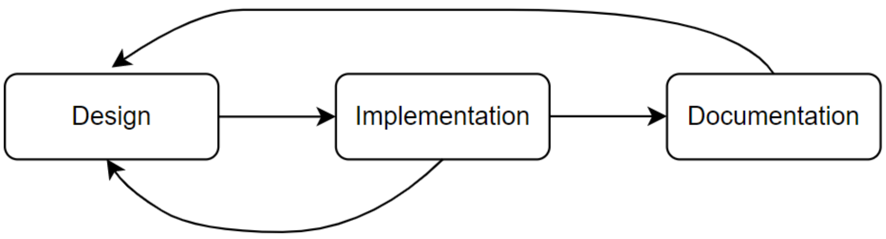
Github blog Project

Date: 2021/5/3 ~ 2021/5/10

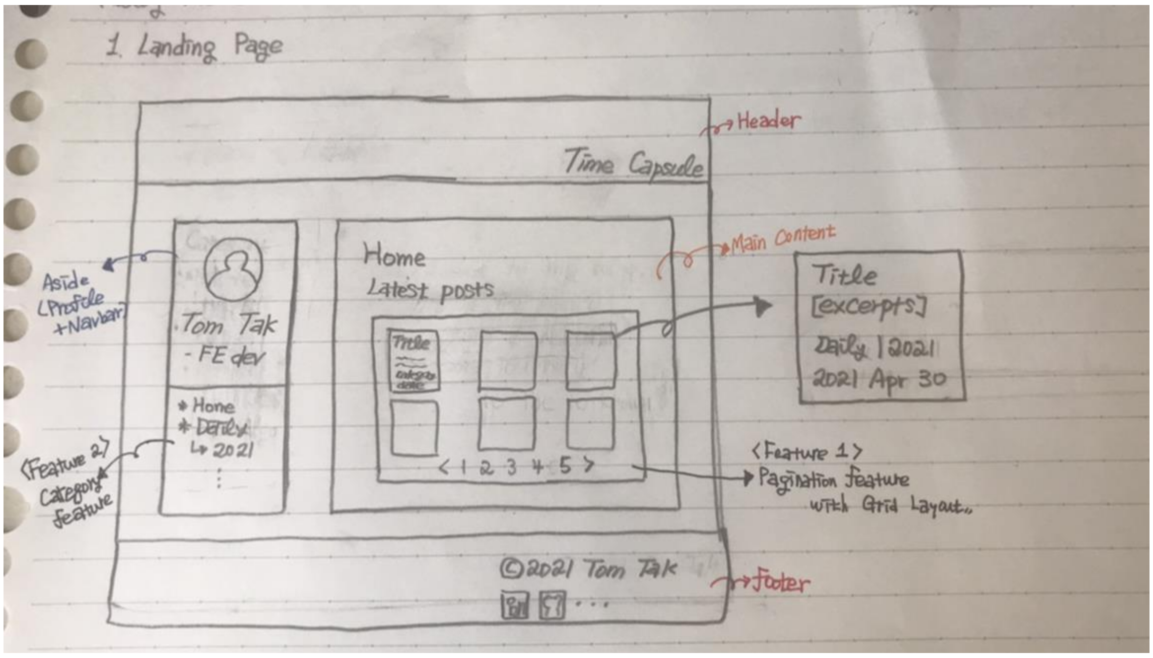
Basic workflow

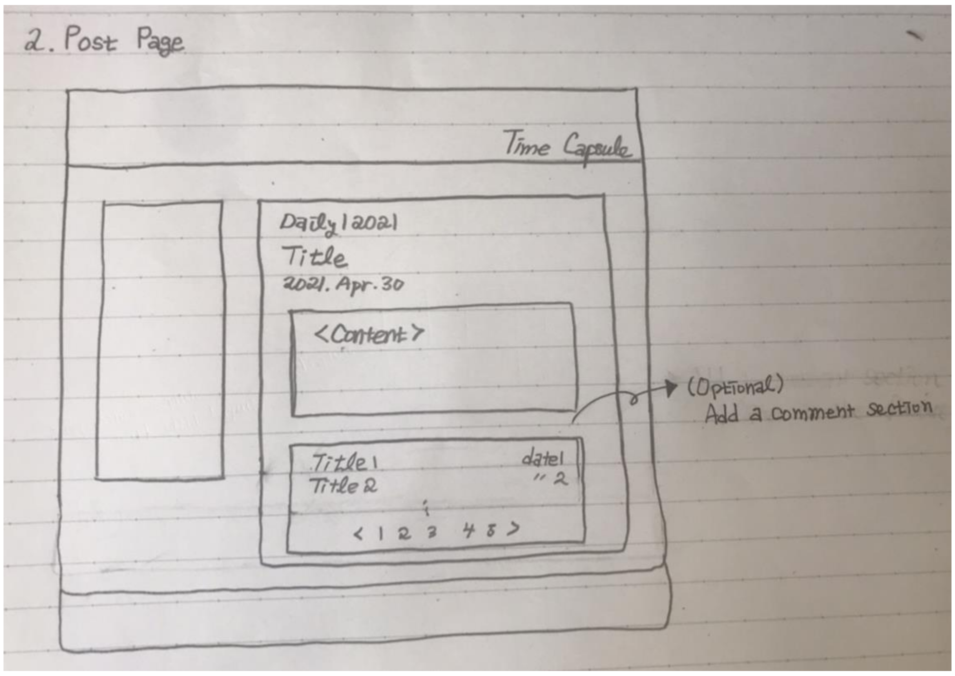


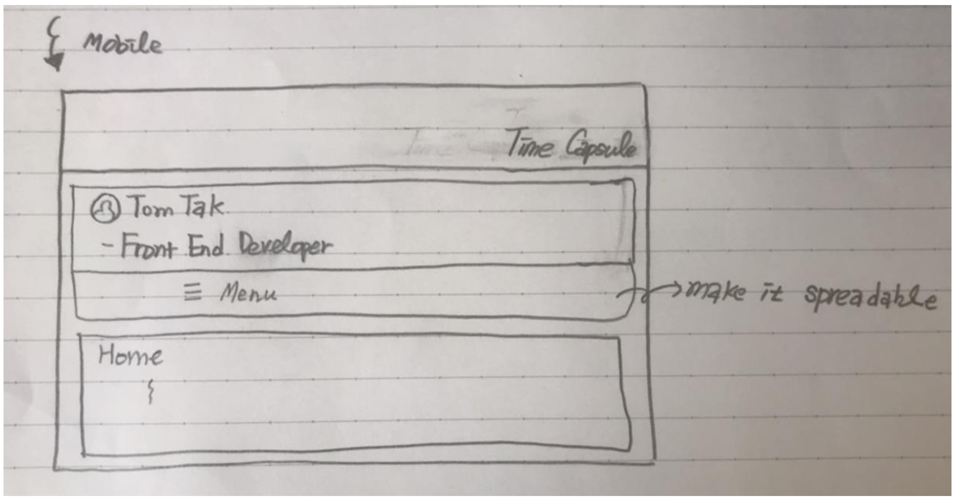
<figure 1. Basic workflow chart>

Briefly explaining the workflow I followed in this project, the basic workflow consists of 3 phases as <figure 1> describes. Firstly, I wireframed a blog design before the implementation phase. In the design phase, I considered not only the looks but also features of a blog. Once I finished the design phase, I started building a blog by actually implementing the design. Also, I often documented how I implemented some features or design of a blog along the way for the record purpose.

1. Design phase







I decided to have a blog with multi-categories and pagination features as the 3 images show.

2. Implementation phase

I was optimistic about building the blog website but sooner or later I found that I was kinda naïve. I struggled a lot at many points to implement some features or understand some concepts required to build a github blog. So, I will try to elaborate at which point I struggled along the way and how I got through it.

2-1. Creating a git repo for a gitblog and deploying it on the web.

This part was quite easy. All I needed to do was simply follow along some steps in the guide on the github pages website (https://pages.github.com/).

2-2. Use Jekyll for the blog features of github page.

With 2-1, I could only put the pages in my git repo out to the web, which is not enough for building a fully-fledged blog website. Of course, I can just manually build it up by writing hundreds and thousands repetitive html documents on the git repo, but it is not efficient and what I want. I wanted to do this in a more systemic and efficient way, so I can minimize the parts I have to do manually.

And the answer for that was to use Jekyll. Github pages actually uses Jekyll for blog website features. And again, github pages Docs (https://docs.github.com/en/pages) was quite helpful to figure out how Jekyll works with github pages for the blog features. And it was actually not that hard to build a blog website with Jekyll Theme. In addition to the github pages docs, there were lot of resources to refer to for building a blog website with Jekyll Themes like short tutorial videos on youtube.

However, there were simply not many themes to use and most of them I didn't feel like to use. I wanted to use my own design for my blog. On top of that, I soon realize that even the things I can customize with the theme is very much limited especially if I don't understand how Jekyll exactly works.

So what? I decided to go down the rabbit hole not knowing how deep it will be.

2-3. Configure an environment to use Jekyll (rvm, ruby etc..)

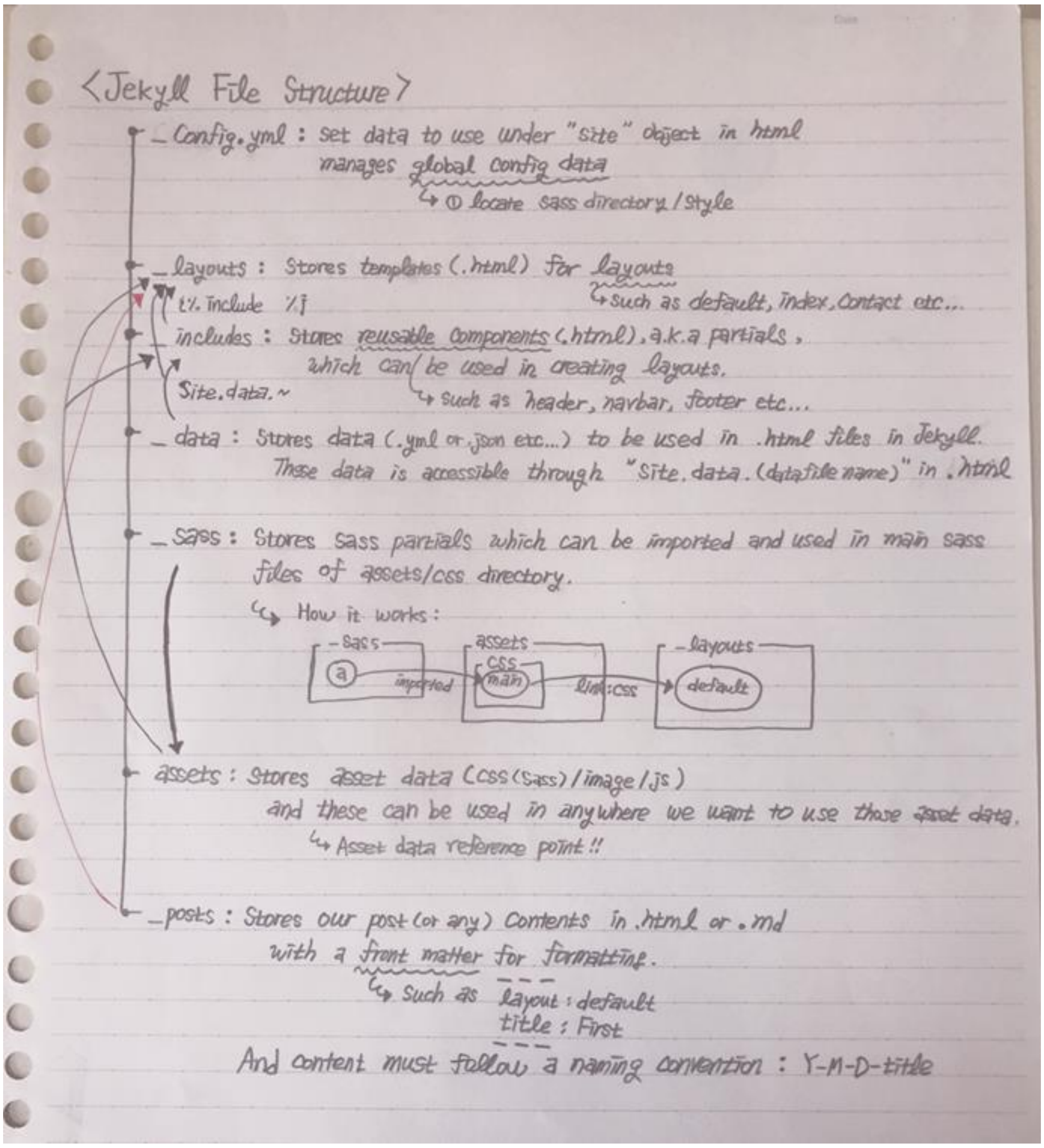
Even before start learning the Jekyll, I was already struggling with configuring the setup env for Jekyll. Jekyll runs on top of Ruby. But, installing Ruby on WSL was not easy at all to me as a novice. It was not only me who struggles to install ruby on their computer, so there were many resources out there about how to install ruby on windows or WSL to be specific. And there were also many incorrect(?) or inefficient ways to install it. And I found the right way simply by a trial and error sadly.

So, what I learnt after all is that the best way to install Ruby is via rvm, Ruby Version control Manager I guess(?). And the following website was the super helpful. (http://bigmatch.i-um.net/2013/12/04/%EB%A9%98%EB%B6%95%EC%97%86%EC%9D%B4-rvm%EA%B3%BC-%EB%A3%A8%EB%B9%84-%EC%84%A4%EC%B9%98%ED%95%98%EA%B8%B0/) The website post is written in Korean by the way.

2-4. Jekyll Tutorial and a bit of Liquid.

Fortunately, Jekyll Docs(https://jekyllrb.com/docs/) are quite well organized and they also provide a simple tutorial for beginners like me. So, after having finished the simple tutorial, I could now briefly understand how Jekyll works in a big picture sense. But it was still a bit confusing so I had to write down the file structure of Jekyll to solidify my understanding.

Additionally, I also had to pick up bit of liquid syntaxes to use Jekyll more efficiently. Jekyll itself is a static site generator which enables templating html file in a structure wise and Liquid enables users to write html codes programmatically with variables and control flow syntax etc. Liquid Docs are also quite well-written, so it was quite easy to pick up. (https://shopify.github.io/liquid/)



<Jekyll File Structure>

_Config.yml : set data to use under "site" object in html
manages global config data
↳ ① locate sass directory / Style

_layouts : Stores templates (.html) for layouts
↳ such as default, index, Contact etc...

{% include %}

_includes : Stores reusable components (.html), a.k.a partials, which can be used in creating layouts.
↳ such as header, navbar, footer etc...

Site.data.~

_data : Stores data (.yml or .json etc...) to be used in .html files in Jekyll.
These data is accessible through "Site.data.(datafile name)" in .html

_sass : Stores sass partials which can be imported and used in main sass files of assets/css directory.
↳ How it works :

assets : Stores asset data (css(sass) / image / js)
and these can be used in anywhere we want to use those asset data.
↳ Asset data reference point !!

_posts : Stores our post (or any) Contents in .html or .md
with a front matter for formatting.
↳ Such as
---
layout : default
title : First
---

And content must follow a naming convention : Y-M-D-title

2-5. Jekyll plugin (jekyll-paginate-v2)

So far, with Jekyll and Liquid, I could already have a working blog website. But, one thing bothered me was that Jekyll natively supports only a single pagination. In other words, we can use pagination feature only for 1 category. However, I wanted to have multiple categories on my blog and pagination feature for each of them. So, I had to look for a Jekyll plugin to make it happen and soon I found out that jekyll-paginate-v2 plugin supports multiple paginations (Hooray~!). You can check how to use the plugin from this site. (https://github.com/sverrirs/jekyll-paginate-v2)

But, the problem was github pages does not support the jekyll-paginate-v2 plugin ☹. You can check which plugin github pages supports from here (https://pages.github.com/versions/) There was also a way around this problem of course. And that was to use git submodule.

2-6. Git submodule

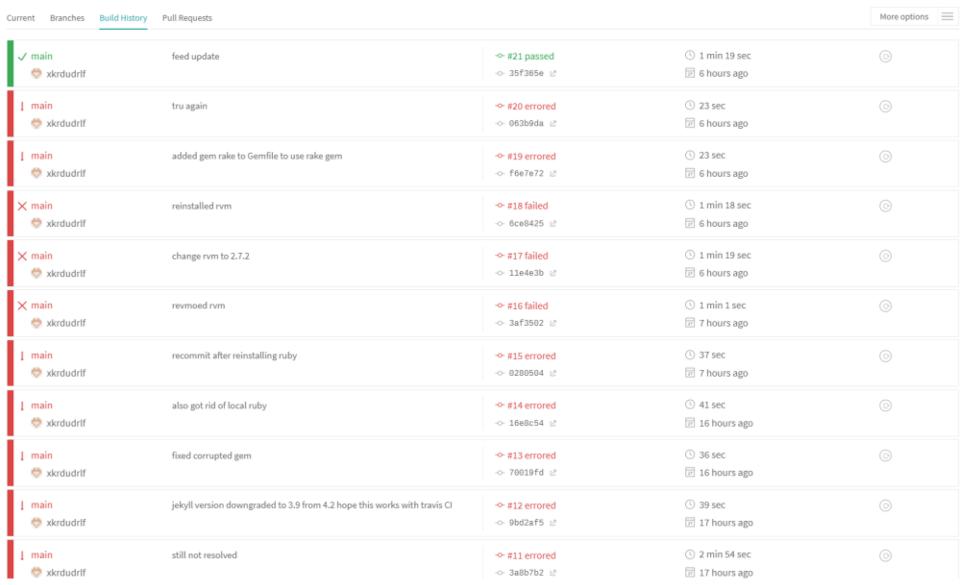
By using a git submodule, I could build a blog website with Jekyll plugin which is not supported by github pages. The way how it works is that we create another git repo with the same content and add original git blog repo as a submodule to the new git repo. Then, with some modifications on _config.yml file, we can direct contents of Jekyll generated _site folder to the original git blog repo and original git blog repo will simply use it to deploy the website without using Jekyll in it. Since Jekyll won't be used in the original blog repo, we also won't have any problem with unsupported Jekyll plugins.

However(Again..), with git submodule only, we have to manually git add/push/commit for 2 repos separately, which can be quite bothersome. So, we can automate this process via Travis CI.

2-7. Travis CI

Travis CI is a platform which automates the aforementioned process for us. All we need to do is write some config files which is also not easy unfortunately ☹.. But don't despair! I found a really helpful website to refer to with regards to this. From this site, I could learn how to use git submodule and Travis CI as well. Really cool one! (https://moon9342.github.io/jekyll-travis-ci-public). One thing to note here is that we should customize the data inside 3 files according to our set up environment. For example, rvm tag of travis.yml should have the version of ruby you want to use, not just 2.4.2 specified on the website manual.

After All setup, if we just add/push/commit to our new git repo, then the generated _site folder contents will be placed into git blog repo automatcially by Travis CI (How convenient! Though it was not convenient at all till I figure out how to debug some errors on Travis CI 😅😅)



Current | Branches | Build History | Pull Requests | More options

✓ main xkrdudrlf	feed update	#21 passed 35f365e	1 min 19 sec 6 hours ago
! main xkrdudrlf	tru again	#20 errored 063b9da	23 sec 6 hours ago
! main xkrdudrlf	added gem rake to Gemfile to use rake gem	#19 errored f6e7e72	23 sec 6 hours ago
✕ main xkrdudrlf	reinstalled rvm	#18 failed 6ce8425	1 min 18 sec 6 hours ago
✕ main xkrdudrlf	change rvm to 2.7.2	#17 failed 11e4e3b	1 min 19 sec 6 hours ago
✕ main xkrdudrlf	revmoed rvm	#16 failed 3af3502	1 min 1 sec 7 hours ago
! main xkrdudrlf	recommit after reinstalling ruby	#15 errored 0280504	37 sec 7 hours ago
! main xkrdudrlf	also got rid of local ruby	#14 errored 16e8c54	41 sec 16 hours ago
! main xkrdudrlf	fixed corrupted gem	#13 errored 70019fd	36 sec 16 hours ago
! main xkrdudrlf	jekyll version downgraded to 3.9 from 4.2 hope this works with travis CI	#12 errored 9bd2af5	39 sec 17 hours ago
! main xkrdudrlf	still not resolved	#11 errored 3a8b7b2	2 min 54 sec 17 hours ago

3. Documentation phase.

Since I've completed my blog website project though I will add some more features and fix some design flaws in the future. At least, I completed a cycle from a design stage to deployment stage. So, I document things I've done here so that I can refer back to this doc later in the future. (I might decide to create another git blog in the future right?)