

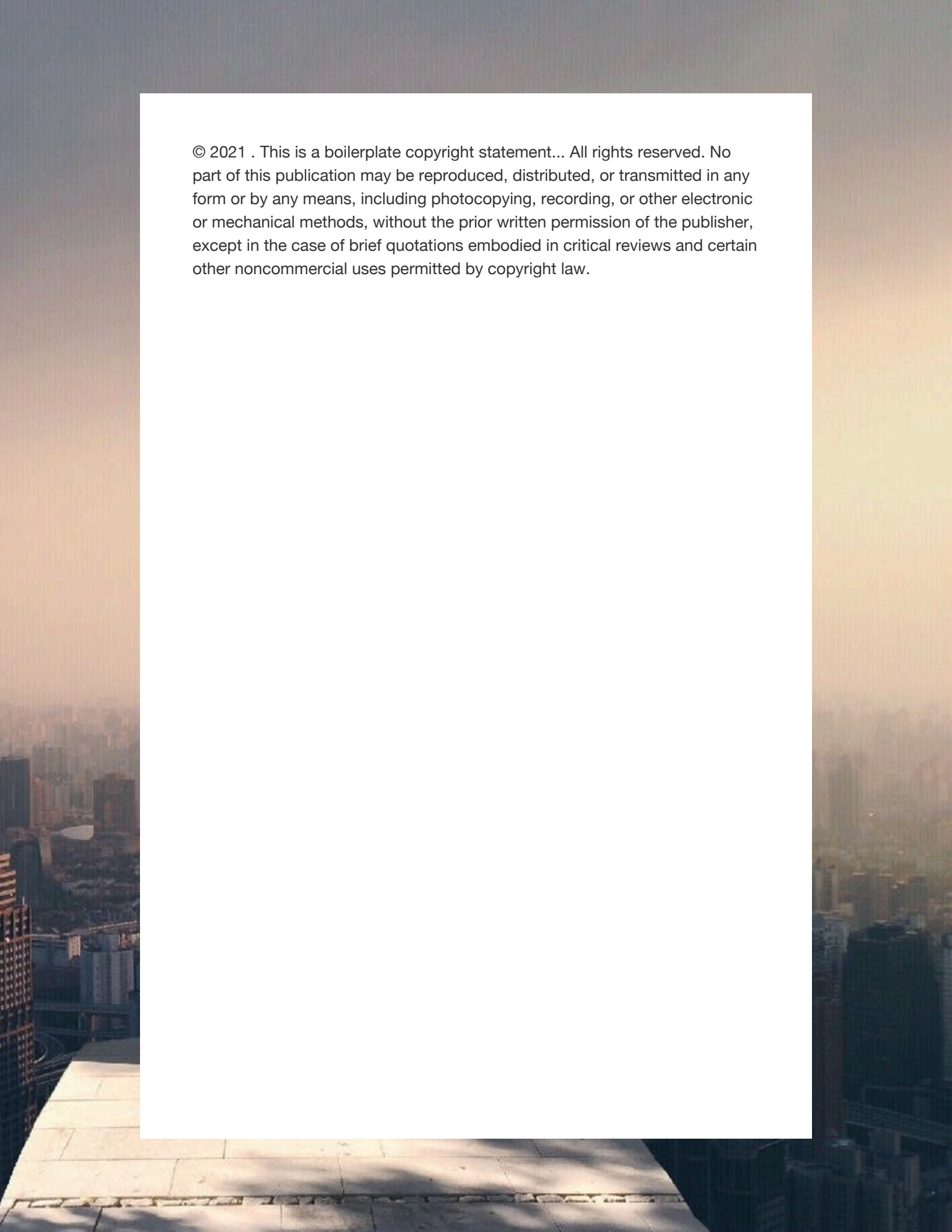


Dragan Dimitrijevic - Software Engineer

version 3.4

Last generated: August 05, 2021



A high-angle photograph of a city skyline, likely New York City, taken from a rooftop. The foreground shows a concrete ledge with a metal railing. The city is filled with numerous skyscrapers and buildings, extending to the horizon. The sky is a mix of orange, yellow, and blue, indicating the time is either early morning or late evening. A large, white rectangular area is superimposed over the center of the image, containing a copyright notice.

© 2021 . This is a boilerplate copyright statement... All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.

Table of Contents

Curriculum Vitae

Skills.....	2
Work Experiences.....	4
Projects.....	8
About Ruby, Gems, Bundler, etc.	13

Authoring

Pages.....	21
------------	----

Navigation

Sidebar navigation.....	29
-------------------------	----

Skills

Summary: This is skillset summary and its a great contnet to use in order to propote. part of the theme that remembers your current page, highlights the active item, stays in a fixed position on the page, and more. This page explains a bit about how the sidebar was put together.

The list of skills comes here

- Nodejs
- Git
- Linux
- Angular
- Scrum
- PHP
- Javascript
- Ruby
- MySQL
- Typescript
- Apache
- Qt/C++
- HTML5/CSS3 xxxx

Skills comming here. The sidebar uses the [Navgoco jQuery plugin](#) as its basis. Why not use Bootstrap? Navgoco provides a few features that I couldn't find in Bootstrap:

- Navgoco sets a cookie to remember page, the cookie allows the plugin to remember the state.
- Navgoco inserts an `active` class based on the navigation option keeping the accordion open.
- Navgoco includes the expand and collapse.

In short, the sidebar has some complex logic here. I've integrated Navgoco's features with the sidebar.html and sidebar data files to build the sidebar. It's probably the most impressive part of thierarchy is important in a documentation site.)

You can see that the `external_url` is a condition that applies a different formatting. Although this feature is available, I recommend putting any external navigation links in the top navigation bar instead of the side navigation bar.

Work Experiences

Summary: This is Work Experiences summary and its a great content to use in order to propote. part of the theme that remembers your current page, highlights the active item, stays in a fixed position on the page, and more.

Work Experiences

- ☒ @mentions, #refs, [links \(page 4\)](#), **formatting**, and ~~tags~~ supported
- ☒ list syntax required (any unordered or ordered list supported)
- ☒ this is a complete item
- ☐ this is an incomplete item

Software Architect / Head of product

Workplace: Nordlid

Jan 2018 - Nov 2019

Role

Developing application for Eloqua Maintaining and developing custom solutions

Head of Products

Developing application for Eloqua Maintaining and developing custom solutions

CTO

Workplace: Globase International (Mailup Group)

Jul 2016 - Dec 2017

Project and challenges

One of the biggest and most challenging projects during my time at Globase was the migration of the 2 cabinets from one hosting location to another at QSC. This meant that most of the underlying services which powered platforms had to be restarted and reconfigured, from database servers, load balancers, web servers, network switches, etc.

Roles

- Managing development team (Agile SCRUM)
- Maintaining in house platforms and custom solutions
- Involved in pre-sales and requirement specification
- Maintaining 6 hardware cabinets in QSC Germany
- Part of the management team

Another big undertaking was the attempt to migrate V2 platform from QSC to a new cloud setup at Mailup hosting center in Cremona. Even though we could not complete the migration due to the high level of complexity and unforeseeable issues, we were able to implement continuous deployment on the existing setup, since the V2 platform was fully virtualized in a development environment.

System Developer

Workplace: Increase

Nov 2015 - Jun 2016

Most of my time at Increase I spent working with Oracles's marketing automation platform Eloqua. I was building apps and data integrations in the AWS cloud. One of the more interesting project during that time, was the development of the Event App. This application was designed to handle the event flow by creating many assets (Email, Landing Page, Registration Form, Custom Data Object etc) on a campaign canvas in Eloqua, which can handle the invitation, registration and keep track of the participation status.

System Developer

Workplace: Globase

Apr 2014 - Oct 2015

Similar to my previous job, at Globase I was developing solutions and integrations against yet another email marketing automation platform. This involved a variety of tasks such as integrating external data sources for email content, implementing contest strategies such as polls, quiz and giveaways.

Software Developer

Workplace: Brandhouse

Mar 2013 - Mar 2014

At Brandhouse, I had a chance of working on a platform (Publicator) made for graphic design and print. The tasks were including integration with data sources, implementing new feature requests and collecting requirement specification from a customer.

Developer

Workplace: Peytz & Co

Sep 2011 - Feb 2013

This was my first job in the Marketing industry where I had an opportunity to work with an email automation platform. I developed new features and custom solutions which would allow users to integrate content from external sources such as SOAP API, REST API, XML Feeds etc.

Developer

Workplace: MOCH

Mar 2007 - Aug 2011

During my time a MOCH, I worked on a modular SCORM compliant LMS (Learning Management System) platform. My tasks were development of modules (LMS Courses, Workflow, Questionnaire), adding features and fixing issues.

Developer

Workplace: Tang Data A/S

Jul 2006 - Feb 2007

My first work experience was developing on an inhouse ERP platform designed for veterinarians. I worked primarily on migrating from an old version, adding new features and troubleshooting reported problems.

Projects

Summary: My most interesting projects for the past 10 years

Projects

- ☒ #refs, [links \(page 8\)](#), **formatting**, and tags supported
- ☒ list syntax required (any unordered or ordered list supported)
- ☒ this is a **complete** item
- ☐ this is an *incomplete* item

Airship Eloqua Integration

This integration project was about making Eloqua send push messages using [Airship](#). The integration was covering the following use cases:

- Send push message from campaign canvas (including scheduled and in-app)
- Send push message from a segment (batch)
- Send push message on form post
- Import devices to Eloqua
- Collect activity response

Send push message from Campagin Canvas (scheduled, in-app)

Send push message from Segment (batch)

Send push on form submit

Import devices to Eloqua

Response activity (send / open)

This project was about Integration push messages within.

Techstack

Angular, Nodejs, serverless, AWS, Lambda, RDS, S3, VPC, CloudFront

Få tiden tilbage

It was the one of the more interesting projects that took almost 6 month to complete. The client was a public transpaoration copmany (DSB) and the concept was focusing on bringing back value to the passangeres who experince dealays, due to the maintenance work done on the railways. The first release of the application was dealing with the journeys, where a passenger was to able to check in on the begining of the journey, and checkout on the destination, earning a certain amount of minutes. Every journey route had a defined award, and every journey was adding to the total of the time saved.

The second release of the application was dealing with the in-app ticket purchase based on the time (points) saved.

You can read more about this project for which Norlid won the Danish Digital Award in the category [“Digital Activation”](#) (DSB)

Techstack

Ionic, Angular, Nodejs, serverless, AWS, Lambda, RDS, S3, VPC, CloudFront

Rich Relevance Eloqua Integration

This project was about Integration push messages within. post will show you how the content will look like in the post pages and how the headlines, quotes and quotes will be represented. Jekyll is mainly used to write simple markdown and after that it renders out a static pages, so you need to know the basics of writing markdown for that.

For more information about writing markdown you can checkout the following markdown cheatsheets:

- [Mastering Markdown](#)
- [Markdown Guide](#)
- [GitHub Flavored Markdown Spec](#)

This is the h1 text

This is the h2 text

Bold Text in the post will look like:

This text is Bold

Italic Text in the post will look like:

This text is Italic

Quotes on your post will look like this

Codes on your post will look like this

Link in the post will look like:

[This is a link \(page 8\)](#)

Bullet list in the post will look like:

- Item 1
- Item 2

Techstack

Angular, Nodejs, serverless, AWS, Lambda, RDS, S3, VPC, CloudFront

Event App

The main objective of this project was to create an application which would help facilitate event management in Eloqua. In many cases managing events can be a repetitive task, requiring many hours of work in order to setup the necessary logic for an event flow. Tracking changes in participation status. and handling waiting list for This application attempts to do exactly that, save the marketer from manual work and create the pattern for the participation flow.

Event Assets

During the event creation process, many Eloqua assets are being created and integrated into the campaign canvas. Assets such as Segment, Email, Form, Landing Page, Custom Data Objects and Campaign are created and configured in order to support invitation, registration, unregistration and waiting list email notifications. When a Segment is ready and the campaign activated, the Contacts on that segment would receive an invitation email and by clicking on the registration link, enter the flow.

Participation Flow

A participant can have different status changes such as registered, attended, unregistered and waiting. All these status changes can be initiated within the application and stored in the Custom Data Object in Eloqua.

Participation flow via Processing steps (registered, attended, waiting)

Asset Templates (pre defined templates for unified look and feel)

- Participation flow (invitation, registration, unregistration, waiting list)
- Asset management such as registration forms, processing steps, landing pages, emails and custom data object connecting all assets in a campaign canvas.

Application can handle participant attendance, different flows (invitation, registration, unregistration) and waiting list for booked events.

Techstack

Sails, Nodejs, jQuery, Docker, AWS, ECS, RDS, VPC, CloudFront

About Ruby, Gems, Bundler, and other prerequisites

Summary: Ruby is a programming language you must have on your computer in order to build Jekyll locally. Ruby has various gems (or plugins) that provide various functionality. Each Jekyll project usually requires certain gems.

About Ruby

Jekyll runs on Ruby, a programming language. You have to have Ruby on your computer in order to run Ruby-based programs like Jekyll. Ruby is installed on the Mac by default, but you must add it to Windows.

About Ruby Gems

Ruby has a number of plugins referred to as “gems.” Just because you have Ruby doesn’t mean you have all the necessary Ruby gems that your program needs to run. Gems provide additional functionality for Ruby programs. There are thousands of [Rubygems](#) available for you to use.

Some gems depend on other gems for functionality. For example, the Jekyll gem might depend on 20 other gems that must also be installed.

Each gem has a version associated with it, and not all gem versions are compatible with each other.

Rubygem package managers

[Bundler](#) is a gem package manager for Ruby, which means it goes out and gets all the gems you need for your Ruby programs. If you tell Bundler you need the [jekyll gem](#), it will retrieve all the dependencies on the jekyll gem as well – automatically.

Not only does Bundler retrieve the right gem dependencies, but it’s smart enough to retrieve the right versions of each gem. For example, if you get the [github-pages](#) gem, it will retrieve all of these other gems:

```
github-pages-health-check = 1.1.0
jekyll = 3.0.3
jekyll-coffeescript = 1.0.1
jekyll-feed = 0.4.0
jekyll-gist = 1.4.0
jekyll-github-metadata = 1.9.0
jekyll-mentions = 1.1.2
jekyll-paginate = 1.1.0
jekyll-redirect-from = 0.10.0
jekyll-sass-converter = 1.3.0
jekyll-seo-tag = 1.3.2
jekyll-sitemap = 0.10.0
jekyll-textile-converter = 0.1.0
jemoji = 0.6.2
kramdown = 1.10.0
liquid = 3.0.6
mercenary ~> 0.3
rdiscout = 2.1.8
redcarpet = 3.3.3
RedCloth = 4.2.9
rouge = 1.10.1
terminal-table ~> 1.
```

See how Bundler retrieved version 3.0.3 of the jekyll gem, even though (as of this writing) the latest version of the jekyll gem is 3.1.2? That's because github-pages is only compatible up to jekyll 3.0.3. Bundler handles all of this dependency and version compatibility for you.

Trying to keep track of which gems and versions are appropriate for your project can be a nightmare. This is the problem Bundler solves. As explained on [Bundler.io](https://bundler.io):

Bundler provides a consistent environment for Ruby projects by tracking and installing the exact gems and versions that are needed.

Bundler is an exit from dependency hell, and ensures that the gems you need are present in development, staging, and production. Starting work on a project is as simple as `bundle install`.

Gemfiles

Bundler looks in a project's "Gemfile" (no file extension) to see which gems are required by the project. The Gemfile lists the source and then any gems, like this:

```
source "https://rubygems.org"

gem 'github-pages'
gem 'jekyll'
```

The source indicates the site where Bundler will retrieve the gems:

<https://rubygems.org> .

The gems it retrieves are listed separately on each line.

Here no versions are specified. Sometimes gemfiles will specify the versions like this:

```
gem 'kramdown', '1.0'
```

This means Bundler should get version 1.0 of the kramdown gem.

To specify a subset of versions, the Gemfile looks like this:

```
gem 'jekyll', '~> 2.3'
```

The `~>` sign means greater than or equal to the *last digit before the last period in the number*.

Here it will get any gem equal to 2.3 but less than 3.0.

If it adds another digit, the scope is affected:

```
gem `jekyll`, `~>2.3.1`
```

This means to get any gem equal to 2.3.1 but less than 2.4.

If it looks like this:

```
gem 'jekyll', '~> 3.0', '>= 3.0.3'
```

This will get any Jekyll gem between versions 3.0 and up to 3.0.3.

See this [Stack Overflow post](#) for more details.

Gemfile.lock

After Bundler retrieves and installs the gems, it makes a detailed list of all the gems and versions it has installed for your project. The snapshot of all gems + versions installed is stored in your Gemfile.lock file, which might look like this:

GEM

```
remote: https://rubygems.org/
specs:
  RedCloth (4.2.9)
  activesupport (4.2.5.1)
    i18n (~> 0.7)
    json (~> 1.7, >= 1.7.7)
    minitest (~> 5.1)
    thread_safe (~> 0.3, >= 0.3.4)
    tzinfo (~> 1.1)
  addressable (2.3.8)
  coffee-script (2.4.1)
    coffee-script-source
    execjs
  coffee-script-source (1.10.0)
  colorator (0.1)
  ethon (0.8.1)
    ffi (>= 1.3.0)
  execjs (2.6.0)
  faraday (0.9.2)
    multipart-post (>= 1.2, < 3)
  ffi (1.9.10)
  gemoji (2.1.0)
  github-pages (52)
    RedCloth (= 4.2.9)
    github-pages-health-check (= 1.0.1)
    jekyll (= 3.0.3)
    jekyll-coffeescript (= 1.0.1)
    jekyll-feed (= 0.4.0)
    jekyll-gist (= 1.4.0)
    jekyll-mentions (= 1.0.1)
    jekyll-paginate (= 1.1.0)
    jekyll-redirect-from (= 0.9.1)
    jekyll-sass-converter (= 1.3.0)
    jekyll-seo-tag (= 1.3.1)
    jekyll-sitemap (= 0.10.0)
    jekyll-textile-converter (= 0.1.0)
    jemoji (= 0.5.1)
    kramdown (= 1.9.0)
    liquid (= 3.0.6)
    mercenary (~> 0.3)
    rdiscount (= 2.1.8)
    redcarpet (= 3.3.3)
    rouge (= 1.10.1)
    terminal-table (~> 1.4)
```

```
github-pages-health-check (1.0.1)
  addressable (~> 2.3)
  net-dns (~> 0.8)
  octokit (~> 4.0)
  public_suffix (~> 1.4)
  typhoeus (~> 0.7)
html-pipeline (2.3.0)
  activesupport (>= 2, < 5)
  nokogiri (>= 1.4)
i18n (0.7.0)
jekyll (3.0.3)
  colorator (~> 0.1)
  jekyll-sass-converter (~> 1.0)
  jekyll-watch (~> 1.1)
  kramdown (~> 1.3)
  liquid (~> 3.0)
  mercenary (~> 0.3.3)
  rouge (~> 1.7)
  safe_yaml (~> 1.0)
jekyll-coffeescript (1.0.1)
  coffee-script (~> 2.2)
jekyll-feed (0.4.0)
jekyll-gist (1.4.0)
  octokit (~> 4.2)
jekyll-mentions (1.0.1)
  html-pipeline (~> 2.3)
  jekyll (~> 3.0)
jekyll-paginate (1.1.0)
jekyll-redirect-from (0.9.1)
  jekyll (>= 2.0)
jekyll-sass-converter (1.3.0)
  sass (~> 3.2)
jekyll-seo-tag (1.3.1)
  jekyll (~> 3.0)
jekyll-sitemap (0.10.0)
jekyll-textile-converter (0.1.0)
  RedCloth (~> 4.0)
jekyll-watch (1.3.1)
  listen (~> 3.0)
jemoji (0.5.1)
  gemoji (~> 2.0)
  html-pipeline (~> 2.2)
  jekyll (>= 2.0)
json (1.8.3)
kramdown (1.9.0)
```

```
liquid (3.0.6)
listen (3.0.6)
  rb-fsevent (>= 0.9.3)
  rb-inotify (>= 0.9.7)
mercenary (0.3.5)
mini_portile2 (2.0.0)
minitest (5.8.4)
multipart-post (2.0.0)
net-dns (0.8.0)
nokogiri (1.6.7.2)
  mini_portile2 (~> 2.0.0.rc2)
octokit (4.2.0)
  sawyer (~> 0.6.0, >= 0.5.3)
public_suffix (1.5.3)
rb-fsevent (0.9.7)
rb-inotify (0.9.7)
  ffi (>= 0.5.0)
rdiscount (2.1.8)
redcarpet (3.3.3)
rouge (1.10.1)
safe_yaml (1.0.4)
sass (3.4.21)
sawyer (0.6.0)
  addressable (~> 2.3.5)
  faraday (~> 0.8, < 0.10)
terminal-table (1.5.2)
thread_safe (0.3.5)
typhoeus (0.8.0)
  ethon (>= 0.8.0)
tzinfo (1.2.2)
  thread_safe (~> 0.1)
```

PLATFORMS

ruby

DEPENDENCIES

github-pages
jekyll

BUNDLED WITH

1.11.2

You can always delete the Gemlock file and run Bundle install again to get the latest versions. You can also run `bundle update`, which will ignore the Gemlock file to get the latest versions of each gem.

To learn more about Bundler, see [Bundler's Purpose and Rationale](#) .

Pages

Summary: This theme primarily uses pages. You need to make sure your pages have the appropriate frontmatter. One frontmatter tag your users might find helpful is the summary tag. This functions similar in purpose to the shortdesc element in DITA.

Where to author content

Use a text editor such as Sublime Text, WebStorm, IntelliJ, Visual Studio Code or Atom to create pages. Atom is recommended because it's created by Github, which is driving some of the Jekyll development through Github Pages.

Where to save pages

You can store your pages in any folder structures you want, with any level of folder nesting. The site output will pull all of those pages out of their folders and put them into the root directory. Check out the `_site` folder, which is where Jekyll is generated, to see the difference between your project's structure and the resulting site output.

The listing of all pages in the root directory (which happens when you add a permalink property to the pages) is what allows the relative linking and offline viewing of the site to work.

Frontmatter

Make sure each page has frontmatter at the top like this:

```
---
title: Alerts
tags: [formatting]
keywords: notes, tips, cautions, warnings, admonitions
last_updated: July 3, 2016
summary: "You can insert notes, tips, warnings, and important alerts in your content."
sidebar: mydoc_sidebar
permalink: mydoc_alerts.html
---
```

Frontmatter is always formatted with three hyphens at the top and bottom. Your frontmatter must have a `title` and `permalink` value. All the other values are optional.

Note that you cannot use variables in frontmatter.

The following table describes each of the frontmatter that you can use with this theme:

Frontmatter	Required?	Description
title	Required	The title for the page
tags	Optional	Tags for the page. Make all tags single words, with underscores if needed (rather than spaces). Separate them with commas. Enclose the whole list within brackets. Also, note that tags must be added to <code>_data/tags_doc.yml</code> to be allowed entrance into the page. This prevents tags from becoming somewhat random and unstructured. You must create a tag page for each one of your tags following the pattern shown in the tags folder. (Tag pages aren't automatically created.)
keywords	Optional	Synonyms and other keywords for the page. This information gets stuffed into the page's metadata to increase SEO. The user won't see the keywords, but if you search for one of the keywords, it will be picked up by the search engine.

Frontmatter	Required?	Description
last_updated	Optional	The date the page was last updated. This information could be helpful for readers trying to evaluate how current and authoritative information is. If included, the last_updated date appears in the footer of the page in small font.
sidebar	Required	Refers to the sidebar data file for this page. Don't include the ".yml" file extension for the sidebar — just provide the file name. If no sidebar is specified, this value will inherit the default property set in your _config.yml file for the page's frontmatter.
summary	Optional	A 1-2 word sentence summarizing the content on the page. This gets formatted into the summary section in the page layout. Adding summaries is a key way to make your content more scannable by users (check out Jakob Nielsen's site for a great example of page summaries.) The only drawback with summaries is that you can't use variables in them.
permalink	Required	The permalink <i>must</i> match the filename in order for automated links to work. Additionally, you must include the ".html" in the filename. Do not put forward slashes around the permalink (this makes Jekyll put the file inside a folder in the output). When Jekyll builds the site, it will put the page into the root directory rather than leaving it in a sub-directory or putting it inside a folder and naming the file index.html. Having all files flattened in the root directory is essential for relative linking to work and for all paths to JS and CSS files to be valid.

Frontmatter	Required?	Description
datatable	Optional	'true'. If you add <code>datatable: true</code> in the frontmatter, scripts for the jQuery Datatables plugin get included on the page. You can see the scripts that conditionally appear by looking in the <code>_layouts/default.html</code> page.
toc	Optional	If you specify <code>toc: false</code> in the frontmatter, the page won't have the table of contents that appears below the title. The toc refers to the list of jump links below the page title, not the sidebar navigation. You probably want to hide the TOC on the homepage and product landing pages.

Colons in page titles

If you want to use a colon in your page title, you must enclose the title's value in quotation marks.

Page names and excluding files from outputs

By default, everything in your project is included in the output. You can exclude all files that don't belong to that project by specifying the file name, the folder name, or by using wildcards in your configuration file:

```
exclude:  
  
- filename.md  
- subfolder_name/  
- mydoc_*  
- gitignore
```

Wildcards will exclude every match after the `*`.

Saving pages as drafts

If you add `published: false` in the frontmatter, your page won't be published. You can also move draft pages into the `_drafts` folder to exclude them from the build. With posts, you can also keep them as drafts by omitting the date in the title.

Markdown or HTML format

Pages can be either Markdown or HTML format (specified through either an `.md` or `.html` file extension).

If you use Markdown, you can also include HTML formatting where needed. But if your format is HTML, you must add a `markdown="1"` attribute to the element in order to use Markdown inside that HTML element:

```
<div markdown="1">This is a [link](http://example.com).</div>
```

For your Markdown files, note that a space or two indent will set text off as code or blocks, so avoid spacing indents unless intentional.

If you have a lot of HTML, as long as the top and bottom tags of the HTML are flush left in a Markdown file, all the tags inside those bookend HTML tags will render as HTML, regardless of their indentation. (This can be especially useful for tables.)

Page names

I recommend prefixing your page names with the product, such as `"mydoc_pages"` instead of just `"pages."` This way if you have other products that also have topics with generic names such as `"pages,"` there won't be naming conflicts.

Additionally, consider adding the product name in parentheses after the title, such as `"Pages (Mydoc)"` so that users can clearly navigate different topics for each product.

Kramdown Markdown

Kramdown is the Markdown flavor used in the theme but you are free to move to [CommonMark](#) or [Redcarpet](#). This mostly aligns with Github-flavored Markdown, but with some differences in the indentation allowed within lists. Basically, Kramdown requires you to line up the indent between list items with the first starting character after the space in your list item numbering. See this [blog post on Kramdown and Rouge](#) for more details.

You can use standard Multimarkdown syntax for tables. You can also use fenced code blocks with lexers specifying the type of code. The configuration file shows the Markdown processor and extension:

```
highlighter: rouge
markdown: kramdown
kramdown:
  input: GFM
  auto_ids: true
  hard_wrap: false
  syntax_highlighter: rouge
```

Automatic mini-TOCs

By default, a TOC appears at the top of your pages and posts. If you don't want the TOC to appear for a specific page, such as for a landing page or other homepage, add `toc: false` in the frontmatter of the page.

The mini-TOC requires you to use the `##` Markdown syntax for headings. If you use `<h2>` elements, you must add an ID attribute for the heading element in order for it to appear in the mini-TOC (for example, `<h2 id="mysampleid">Heading</h2>`).

Headings

Use pound signs before the heading title to designate the level. Note that kramdown requires headings to have one space before and after the heading. Without this space above and below, the heading won't render into HTML.

```
## Second-level heading
```

Result:

Second-level heading

```
### Third-level heading
```

Result:

Third-level heading

```
#### Fourth-level heading
```

Result:

Fourth-level heading

Headings with ID Tags

If you want to use a specific ID tag with your heading, add it like this:

```
## Headings with ID Tags {#someIdTag}
```

Then you can reference it with a link like this on the same page:

```
[Some link] (#someIdTag)
```

Result:

[Some link \(page 27\)](#)

For details about linking to headings on different pages, see [Automated links to headings on pages][mydoc_hyperlinks.html#bookmarklinks].

Specify a particular page layout

The configuration file sets the default layout for pages as the “page” layout.

You can create other layouts inside the layouts folder. If you create a new layout, you can specify that your page use your new layout by adding `layout: mylayout.html` in the page’s frontmatter. Whatever layout you specify in the frontmatter of a page will override the layout default set in the configuration file.

Comments

Disqus, a commenting system, is integrated into the theme. In the configuration file, specify the Disqus code for the universal code, and Disqus will appear. If you don’t add a Disqus value, the Disqus form isn’t included.

Sidebar Navigation

Summary: The sidebar navigation uses a jQuery component called Navgoco. The sidebar is a somewhat complex part of the theme that remembers your current page, highlights the active item, stays in a fixed position on the page, and more. This page explains a bit about how the sidebar was put together.

Navgoco foundation

The sidebar uses the [Navgoco jQuery plugin](#) as its basis. Why not use Bootstrap? Navgoco provides a few features that I couldn't find in Bootstrap:

- Navgoco sets a cookie to remember the user's position in the sidebar. If you refresh the page, the cookie allows the plugin to remember the state.
- Navgoco inserts an `active` class based on the navigation option that's open. This is essential for keeping the accordion open.
- Navgoco includes the expand and collapse features of a sidebar.

In short, the sidebar has some complex logic here. I've integrated Navgoco's features with the sidebar.html and sidebar data files to build the sidebar. It's probably the most impressive part of this theme. (Other themes usually aren't focused on creating hierarchies of pages, but this kind of hierarchy is important in a documentation site.)

Accordion sidebar feature

The sidebar.html file (inside the `_includes` folder) contains the `.navgoco` method called on the `#mysidebar` element.

There are some options to set within the `.navgoco` method. The only noteworthy option is `accordion`. This option makes it so when you expand a section, the other sections collapse. It's a way of keeping your navigation controls condensed.

The value for `accordion` is a Boolean (`true` or `false`). By default, the `accordion` option is set as `true`. If you don't want the accordion, set it to `false`. Note that there's also a block of code near the bottom of sidebar.html that is commented out. Uncomment out that section to have the Collapse all and Expand All buttons appear.

There's a danger with setting the accordion to `false`. If you click Expand All and the sidebar expands beyond the dimensions of the browser, users will be stuck. When that happens, it's hard to collapse it. As a best practice, leave the sidebar's accordion option set to `true`.

Fixed position sidebar

The sidebar has one other feature — this one from Bootstrap. If the user's viewport is tall enough, the sidebar remains fixed on the page. This allows the user to scroll down the page and still keep the sidebar in view.

In the `customscripts.js` file in the `js` folder, there's a function that adds an `affix` class if the height of the browser window is greater than 800 pixels. If the browser's height is less than 800 pixels, the `nav affix` class does not get inserted. As a result, the sidebar can slide up and down as the user scrolls up and down the page.

Depending on your content, you may need to adjust `800` pixel number. If your sidebar is so long that having it in a fixed position makes it so the bottom of the sidebar gets cut off, increase the `800` pixel number here to a higher number.

Opening sidebar links into external pages

In the attributes for each sidebar item, if you use `external_url` instead of `url`, the theme will insert the link into an `a href` element that opens in a blank target.

For example, the `sidebar.html` file contains the following code:

```
{% if folderitem.external_url %}
  <li><a href="{{folderitem.external_url}}" target="_blank"
  rel="noopener">{{folderitem.title}}</a></li>
```

You can see that the `external_url` is a condition that applies a different formatting. Although this feature is available, I recommend putting any external navigation links in the top navigation bar instead of the side navigation bar.

Sidebar item highlighting

The `sidebar.html` file inserts an `active` class into the sidebar element when the `url` attribute in the sidebar data file matches the page URL.

For example, the sidebar.html file contains the following code:

```
{% elsif page.url == folderitem.url %}
  <li class="active"><a href="{{folderitem.url | remove:
"/"}}">{{folderitem.title}}</a></li>
```

If the `page.url` matches the `subfolderitem.url`, then an `active` class gets applied. If not, the `active` class does not get applied.

The `page.url` in Jekyll is a site-wide variable. If you insert `{{page.url}}` on a page, it will render as follows: `/mydoc_sidebar_navigation.html`. The `url` attribute in the sidebar item must match the page URL in order to get the `active` class applied.

This is why the `url` value in the sidebar data file looks something like this:

```
- title: Understanding how the sidebar works
  permalink: mydoc_understand_sidebar.html
  output: web, pdf
```

Note that the url does not include the project folder where the file is stored. This is because the site uses permalinks, which pulls the topics out of subfolders and places them into the root directory when the site builds.

Now the `page.url` and the `item.url` can match and the `active` class can get applied. With the `active` class applied, the sidebar section remains open.