM YYYY hh:mm:ss Z)]</pubDate>
</item>
```

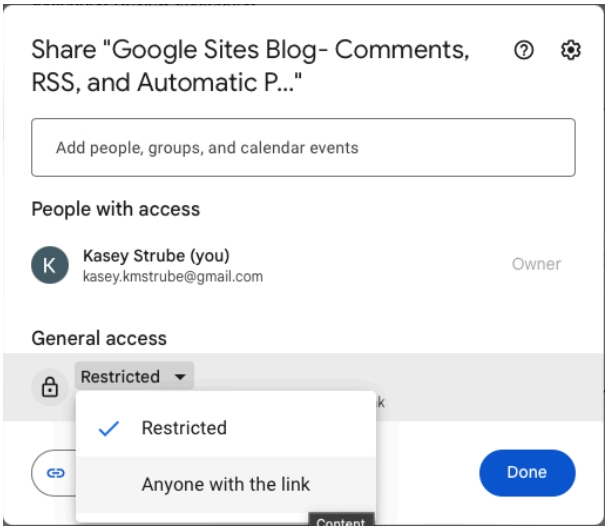
You can insert the item inside the channel section, anywhere below the title is fine. In my example I do it below the <copyright> element. Additional items can be inserted below the latest item and they do not need to be in chronological order if you are using the <pubDate> element. Be sure to close the XML with the closing </channel> and </rss> tags.

```
  </channel>
</rss>
```

To get you started, here is an example of my RSS feed with a single item.

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
   <channel>
      <title>KMSTRUBE.NET Blog</title>
      <link>https://www.kmstrube.net/blog</link>
      <description>Reviews and Ramblings from yours truly</description>
      <category>Technology</category>
      <copyright>Copyright 2024 KMSTRUBE.NET</copyright>
      <item>
          <title>Review: Super Mario Bros.</title>
          <link>https://www.kmstrube.net/blog/2</link>
          <description>In this post I give my review of the classic NES game.</description>
          <guid isPermaLink="true">https://www.kmstrube.net/blog/2</guid>
         <category>Retro Gaming</category>
         <category>Review</category>
          <pubDate>Thu, 01 Sep 2022 13:00:00 CST</pubDate>
       </item>
</channel>
</rss>
```

Once you are ready to publish your RSS Feed, share the document using the “Anyone with the link” option. Leave the permission level at viewer and click “Done”.



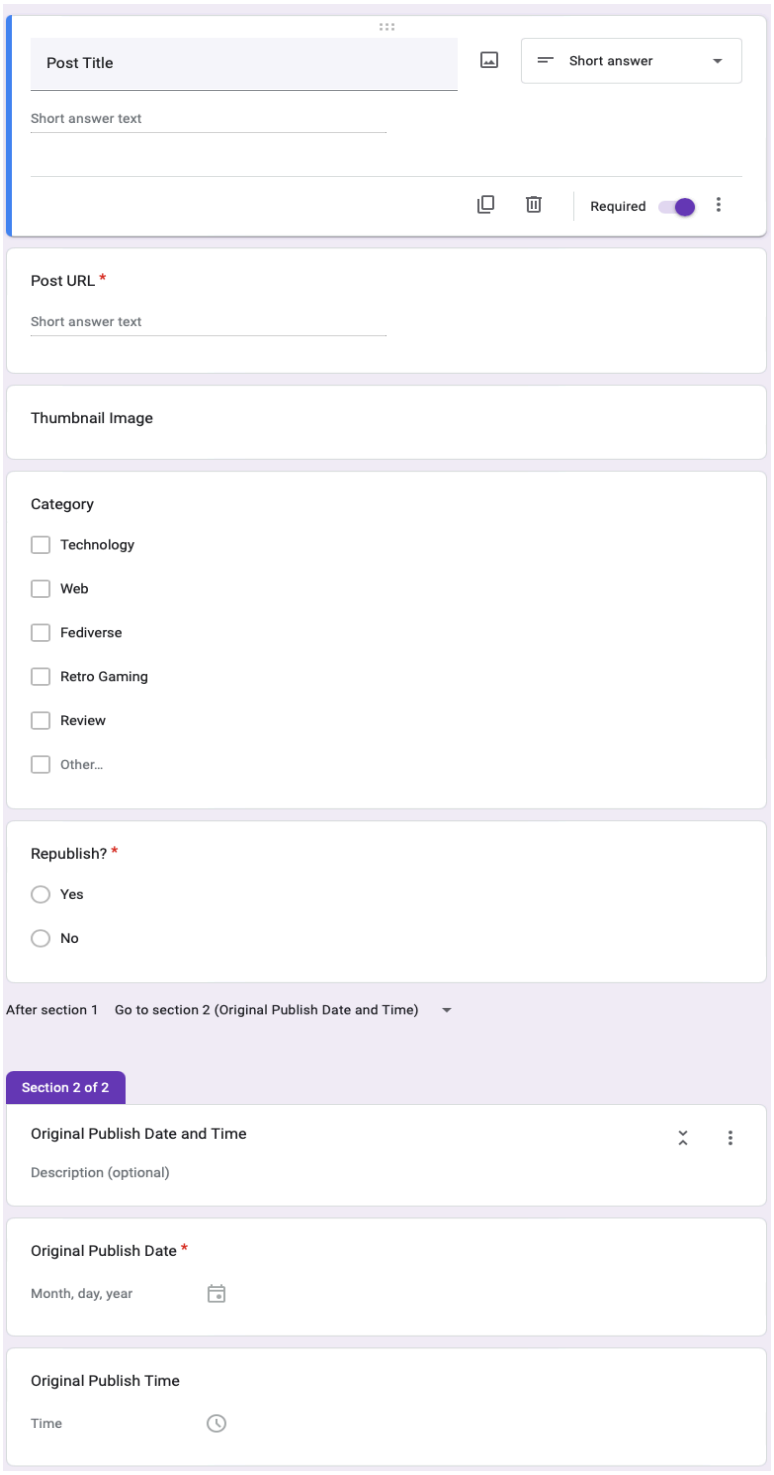
Open the URL in the address bar and take note of the Google Doc ID, the section between "...d/" and "/edit" highlighted below.

The link to your RSS feed with the txt extension is as follows, replacing [googledocid] with the Google Doc ID as determined above.
https://docs.google.com/feeds/download/documents/export/Export?id=[googledocid]&exportFormat=txt

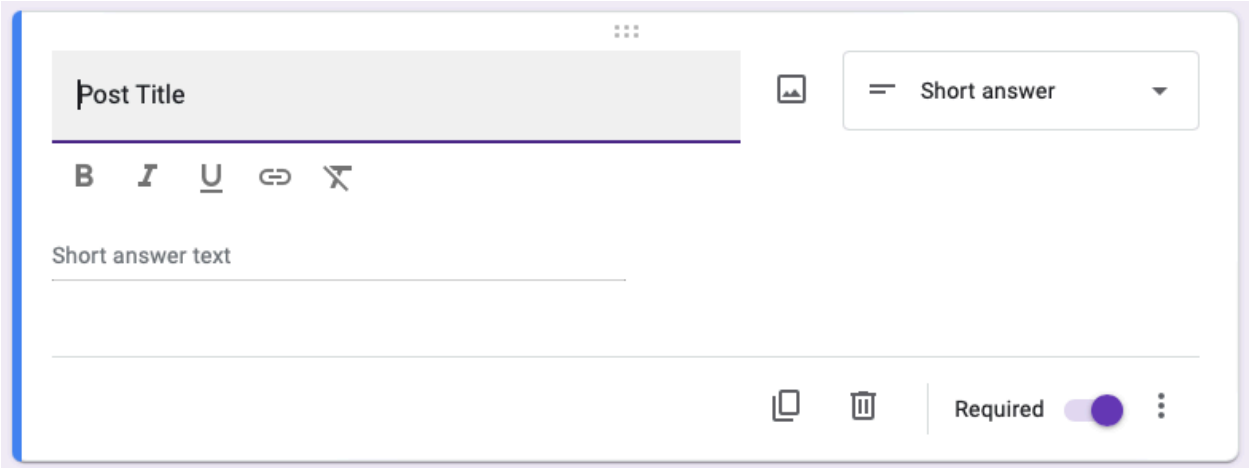
Automating RSS Publishing with Google Forms and App Script

Create the Form

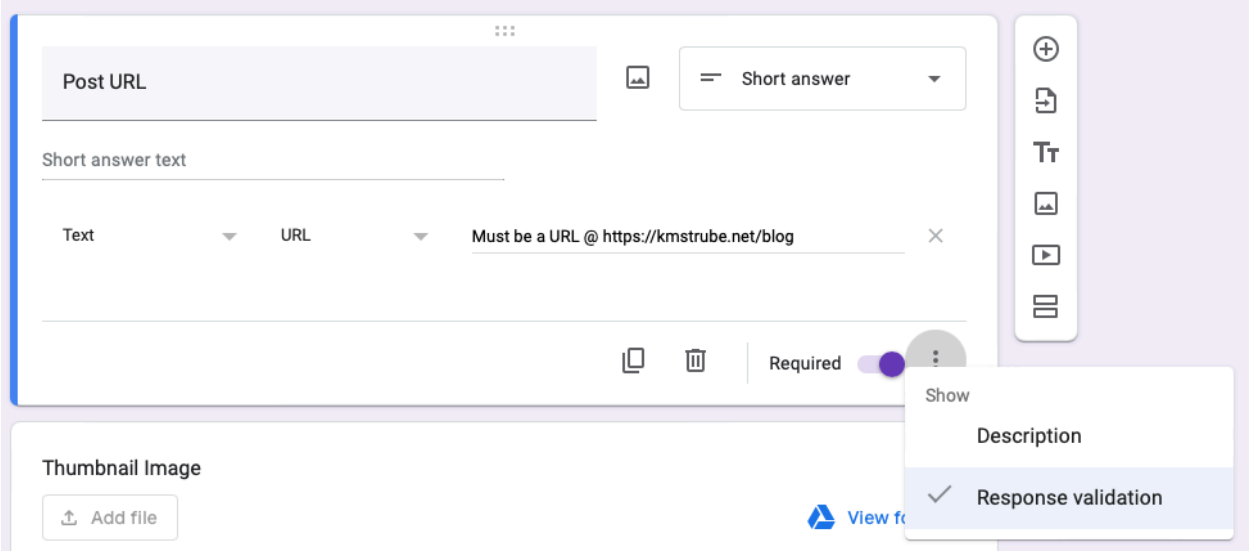
Create a new form in Google Drive. Use my form as an example for the kind of fields your form will need.



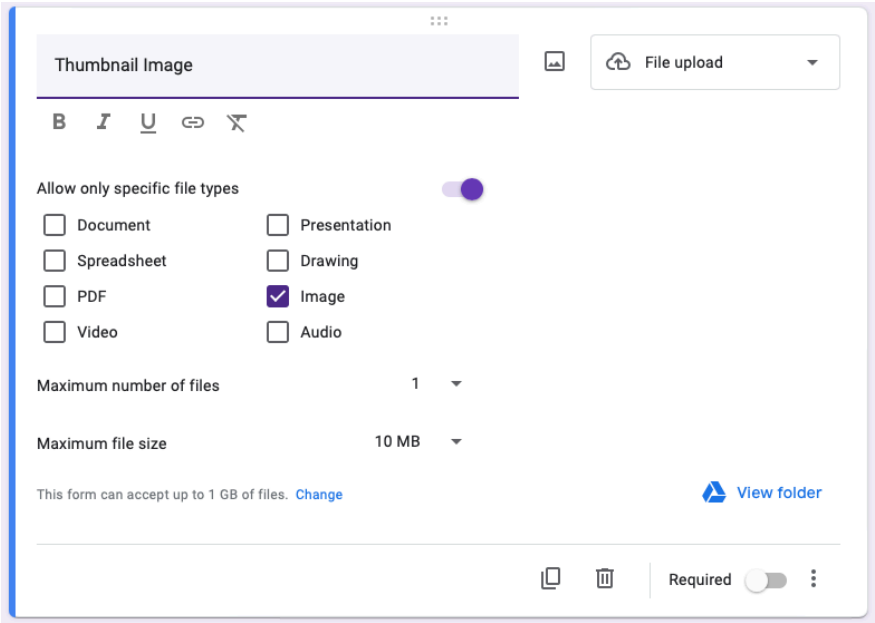
Using my form as an example, I take the post title as a required “short answer” input with no validation.



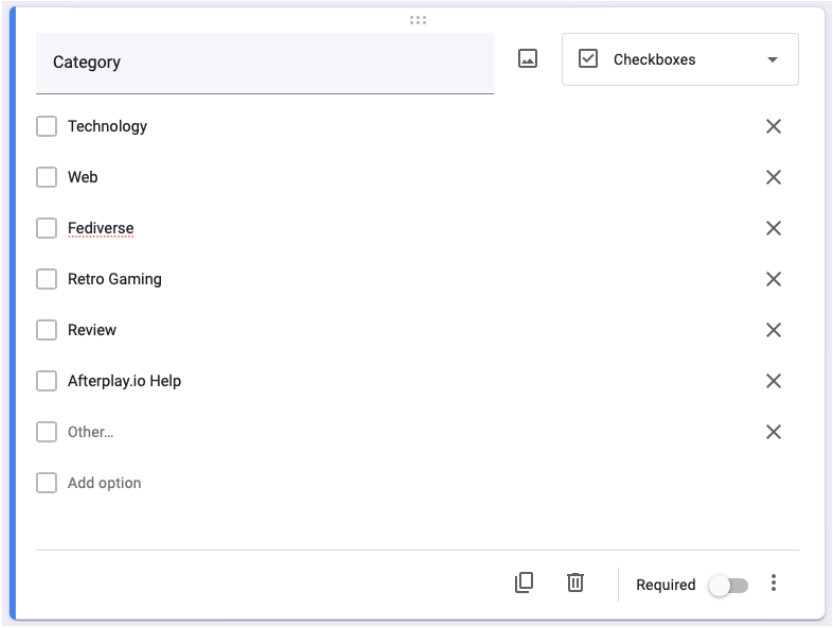
I take the post URL as a required “short answer” input with URL validation. I do additional validation based on domain in the script that runs on submission.



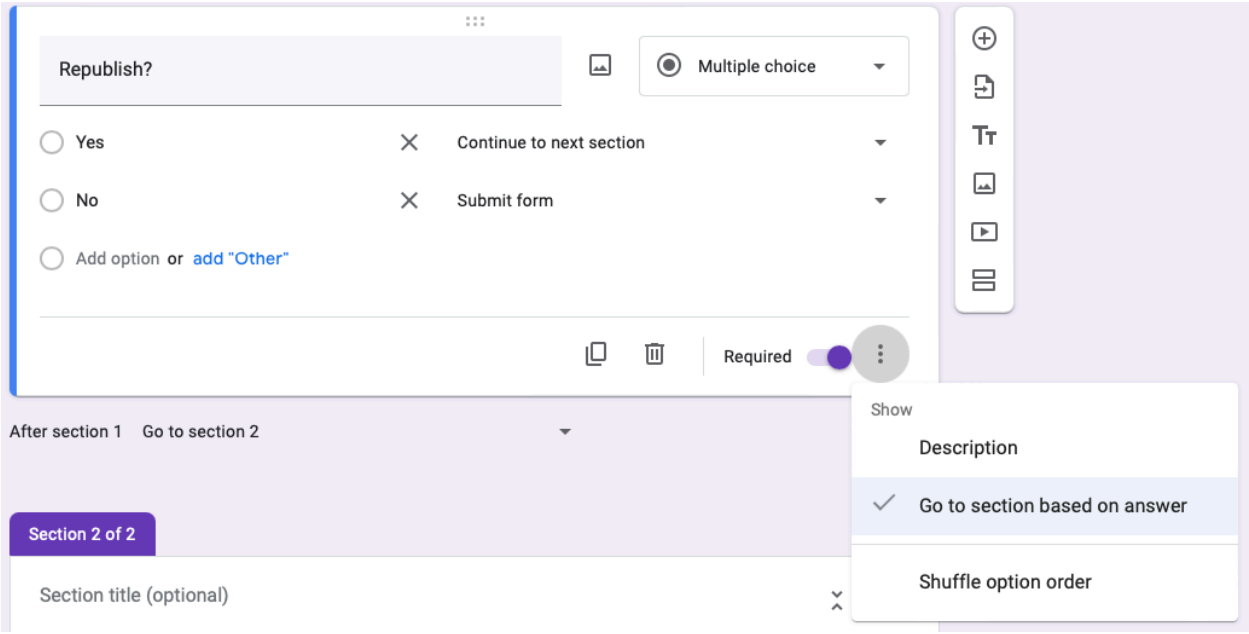
I have a “File Upload” input for Thumbnails. This is for prepending a thumbnail to the description field to make it look nice in feed readers and to give the post a bit more life since images are stripped by Google Sites automatically when exporting the HTML. I set the file type to Image and number of files to 1. I set the file limit to 10MB and total limit to 1GB.



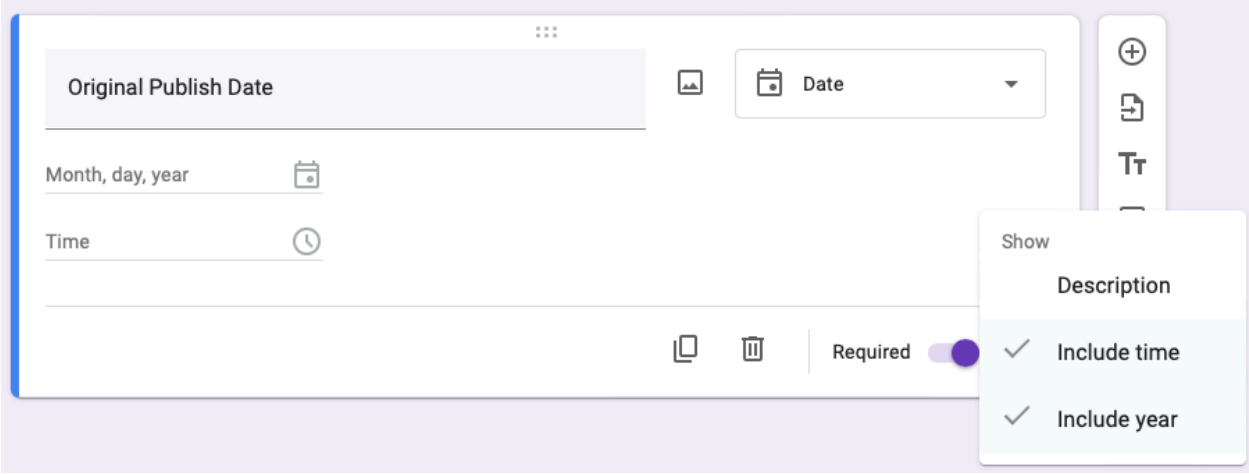
For setting the post category(ies), I have a checkboxes input field. I just set arbitrary categories of things I'd like to write about as well as including an "Other.." option in case I want to add a new category at the time of submission.



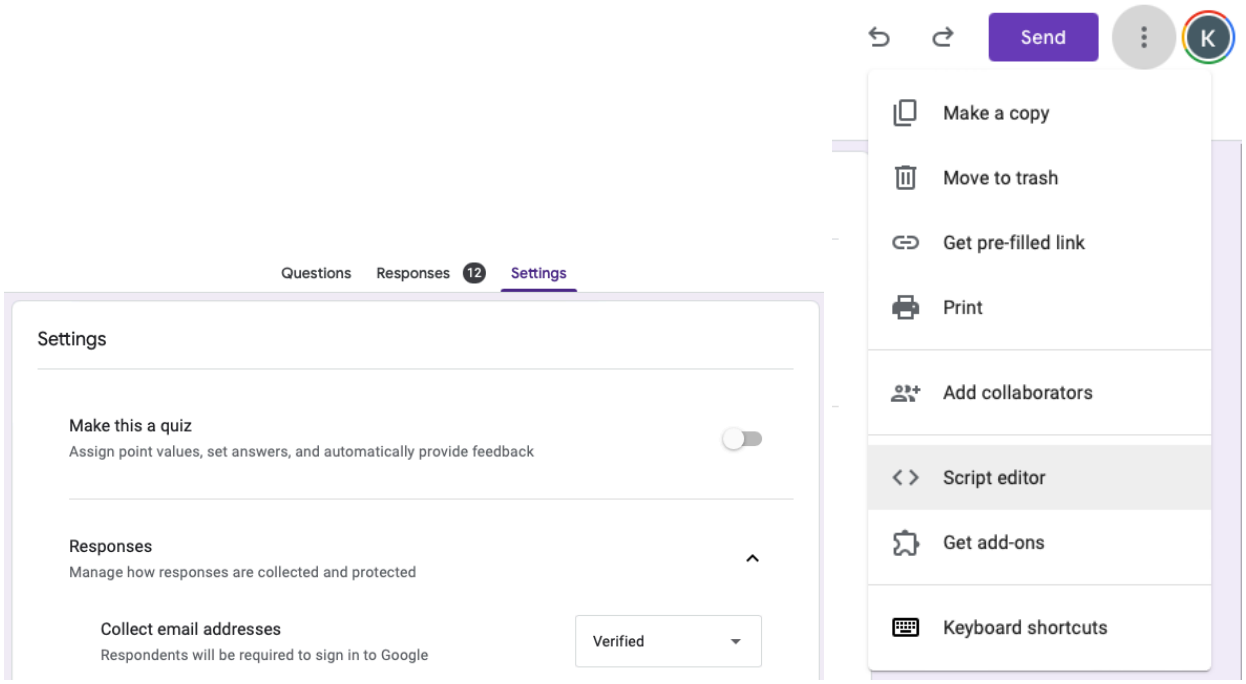
Finally for determining pubDate, I have a bit of navigation logic using a Yes/No field. If Re-publish? is marked "Yes" the form navigates to section 2. If marked "No" it submits the form and the submission time is used as the post's pubDate.



For posts that are being updated or republished, section 2 consists of a single Date/Time field with the Year and Time options set.

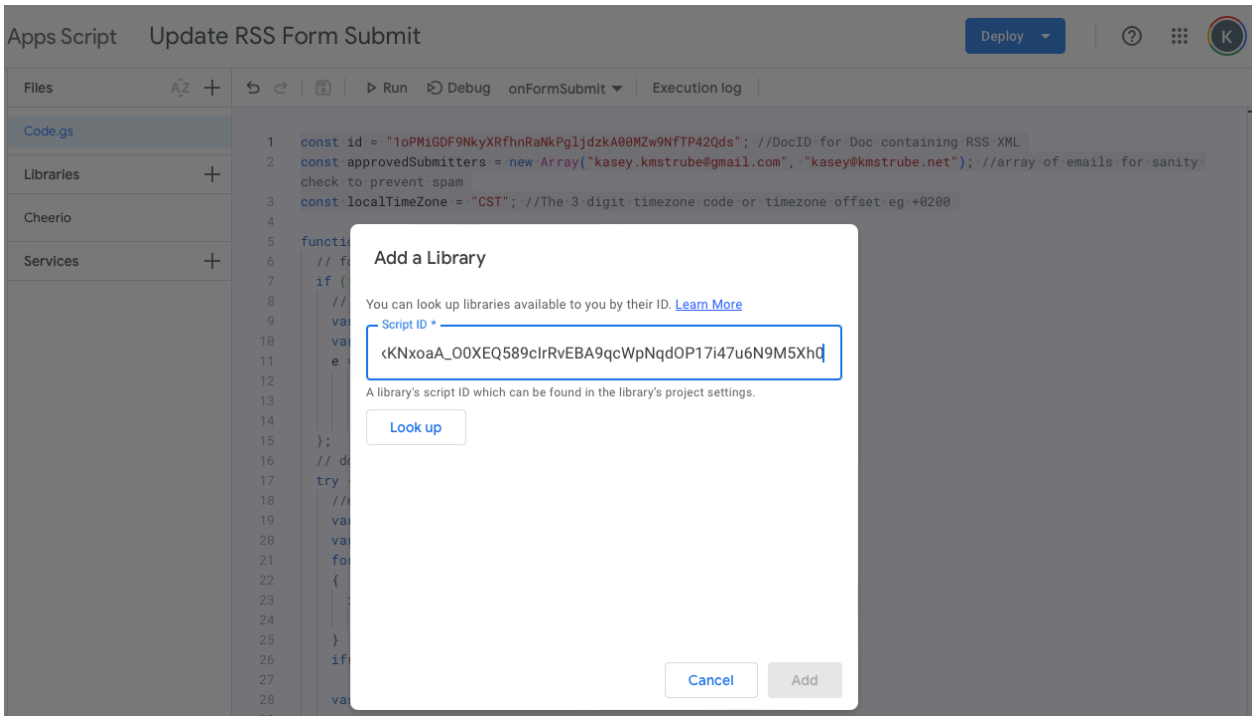


Once you have your form configured to your liking, set the form to collect email addresses as “verified” (used for email verification so that the internet as a whole can’t spam your feed with invalid updates). And enable the script editor to create a Google Apps Script project that runs on form submission.



Create the Script

To parse the HTML of the blog post, the Cheerio library is needed. Select the + symbol to the right of "Libraries" and pasting in the Script ID for Cheerio (1ReeQ6WO8kKNxoaA_O0XEQ589clrRvEBA9qcWpNqdOP17i47u6N9M5Xh0)



Copy and Paste the code included below into your code editor window. Edit the 6 global variables to fit your enviroment.

```
const id = "[googledocid]"; //DocID for Doc containing RSS XML
const approvedSubmitters = new Array("email@example.com",
"example@email.net"); //array of emails for sanity check to prevent spam
const approvedDomains = new Array("www.kmstrube.net"); //array of
approved domains that are hosted on Google sites. to prevent code
breaking on html that doesn't match spec.
const localTimeZone = "CST"; //The 3 digit timezone code or timezone
offset eg +0200
const parentDiv = "div.JNdkSc-SmKAyb.LkDMRd";
const contentSection = ".yaqOZd";
```

id is the DocID of the Google Doc containing the feed XML that we determined earlier. As a refresher you can find it by opening the document, grabbing the URL and copy-pasting the part between “https://docs.google.com/document/d/“ and “/edit”. **approvedSubmitters** is an array of email addresses of the users allowed to contribute to the RSS feed. **approvedDomains** is an array of domains that are hosted on Google Sites and are approved to be posted to the RSS feed. **localTimeZone** is for properly appending the timeZone code to the pubDate. It is off by an hour for 3 quarters of the year because of Daylight Savings Time but I don't care enough to implement a fix since not all locales have daylight savings time and it doesn't bother me that much if the time is off 1 hour. **parentDiv** is a class selector for the parent div that holds images in the blogHTML, needed to drop the broken images from the HTML. **contentSection** is a class sector for the content blocks in the blog page, ensuring that only content and not the page header or footers are in the blog HTML. **parentDiv** and **contentSection** were always the same for all the sites I created in my testing. It should be fine to leave it static for yours but I can’t confirm. In my onFormSubmit function there is a switch statement that loads answers from the form into variables. If you reworded the questions on the form, make sure the case statements in that switch match the questions exactly as they are worded in your form. Here is the full code of my script below.

```
const id = "[googledocid]"; //DocID for Doc containing RSS XML
const approvedSubmitters = new Array("email@example.com",
"example@email.net"); //array of emails for sanity check to prevent spam
const approvedDomains = new Array("www.kmstrube.net"); //array of
approved domains that are hosted on Google sites. to prevent code
breaking on html that doesn't match spec.
```

```
const localTimeZone = "CST"; //The 3 digit timezone code or timezone
offset eg +0200
const parentDiv = "div.JNdkSc-SmKAyb.LkDMRd";
const contentSection = ".yaqOZd";
//load URI code
eval(UrlFetchApp.fetch('https://cdnjs.cloudflare.com/ajax/libs/URI.js/1.
19.11/URI.min.js').getContentText());

function onFormSubmit(e) {
// for tests
if (!e) {
// test object
var form = FormApp.getActiveForm()
var responses = form.getResponses();
e = {"authMode":"FULL",
"response": responses[responses.length - 1], // last response
"source": form,
"triggerUid":"5125265"};
};
// do your job here
try {
//email sanity check
var mail = e.response.getRespondentEmail();
var errFlag = true;
for(var n = 0; n < approvedSubmitters.length; n++)
{
if(approvedSubmitters[n] == mail)
errFlag = false;
}
if(errFlag) throw new Error( "Non approved submitter." );

var i; //init i as counter
var url; //"Post URL" => url
```

```
var title; //"Post Title" => title
var thumb; //"Thumbnail Image" => inject into blog HTML
var category; //"Category" = > categories
var source; //Not currently part of form, for reposts
var sourceUrl; //Not currently part of form, for reposts
var date; //"Original Publish Date" = for pubDate
var author; //Not currently part of the form, all posts are authored by
me
var enclosure; //Not currently part of the form, not podcasting.
var timeFlag = false; //set flag for republishing old content to false

answers = e.response.getItemResponses(); //get responses
for(i = 0; i < answers.length; i++){ //loop through responses and save
them to vars
//load responses into vars
switch(answers[i].getItem().getTitle()){
case "Post URL":
url = answers[i].getResponse();
break;
case "Post Title":
title = answers[i].getResponse();
break;
case "Thumbnail Image":
thumb = answers[i].getResponse();
break;
case "Category":
category = answers[i].getResponse();
break;
case "Republish?":
if(answers[i].getResponse()=="Yes") timeFlag = true; else timeFlag =
false;
break;
case "Original Publish Date":
```

```
date = answers[i].getResponse();
break;
default: break;
}
}

//domain sanity check
errFlag = true;
for(n = 0; n < approvedDomains.length; n++)
{
if(new URL(url).hostname.toLowerCase() ===
approvedDomains[n].toLowerCase())
errFlag = false;
}
if(errFlag) throw new Error( "Non approved domain." );

//process vars
var blog = getBlogHTML(url, true); //get blog HTML
if(!blog) throw new Error ("Page not found");
var blogHtml;
//create HTML for uploaded thumbnail
if(thumb){
var img = DriveApp.getFileById(thumb);
img.setSharing(DriveApp.Access.ANYONE, DriveApp.Permission.VIEW);
var imgHtml = '<img src="https://lh3.googleusercontent.com/d/' + thumb +
'" style="max-width:560px;max-height:560px" alt="Thumbnail"/>';
blogHtml = imgHtml + blog;
} else blogHtml = blog;
//create pubDate
var pubDate;
if(timeFlag){
pubDate = buildRFC822Date(date);
} else {
```

```
pubDate = buildRFC822Date(new Date());
}

var item = formatItemForRSS(title, blogHtml, author, category, url,
enclosure, url, pubDate, source, sourceUrl);

updateRSS(id, item);
}
catch (err) {
throw err;
}
return 0;
}

function getBlogHTML(url, pop = false){ //URL: url to blog post on
google sites; pop: if set to true then pop off last section of post. IE
if there comment section.
const htmlContent = UrlFetchApp.fetch(url).getContentText(); //get blog
post from URL
const $ = Cheerio.load(htmlContent); //load blogpost into cheerio object

//sanity checks
if(!$) return false;
var error404 = '<div class="ZQSezd"><h1 class="NVoTp">404</h1><p
class="nsUy4b">The page you have entered does not exist</p><p><a
class="Clk3Bd" href="/home">Go to site home</a></p></div>';
if ($.html().toString().includes(error404) ) return false;

var blog = new Array(); //save the blog contents to array
var i = 0; //init counter
var html = $(contentSection).first(); //load first section of content
var bloghtml; var temp;
var img; var breakimg; var removeme; //init working vars
```

```
do { //loop trhough post sections while html is valid
img = html.find('img').first(); //get first image
do{ //loop until all images are removed from post section
removeme = img.parentsUntil(parentDiv).last().html();
temphtml = html.html();
bloghtml = temphtml.replace(removeme,"");
img = img.next();
} while (img.html() !== null)

blog[i] = bloghtml; // write current section to array

html = html.next(); //load next section
i++; //increment counter
} while (html.html() !== null && i < 100)

if(pop) blog.pop(); //pop last section

i = 1; //reinit counter, start at 1 because we don't care about the
title card
var post = ""; //var for flattened blog array
while( i < blog.length ){ //loop through array to flatten it
post += blog[i];
i++;
}
return post;
}

function formatItemForRSS(title, description, author, category, link,
enclosure, guid, pubDate, source, sourceUrl) {
/*
var title; //post title <title> element
var description; //post content <description> element
var author; //post author <author> element, probably unused.
```

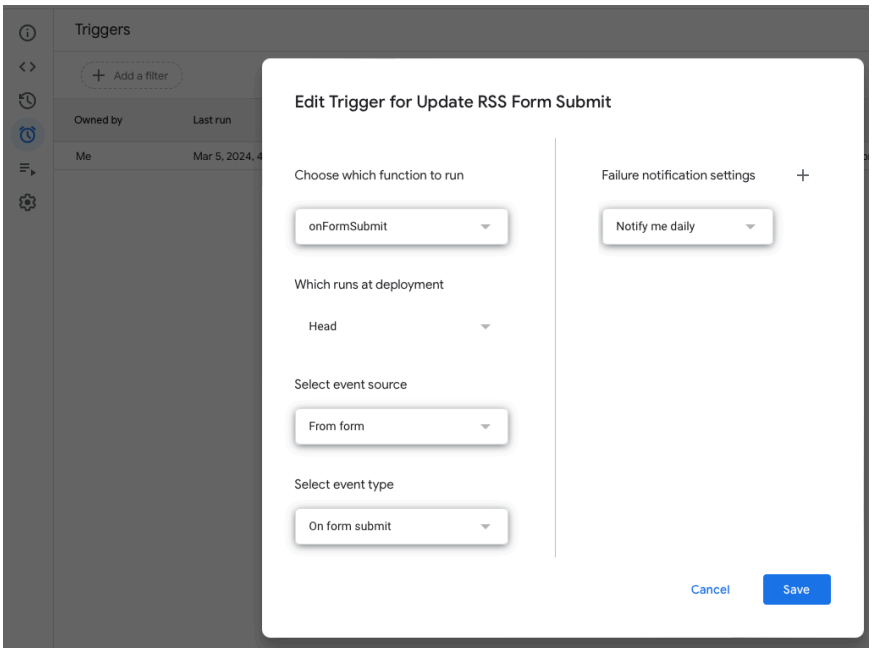
```
var categories; //post categories. may be multiple, passed as an array
<category> element
var link; //link to post on website <link> element
var enclosure; //video or audio file for use in the feed, passed as an
arrary 0 => length of file in bytes 1 => file mime type 2=> file URL.
<enclosure> element used for podcasting. probably unused
var guid; //unique identifier for the element. url to item usually.
<guid> element
var pubDate; //date that item was published to the site. RFC3339 Format
<pubDate> element
var source; //link to original URL if it was reposted from another site.
<source> element. probably unused.
var sourceUrl; //link to original URL if it was reposted from another
site. <source> element. probably unused. */
var item; //xml string for inserting into the RSS feed.
var i = 0; //init i as counter
item = " <item>\n";
item += " <title>" + title + "</title>\n";
item += " <link>" + link + "</link>\n";
item += " <description><![CDATA[" + description + "]]></description>\n";
if (author) item += " <author>" + author + "</author>\n";
if(category) { while(i < category.length) {
item += " <category>" + category[i] + "</category>\n";
i++;
} }
if(enclosure) item += " <enclosure length=\"" + enclosure[0] +"\"
type=\"" + enclosure[1] + "\" url=\"" + enclosure[2] + "\" />\n";
item += " <guid isPermaLink=\"true\">" + guid + "</guid>\n";
item += " <pubDate>" + pubDate + "</pubDate>\n";
if(source) item += " <source url="+ sourceUrl + "\">" + source +
"</source>\n";
item += " </item>\n";
```

```
return item;
}

function updateRSS(id, item) { //id: document id of Google Doc
containing rss xml; item: formatted xml item of latest post

const doc = DocumentApp.openById(id).getBody();
var text = doc.getText();

var offset = text.length - 17; //insert point is 17 characters from end
of document

text = doc.editAsText();

text.insertText(offset, item);

return;
}

/*Date Formatting functions taken from
https://github.com/whitep4nth3r/rfc-822 Thanks

MIT License Copyright (c) 2022 Salma Alam-Naylor

Permission is hereby granted,
free of charge, to any person obtaining a copy of this software and
associated
documentation files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use, copy,
modify, merge,
publish, distribute, sublicense, and/or sell copies of the Software, and
to
```

```
permit persons to whom the Software is furnished to do so, subject to
the
following conditions:

The above copyright notice and this permission notice
(including the next paragraph) shall be included in all copies or
substantial
portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF
ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO
EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN
THE SOFTWARE.
*/
function buildRFC822Date(dateString) {
const dayStrings = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"];
const monthStrings = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
"Aug", "Sep", "Oct", "Nov", "Dec"];

const timeStamp = Date.parse(dateString);
const date = new Date(timeStamp);

const day = dayStrings[date.getDay()];
const dayNumber = addLeadingZero(date.getDate());
const month = monthStrings[date.getMonth()];
```

```
const year = date.getFullYear();
const time =
`${addLeadingZero(date.getHours())}:${addLeadingZero(date.getMinutes())}
:00`;
const timezone = date.getTimezoneOffset() === 0 ? "GMT" : localTimeZone;

//Wed, 02 Oct 2002 13:00:00 GMT
return `${day}, ${dayNumber} ${month} ${year} ${time} ${timezone}`;
}
// add a leading 0 to a number if it is only one digit
function addLeadingZero(num) {
num = num.toString();
while (num.length < 2) num = "0" + num;
return num;
}
```

After pasting in the code, click the “Clock” icon and “Add a trigger”. Change “Choose which function to run” to onFormSubmit and change “Select event type to “On form submit”



Once saved you will need to give the script permission to edit files in your Google Drive and to connect to external applications. You will now be able to use the form to update your feed automatically. There is still work that can be done with this script, such as adding the functionality for updating posts by deleting an existing post if the link already exists and

replacing the content with the new submission but for now. This works good enough for me. I'll update this guide when I take on that functionality. Thank you for reading. Be sure to leave a comment on this blog post if you need some help getting this to work in your environment.