Title: M-SAT Website Administrator Guide

Author: Ian Roberts

Date: January 14, 2017

# Introduction

## Purpose

This document is to provide insight to the construction and use of the M-SAT website, which was heavily overhauled during Fall-Winter 2016. This site is meant to promote the Missouri Satellite Research Team (M-SAT) and caters its content to current and prospective sponsors (AFRL, NASA, Missouri S&T, etc) and to students wishing to get involved.

## Frontend

The site’s frontend is built on MaterializeCSS (<http://materializecss.com>), commonly referred to as “Materialize”. The design follows Google’s Material Design guidelines, which have proven to be very successful in providing users with an attractive and effective experience. The colors are largely based on Missouri S&T colors and colors found from the M-SAT logo.

## Backend

The site is built with Jekyll, a static site generator. This decision was made primarily because it would allow the site to continue to be hosted on the “tilde site” (~mrsat). Jekyll generates what is known as a “static site”. This means that there are no database queries or other server-side actions during a user’s experience. When a user visits a static site, the site is loaded primarily by the web browser, which renders the HTML, CSS, JavaScript, and the remaining content. Jekyll was found to be a good solution for the M-SAT site primarily for its low server dependencies and ease of updating.

# Materialize

## Documentation

The creators’ documentation for this framework is very well done, so please refer to it as much as possible via the link in the introduction.

## Grid System

Like most frontend frameworks when this guide was written, Materialize uses a grid system for its layout. The grid is always 12 columns wide, regardless of screen size. There are three screen size categories: small (s), medium (m), and large (l). These allow for a responsive behavior to most any device.

## Includes

For Materialize to work as expected, the following should be included on each page it’s used: jQuery, materialize.js or materialize.min.js, Google Material Icons (using Google API ideally), materialize.css or materialize.min.css, and any custom css/js files.

|  |
| --- |
| <!--Import jQuery before materialize.js-->  <script type="text/javascript" src="http://code.jquery.com/jquery-2.1.1.min.js"></script>  <script type="text/javascript" src="{{site.baseurl}}js/materialize.min.js"></script>  <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">  <link rel="stylesheet" href="{{ site.baseurl }}css/materialize.css" media="screen,projection"/>  <link rel="stylesheet" href="{{ site.baseurl }}css/custom.css" media="screen,projection"/> |

This can be included on every page by placing it in the <head> of \_layouts/default.html (assuming all other pages are derivatives of the default layout, which they most likely are).

# Jekyll Overview

## File Tree

Jekyll is very unique and can take some time to understand its use. The main thing to understand is that Jekyll uses a set of many files to produce and populate an entire static site. There are many moving parts that go into the production of any Jekyll site. The typical Jekyll project tree looks like this.

|  |
| --- |
| M-SAT Jekyll Site  |  |- \_config.yml  | |  | |- Used as a settings file for site  |  |- site pages (index.md, about.md, etc.)  | |  | |- Pages to be generated from Markdown  |  |- \_data/  | |  | |- Data files (YML, JSON, or CSV)  |  |- \_layouts/  | |  | |- Templates for various page types (HTML)  |  |- \_includes/  | |  | |- Modules that can be included throughout the site (HTML)  |  |- \_plugins/  | |  | |- Add-ons to give site more functionality  |  |- \_posts/  | |  | |- Blog posts or the like written in Markdown  |  |- \_site/  | |  | |- This is the static site created by Jekyll  |  |- css/  |- js/  |- fonts/  |- images/  |- documents/  | |- These folders are automatically added to \_site/ upon production and  | can be used in the static site.  |  |- Miscellaneous files for Jekyll |

## Generate the Site

To generate the \_site folder, the static site, simple navigate to the project folder and   
run jekyll serve.

|  |
| --- |
| cd /your/project/directory  jekyll serve |

## YAML Headers

YAML Headers are bits of information that can be added to page and \_layout files that allows you to use that information on the page. This allows for simpler editing and configuring. These headers will be a primary way of making changes to the M-SAT site. See an example YAML header below for a page outreach.md.

|  |
| --- |
| ---  # Defines which layout to use when generating the outreach page layout: outreach  # Defines what link to use when generating the outreach page  Ex: “http://example.com/outreach” permalink: /outreach/  # Passes the cover\_image variable to the outreach layout when generating the outreach page  cover\_image: 'images/space.jpg' --- |

## Liquid

Liquid is a “template engine” as described by its [GitHub repository](https://github.com/Shopify/liquid). Jekyll uses Liquid as a way to use logic such as if/else, for loops, variables, site data, and in many other ways. For example, if one would want to place a cover image on a page by accessing the YAML header variable cover\_image, it would be done like so.

|  |
| --- |
| ***# File: outreach.md***  ---  # YAML Header  layout: outreach permalink: /outreach/  cover\_image: 'images/space.jpg’  --- |

|  |
| --- |
| ***# File: \_layouts/outreach.html***  # Since outreach.md has a layout of outreach, when generating the page, it will use the layout found at \_layouts/outreach.html and the YAML header data can be accessed on the layout.  # To call the variable, we use this syntax: {{variable}}  # The variable in this case is: cover\_image  # But it was passed from the page file *outreach.md*  # So we access the variable like this: {{page.cover\_image}}  <img src=”{{page.cover\_image}}” /> |

Let’s say the cover image should only be displayed if it exists. An if statement would need to be used as shown below.

|  |
| --- |
| ***# File: \_layouts/outreach.html***  # If statements are done like this:  # {% if insert\_condition\_here %}  # ...  # {% endif %}  {% if page.cover\_image %}  <img src=”{{page.cover\_image}}” />  {% endif %} |

## Data Elements

### Basic Use

I mentioned before that Jekyll produces static sites that don’t call a database when viewed. A good alternative to small amounts of data is the \_data directory. JSON, YML, and CSV files can all be used as data elements, which can then be used throughout the site’s development. YML files have been used primarily on the M-SAT site. YML documents have a distinct file format that can be found online.

For example, if there is a file \_data/partners.yml as shown below, its data can be pulled when desired to generate the page content on the layout file \_layouts/partners.html.

|  |
| --- |
| ***# File: partners.yml***  # This YML file holds the data that we wish to use across the site.  - title: 'AFRL'  logo: 'images/partners/afrl\_Logo.jpg'  url: 'http://afrl.gov’   text: |  We love math and science.  - title: 'NASA'  logo: 'images/partners/NASA.png'  url: 'http://nasa.gov'  text: |  We do space research and produce astronauts. |

|  |
| --- |
| ***# File: \_layouts/partners.html***  # To output each piece of data from each data entry, use a for loop.  {% for partner in site.data.partners %}  {{ partner.title }}  {{ partner.logo }}  {{ partner.url }}  {{ partner.text }}  {% endfor %} |

### Creating Pages For Each Data Element

The previous example outlines how to use data elements on a single page. Many times it would be useful to create an individual page for each data entry.

Using a plugin called \_plugins/data\_page\_generator.rb (see [GitHub](https://github.com/avillafiorita/jekyll-datapage_gen)), separate pages can be created automatically. The first step in doing so is to edit the \_config.yml file by adding the following.

|  |
| --- |
| ***# File: \_config.yml***  ...  page\_gen:  - data: <<name of the data>>  template: <<name of the template to use to generate the page>>  name: <<field used to generate the filename>>  dir: <<directory in which files are to be generated>>  extension: <<extension used to generate the filename>>  - ... |

Each page created will have the individual data element’s data available for use in the layout. For example, here’s how one would create a separate page for each of the partner data elements in \_data/partners.yml.

|  |
| --- |
| ***# File: \_config.yml***  page\_gen:  - data: 'partners’  template: 'partner'  name: 'title'  dir: 'partners’  # Based on these settings, two pages will be created:   * http://mysite.com/partners/afrl.html * http://mysite.com/partners/nasa.html |

|  |
| --- |
| ***# File: \_layouts/partner.html***  # The data from the data element is now part of the page object.  # The data element’s text field would be called like so: {{ page.text }}  <html>  <head>  <title>Title Goes Here</title>  </head>  <body>  {{ page.title }}  {{ page.logo }}  {{ page.url }}  {{ page.text }}  </body>  </html> |

## Markdown

Jekyll is optimized for blogs. It utilizes Markdown to produce sites that are easy to write. Essentially, Markdown is converted to HTML upon site generation. This allows for fast development since the use of HTML can be cumbersome, especially for those that haven’t ever used it. Markdown files are stored as .md and .markdown files. The content from these files is then sent to the appropriate layout after converting the Markdown to HTML. For example, if there is an about.md file that contains our Markdown content (text, lists, tables, images, etc.) and it utilizes a layout found at \_layout/about.html, here’s how it would look.

|  |
| --- |
| ***# File: about.md***  ---  layout: about  permalink: /about/  cover\_image: 'images/space.jpg'  ---  ### Who we are  We are the Missouri University of Science and Technology (Missouri S&T) satellite research team (M-SAT). M-SAT works in conjunction with a number of AFRL/NASA/industry mentors to design, construct, and launch small satellites.  ### Our Mission  #### Primary Mission Objectives  \* Successfully track MRS SAT using stereoscopic imaging  \* Utilize R-134a-based propulsion system for satellite translational and attitude control |

|  |
| --- |
| ***# File: \_layouts/about.html***  ---  # Derive this layout from the default layout found at \_layouts/default.html  layout: default  ---  <div>  {{content}} <!--about.md content will be injected here-->  </div> |

# Making Changes

## M-SAT Projects

In this site’s files, a project or msat-project is the overall team project such as Nanosat-8, Nanosat-9, USIP, BalloonSat, etc.

### Project Data

Project data can be found in \_data/msat-projects.yml. Reference the example code below.

|  |
| --- |
| ***# File: \_data/msat-projects.yml***  - title: 'Nanosat-8'  url: 'nanosat-8'  tagline: 'Testing an imaging system to determine precise position of objects in space.'  image: 'images/cube2.jpg'  full\_description: |  The goals of the MR/MRS SAT project are to test new technologies in support of missions involving proximity operations, including the study of an R-134a-based cold gas propulsion system for use in formation flying applications and the development of a stereoscopic imager sensor used to determine the real-time relative position/velocity vectors between the satellite pair. Data obtained during the close formation flight phase of MR/MRS SAT will be evaluated for the benefit of future missions.  timeline:  - date: 'January 2016'  text: 'Important thing happened.'  - date: 'July 2016'  text: 'Working on stuff.'  - date: 'December 2016'  text: 'Awesome stuff is done!'  people:  - role: 'Principal Investigator'  name: 'Dr. Hank Pernicka'  email: 'pernicka@mst.edu'  image: 'images/profile2.jpg'  - role: 'Program Manager'  name: 'Damon Wendt'  email: 'dkwzq4@mst.edu'  image: 'images/profile3.png'  - role: 'Chief Engineer'  name: 'Pavel Galchenko'  email: 'pgwv8@mst.edu'  image: 'images/profile2.jpg' |

#### Update Text

Simply edit the text seen in the desired data field. It is important to note that the full\_description field can contain Markdown. In fact, if desired, in the respective layout use the below code to convert the Markdown to HTML.

|  |
| --- |
| ***# File: \_layouts/msat-project.html***  # markdownify is a built-in function of Jekyll  {{ data\_element.field | markdownify}}  # Be sure to set data\_element.field to whatever piece of data to you wish to convert to HTML from Markdown |

#### Update/Add Image

Add the image to desired folder within the images/ directory if needed and update the image field.

Note: On any URL or path to a file, do not precede it by a / or else the file-paths and URLs will be incorrect and will not work properly. The file-paths and URLs are often associated with {% site.baseurl %} which is set in \_config.yml.

|  |
| --- |
| # Use this:  image: ‘path/to/file.png’  # Do NOT use this:  image: ‘/path/to/file.png’ |

#### Add New Person/Timeline Event

Principal Investigator, Program Manager, and Chief Investigator are all part of the people of an msat-project. Each of these people have a role, name, and email. Each of these people is automatically added to the msat-project’s page, so to add another person, simply add on to the existing list like so. The order that the people appear on the website is the same order that they appear in the people list.

|  |
| --- |
| ***# File: \_data/msat-projects.yml***  - title: ‘Nanosat-8’  ...  people:  - role: 'Principal Investigator'  name: 'Dr. Hank Pernicka'  email: 'pernicka@mst.edu'  image: 'images/profile\_hank.jpg'  - role: 'Program Manager'  name: 'Damon Wendt'  email: 'dkwzq4@mst.edu'  image: 'images/profile\_damon.png'  - role: 'Chief Engineer'  name: 'Pavel Galchenko'  email: 'pgwv8@mst.edu'  image: 'images/profile\_pavel.jpg'  - role: ‘Web Admin'  name: 'Ian Roberts’  email: 'iprnq9@mst.edu'  image: 'images/profile\_ian.jpg' |

Similarly, timeline events can be added on to the existing events. Again, the order that they appear on the list is the order they will appear on the site to avoid any sort of confusion.

|  |
| --- |
| ***# File: \_data/msat-projects.yml***  - title: ‘Nanosat-8’  ...  timeline:  - date: 'January 2016'  text: 'Important thing happened.'  - date: 'July 2016'  text: 'Working on stuff.'  - date: 'December 2016'  text: 'Awesome stuff is done!'  - date: ‘January 2017’  text: ‘New website up and running!’ |

### Layouts

The layout files associated with the msat-project data is \_layouts/msat-project.html and \_layouts/index.html. Changes here should be made only if changes in the appearance of the template ought to be changed; if only text or images want to be changed, do so changing the data file or the YAML header if possible.

#### \_layouts/index.html

This layout derives from the default layout, and is the homepage for the entire site. It has a parallax image, a welcome note, and displays a small card for each msat-project that exists in the \_data/msat-projects.yml file.

#### \_layouts/msat-project.html

This layout derives from the default layout, and is the entire overview of the msat-project. It displays the majority of the information about that msat-project.