

# Sharing What We've Learned



## Anthony Reimer

University of Calgary

 @AnthonyReimer

 @jazzace

 <https://github.com/jazzace>

## Vaughn Miller

Lafayette College

 @Vaughn\_E\_Miller

 @vmiller

 <https://github.com/vmiller>

## Out of the Box

How to best create deployment images?

- Use disk images and packages
- Really, use packages, not scripts or hand-rolled collections of files
- Really, use packages
- Packagemaker, JAMF Composer, Iceberg, loggen

*Josh Wisenbaker & Joel Rennich*

*"Best Practices for Client Management"*

*Macworld Conference and Expo 2008*

[Anthony:] I went to my first Mac conference in 2008. (reference presentation, mention how I wanted to tell everyone about this new way of doing things that made so much sense)

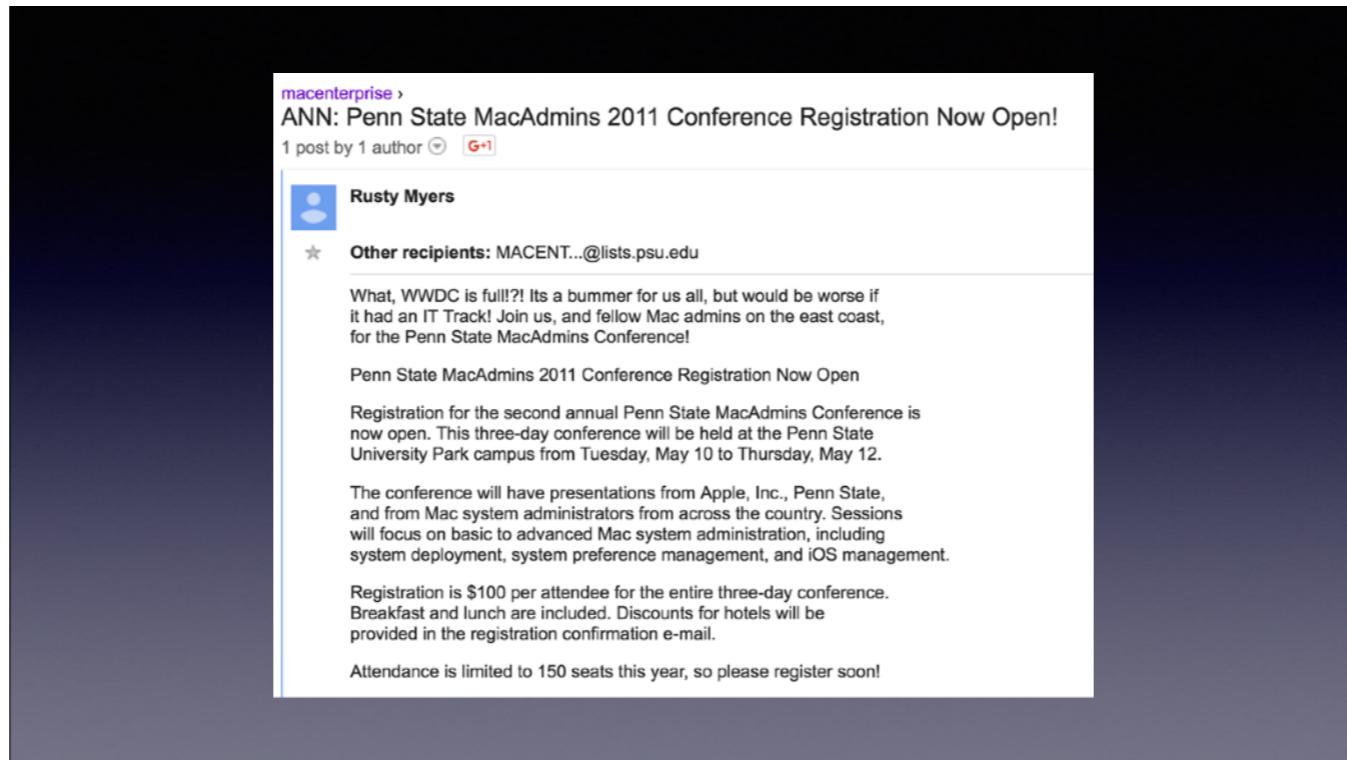
# Disk Imaging and Software Deployment

How do we do it?  
How should we do it?

Anthony Reimer  
Faculty of Fine Arts

*Presentation to University of Calgary  
Mac Lab Managers/Support Annual Meeting, Feb. 2008*

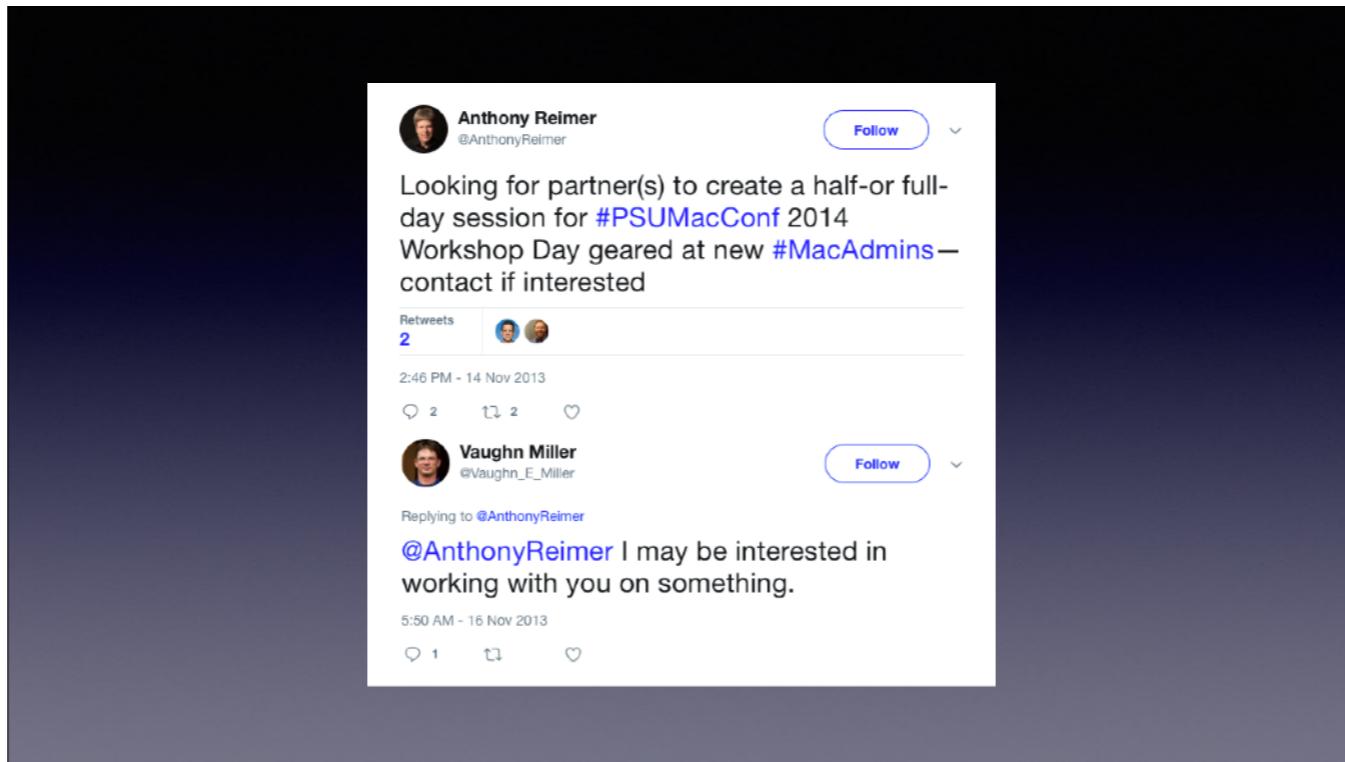
So a month later, I shared what I learned at a campus meetup (as we would call it now). This was the most natural way for me, as I trained to become a teacher.



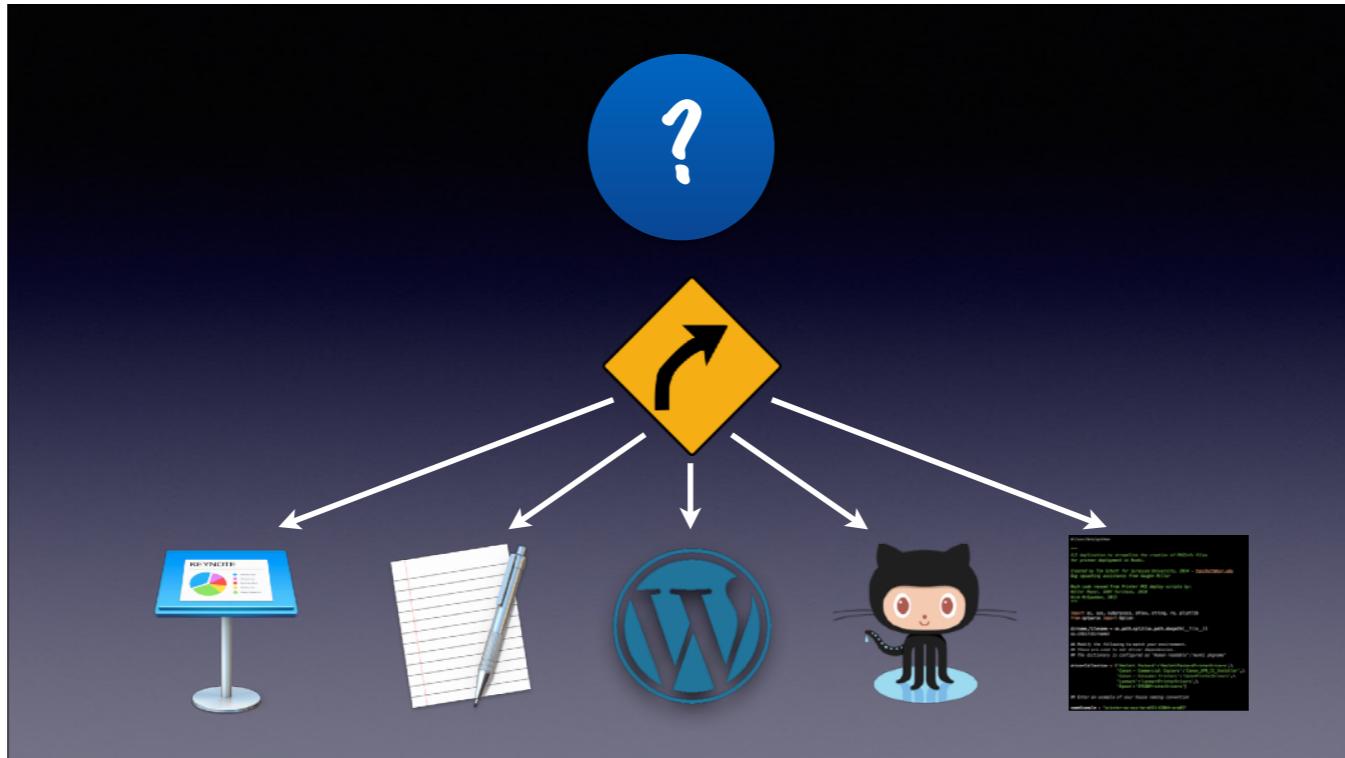
[Vaughn:] For me it all started on the Mac Enterprise email list. I joined up when I first started digging into deploying Macs, and it is the first place I shared information with others. MacE is also where I first heard about this conference, and started meeting peers.

I was inspired by others willingness to share and created a GitHub account to share some scripts. I wanted to learn some about running a website, so I started a blog running on WordPress.

Some of my first speaking was done at departmental brown bag lunches.



[which is how Vaughn started presenting at bigger conferences like this one: he wanted to give back]



We'll briefly talk about Why we share, and then • focus the rest of our time on the Ways we share [auto•], including help on How to share most effectively.

**Why?**





## Help Future You

- Online memory
- Two for one

Write a blog post to document something you need to remember

If it helps you, it is likely helpful to someone else



## Giving Back

- Just Google it
- Pay it forward

Who among us haven't Googled an answer? If someone hadn't shared, we would not be able to google the answer

Has someone more experienced than you taken the time to help you with a problem? Consider doing the same for someone with less experience than you



## Build a Personal Brand

- A more selfish motivation
- Digital portfolio
- Be aware of online personality

Lets be realistic, the motivation does not need to be purely altruistic.

Your writings and code form a digital portfolio that a future employee can review. I actually was asked about one of my blog posts in a job interview.

Works both ways – don't leave a negative impression



What are the ways we can share, and how do we get there?



## Ways to Share

- Online
- In Person
- Writing
- Sharing Code

Some broad categories of sharing



## Online Communities

- "Easy"
- Note taking
- Diversity
- Cultural norms

Low barrier to entry makes this an easy way to become involved in the community. This is how many get started.

Can be used for real time note taking. This could be during events like a conference, or perhaps analyzing a breaking event like a Malware outbreak.

An advantage to online community is the diversity. It becomes easy to interact with people you may not get a chance to meet in person. Online communities easily cross time zones, national borders, and diverse industries.

This may be obvious, but every community has it's cultural norms. Spending a bit of time in "read only" mode can be helpful with starting out.



## Online Communities

- Slack
- Twitter
- Jamf Nation
- Email Lists



# Slack

- "In" place to be
- Questions are OK
- Graduate to answers
- Start small

Is the latest "In" place to be. Slack is quite popular for many communities, and Mac Admins are no exception. Much of the coordination for this presentation was done using Slack

Just as acceptable to ask questions as to give answers

Join for the questions answered, stay for what you can give back. Before too long, someone else will come along and ask the same question you got answered earlier.

Contributions can be small – someone may put out a call for testing or be collecting information.



## Slack

- Rapid interaction
- Walled garden
- Report out

Good for rapid feedback and interaction. This can be great when troubleshooting a perplexing problem.

Does not show up search engine results

Because of the "walled garden" aspect, solutions still need to be documented. This could be email lists, or blog posts.



## Twitter

- Amplification
- Discoverability
- One way
- Bite sized

Good for amplifying. Often used for sharing finds, and likes and retweets share it with a larger audience

Discoverability – Perhaps Twitter's greatest strength is that it is searchable. Follow a hashtag, or even do a full text search. Search for topics of interest, find new accounts to follow.

Relationships can be one way

Hard to have nuanced conversations in 140 characters



## Jamf Nation

- Hosted by Jamf
- All are welcome
- Searchable

Jamf Nation is a discussion forum hosted by Jamf. Despite being hosted by a vendor, there is much generally applicable content.

There is no need to be a Jamf customer to participate. Anyone can sign up for an account, and there are non Jamf customers who contribute.

Jamf Nation is well indexed on the web. Many times a google search looking for an answer will lead to a thread on Jamf Nation.



## MacEnterprise

- Long form
- Searchable
- Declining

The MacEnterprise email list is hosted by Penn State University and has been around for a long time. (Archives go back to 2004)

Good for longer form correspondence

Archives are available online and are well indexed by search engines.

In more recent times, the volume of discussions seems to have moved to other venues (Probably Slack). The list is still valuable for announcements.



## Other Lists

- Long form
- Report back
- Roadmap

Technology specific lists such as for Munki or AutoPkg.

Again, good for longer form discussion

Sometimes a report back from Slack. Since Slack is not searchable on the web, important discussions can be summarized on the list.

Often used for discussing future/roadmap. This type of discussion lends itself to a longer form, with thoughtful writing.



## In Person

- In our work place
- Local or regional meetups
- Larger conferences

Some ways we might share in person



## In our work place

- Lunch talks
- Staff meeting

At Vaughn's previous employer they had some brown bag tech talk lunches within the department. It was a venue to present something interesting that you were working on.

Present in front of co-workers at a staff meeting



## Local meetups

- Informal
- Speakers needed
- Test run

Local meetups are becoming more common. These tend to be fairly informal affairs.

Most meetups are looking for more speakers.

A local meetup is a great place to test run a talk you plan to give at a bigger event. Because it is a less formal setting, you can even ask for feedback.



## Conferences

- Meet peers
- Informal Sharing
- Presenting

Great place to interact with peers that you may not have known you had.

Lots of informal sharing opportunities sometimes referred to as the "Hallway Track" You need not be a presenter to share with others at a conference.

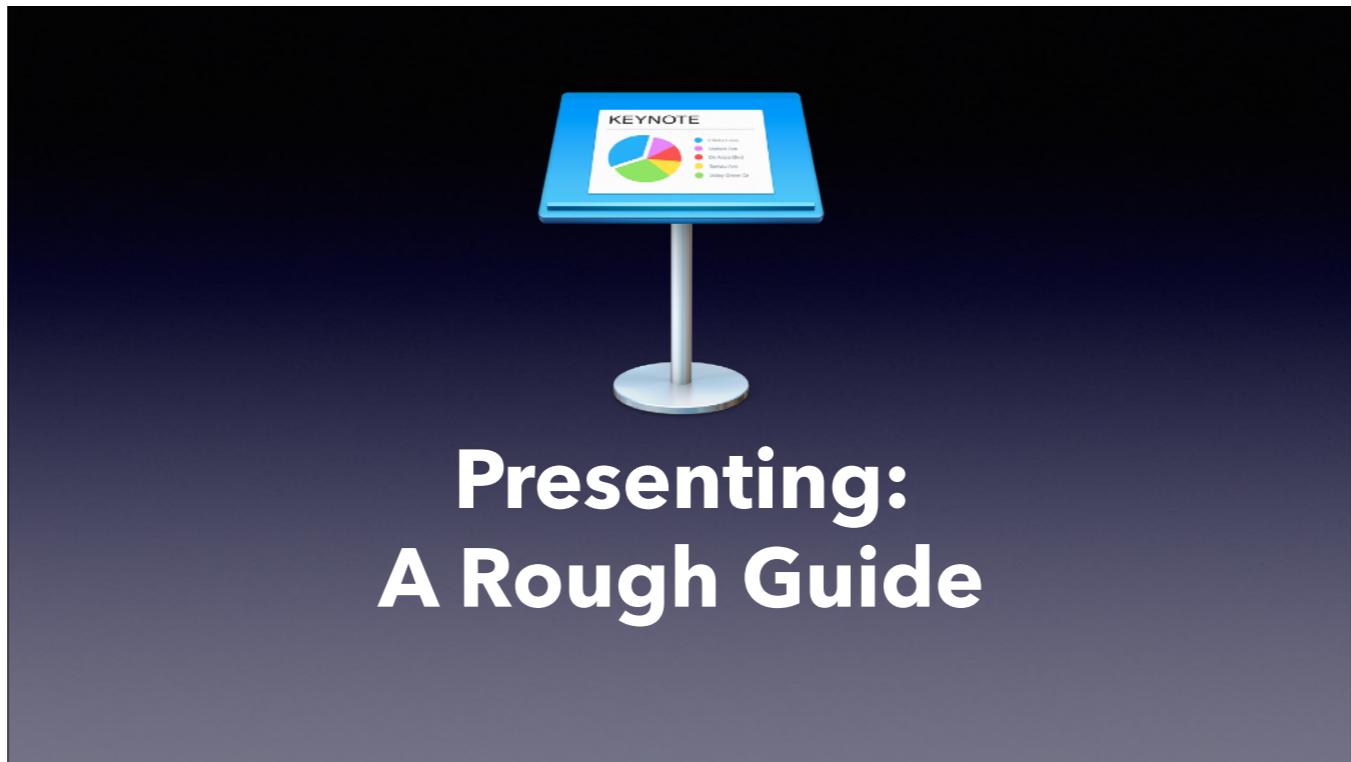
That said, consider presenting – Everyone has a story!



# Presenting at Mac Admins

- Thursday @ 12:30, Deans Hall 2

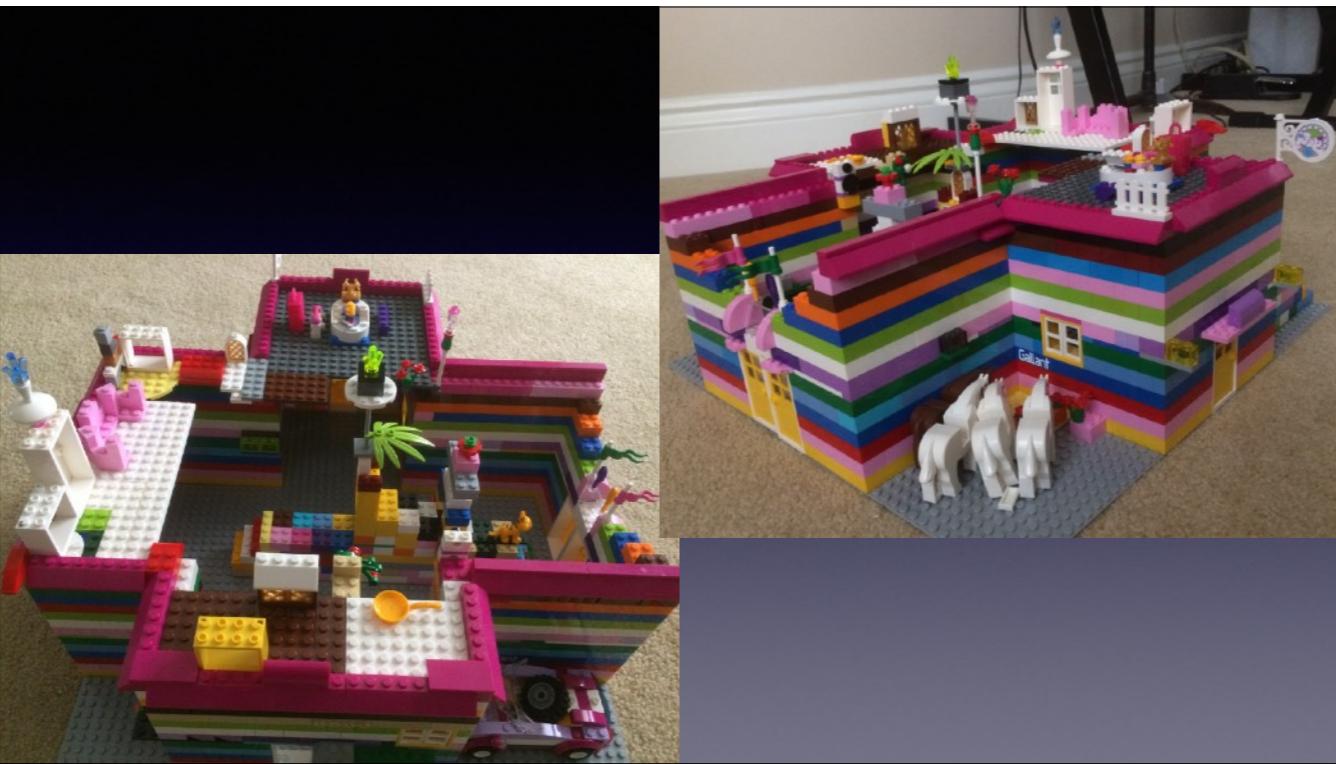
Rusty Myers





# Presenting

1. Finding your process
2. Finding your style
3. Technical considerations
4. Rehearsal
5. It's Showtime, Folks!



Lego castle I built with my 10-year-old niece this past December. What can you tell me about our process?



## Lego Process

- Build to edge of platform
- Pick 1-2 colours per layer
- Build a layer at a time, then evaluate
- Stagger bricks for better structural integrity
- It should appear to be functional
- Theme: Princess Castle



# Finding Your Process

- What do you know / already have?
- How do you think?
- What is the form / structure you will use?
- What story are you telling?

1. What are your Lego blocks? Some documentation or blog articles you wrote? A recent project you did at work? A research process that you are about to document? You'll might need more Lego (or certain shapes or colours of Lego) before you're done.
2. Do your Lego blocks need to be sorted into separate colours before you start, or can you just grab bits as you need them?
3. Do you design the Lego building framework and then fill it in, do you build it by section and then assemble it into something bigger, or do you start at the beginning and design it "just in time"? [adopted pieces from previous buildings in some cases]
4. What are you building with your Lego? In our case, it wasn't just a building, it was a castle. It had definite ideas about colour and size, about place (horses, car). Can you imagine who lives in this castle?

# Finding My Process

- What I've learned that I want to share
  - *My work context*
  - *A discussion where I had something to contribute*
- An abstract to describe that
- A general outline with some specifics,  
subject to change

1. examples of my presentations, my AFP548 article
2. Need it for applying to conferences; good to refer back to



## Finding Your Style

- Slide Content
- Slide Frequency
- Script or Outline
- Podium or not

1. Highlight vs. Outline (Apple example; what do you see at your business?); mix of words and pictures
2. Fast-paced (PechaKucha 20 x 20s) or not; builds on slide or not
3. Reading is fine; outline requires more practice for timing but can be more engaging
4. Podium can provide security; sometimes, that's the only option available; if you like to walk, then you need to be able to be off notes or have some system to get your notes, you'll also want a remote; podium or not (and room config.) will also drive whether you want a laser pointer or use highlighting either via animation • or a markup tool like you have on the iPad



## Technical Considerations

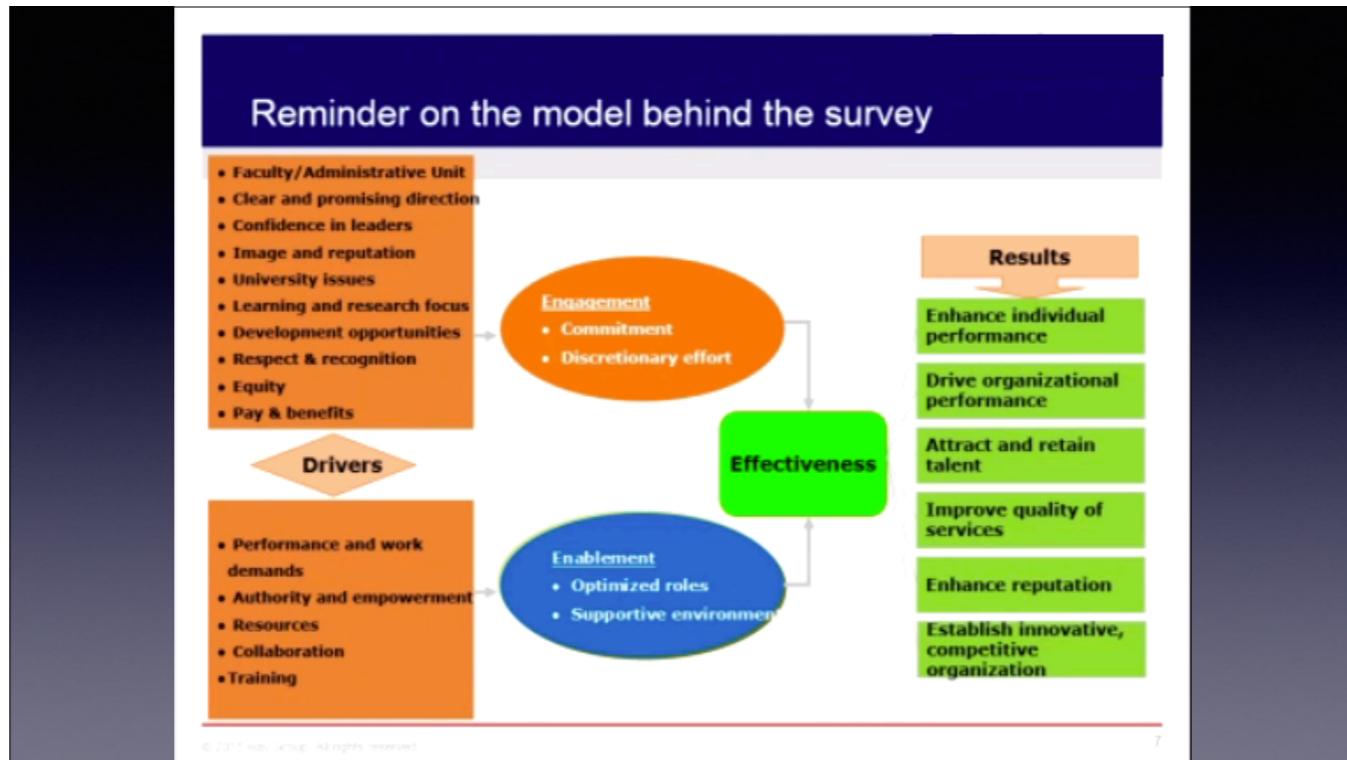
Your slide deck, regardless of style, shouldn't just be a large block of text that you read out. If all you do is read the text on the screen, the audience will start ignoring you. Just put key words or visuals on the screen as a reminder of your point and elaborate on them as you speak. The only thing right about this slide is that the font size is decent for reading.



## Technical Considerations

- Text should highlight





Actual slide (from video). I have blocked out the branding on the slide to protect the guilty. This might be OK in a small room with 10 people or less. It is certainly better as multiple slides. In this case, the speaker didn't write the deck and referenced that people couldn't read it, so she went through it by section—made the best of it. So many ways to fix this. (ideas from audience?)

# Technical Considerations

- Text should highlight
- Size Matters
  - ▶ *Simplify*
  - ▶ *Animate*
  - ▶ *Slides are free*
  - ▶ *Fault tolerance*

Accept the default font sizes in templates/themes until you have more experience.

# Technical Considerations

- Projector/Display
  - ▶ *Aspect Ratio*
  - ▶ *Resolution*
  - ▶ *Fidelity*
- Bullet Count

1. mostly 16:9 or 16:10 these days
  2. 1920 x 1080 will give small menu bar if you give demos
  3. Colours wash out and contrast is often lower on projectors; author accordingly
- Aim to limit to 3 or 4; think of Apple's summary slide model as a valid exception

# Technical Considerations

- Intellectual Property
- Demos
  - ▶ *Live or Canned?*
  - ▶ *Resolution*
  - ▶ *Presentation Account/Screen*

QuickTime recording

# Technical Considerations

- Sources for icons:
  - ▶ /Applications/\*.app/Contents/Resources/\*.icns
  - ▶ /System/Library/Core Services/\*.app/...
  - ▶ /System/Library/Extensions/  
IOSCSIArchitectureModelFamily.kext/...
  - ▶ /System/Library/Core Services/CoreTypes.bundle/...

You don't have to screenshot or do fancy tricks to get icons from your Mac. Of course, you can do a Get Info and try to copy an icon, but that doesn't always work. Why not go to the source?

# **Keynote Technique Demo**

Meta demo: screen record the creation of a deck in Keynote that uses icons from the locations mentioned on the previous slide. Demonstrate Magic Move both by sliding and scaling. Stop recording, trim video, drop into Keynote. Target time: 5 minutes.

# Rehearsal

- Private run with timer
  - ▶ *include markers*
  - ▶ *Keynote Rehearsal Mode*
  - ▶ *HDTV as projector proxy*
- Dry Run

know that you might talk faster than normal (or than you've ever rehearsed); have some time landmarks in your notes to give yourself an idea of how fast or slow you are moving; if you talk from an outline rather than from a script, this also helps you determine whether you are getting sidetracked or whether you have time to elaborate more.



**It's Showtime,  
Folks!**



# Showtime!

- Adapters (VGA, HDMI)
- Remote

# Showtime!

- Backup plans
  - ▶ Copies
  - ▶ Format
  - ▶ Device
- Sharing your deck

1.2 PDF as backup; can you give your presentation without slides?

Mac Admins specific: put links in your deck and share your deck afterwards, so you don't have to wait for people to finish writing things down. Tell people you'll be sharing your deck. If you don't want to share your deck for some reason, do a podcast-style "Show Notes" with all the links and highlights. Conf. organizers will often share resources that speakers provide; you can share yourself with the tips in this workshop.

# Writing





# Writing

- Notes
- Documentation
- Blogging

You can start with writing notes for yourself. These could be text notes, or in a note taking app if you prefer.

Internal documentation for you and your co-workers can be as simple as sharing your notes on shared storage. A wiki is a good option for shared documentation, but notes shared through note taking apps are an option too.

There is no reason a blog can't be notes to yourself shared publicly. If the information is helpful to you, there will be someone else who finds it useful.

Note that you will probably need to continue to maintain separate internal documentation for both sensitive information and for institutional control.

For this workshop we are concentrating more on public sharing.



# Writing

- Making available publicly
  - *Work or Personal?*

But if you are going to publicly share, then you have to decide whether you will be doing this using your own time and resources or that of your employer. One reason to do it on your own time would be building that personal brand we spoke about off the top. But we are not telling you that you need to do it that way. [Anthony:] Even though I've had a personal web site for 20 years, I started posting my presentations on my Lab's web site at work because I was doing that work on work time and hosting bandwidth was not an issue. [back to Vaughn:]...



WordPress



# WordPress

- Popular
- Simple
- Options

WordPress is a widely adopted content management system. There are many themes available

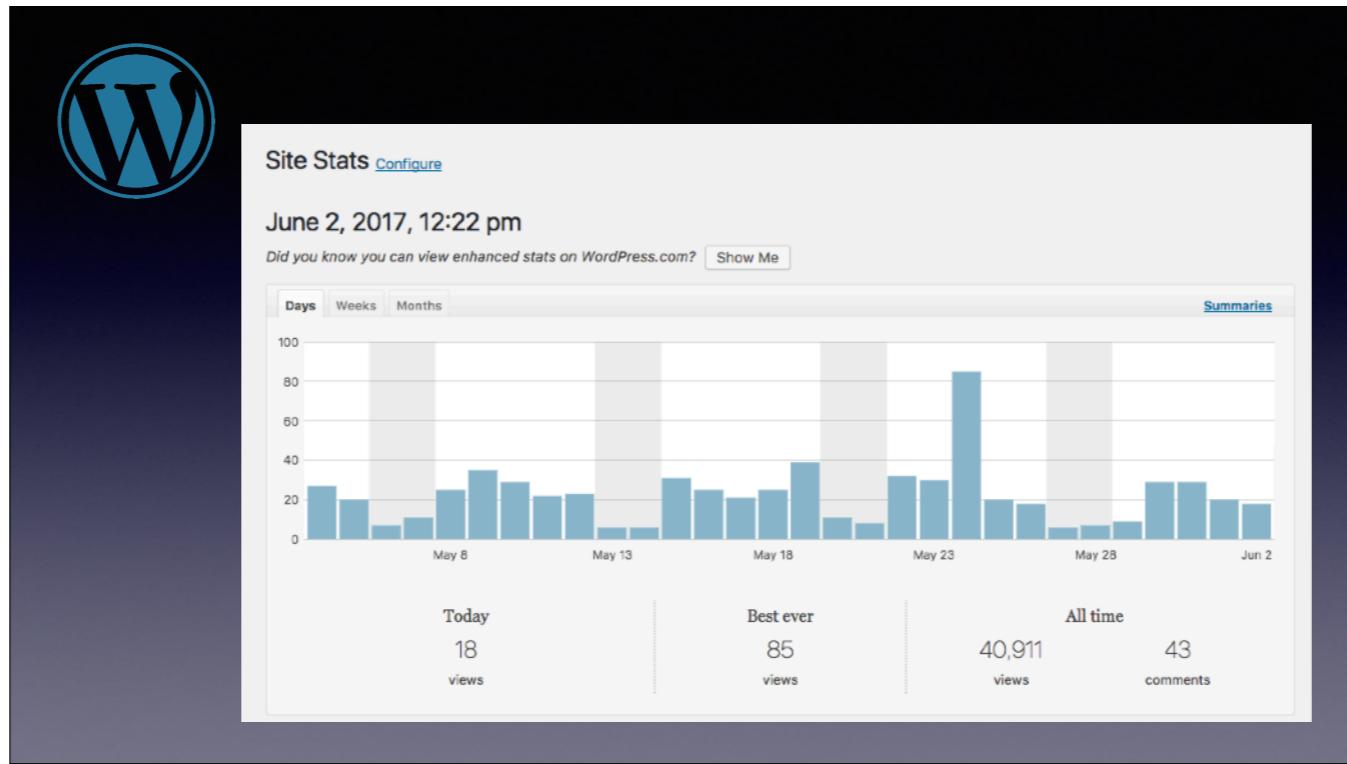
Simple to setup, allowing you to concentrate on the content rather than the technology. You do have the ability to develop your own custom theme if web development is something you are interested in.

Lots of hosting options. Simplest is to setup a website on [wordpress.com](https://wordpress.com), a free tier is available. Pay a little and get rid of the ads. You can also pay for traditional hosting and register a domain, you could also run your own installation on a VPS if you do choose to dig into the technology.

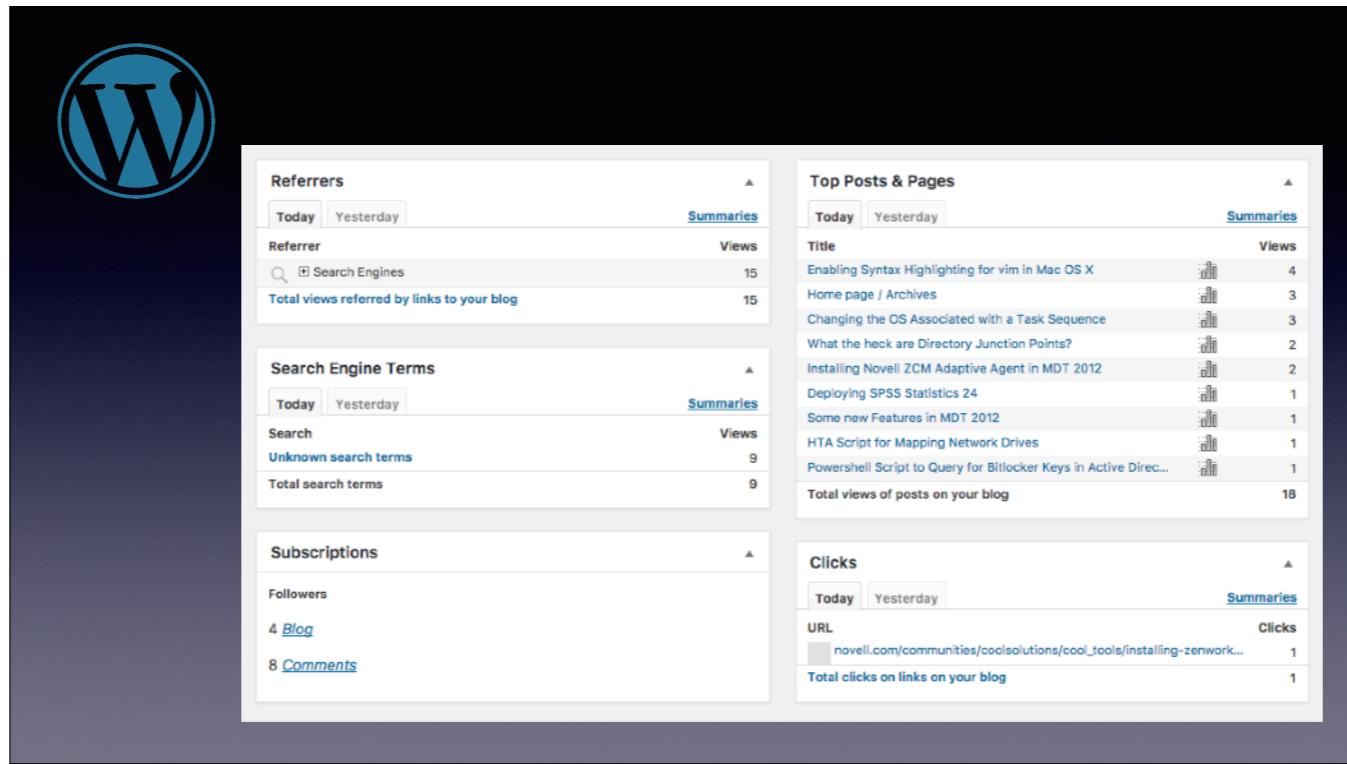


## Statistics

One nice feature you get with WordPress is statistics about your site



Usage stats. See how many visits to the site over time.



and which pages are most used. See search terms people used to get to your site and referral links

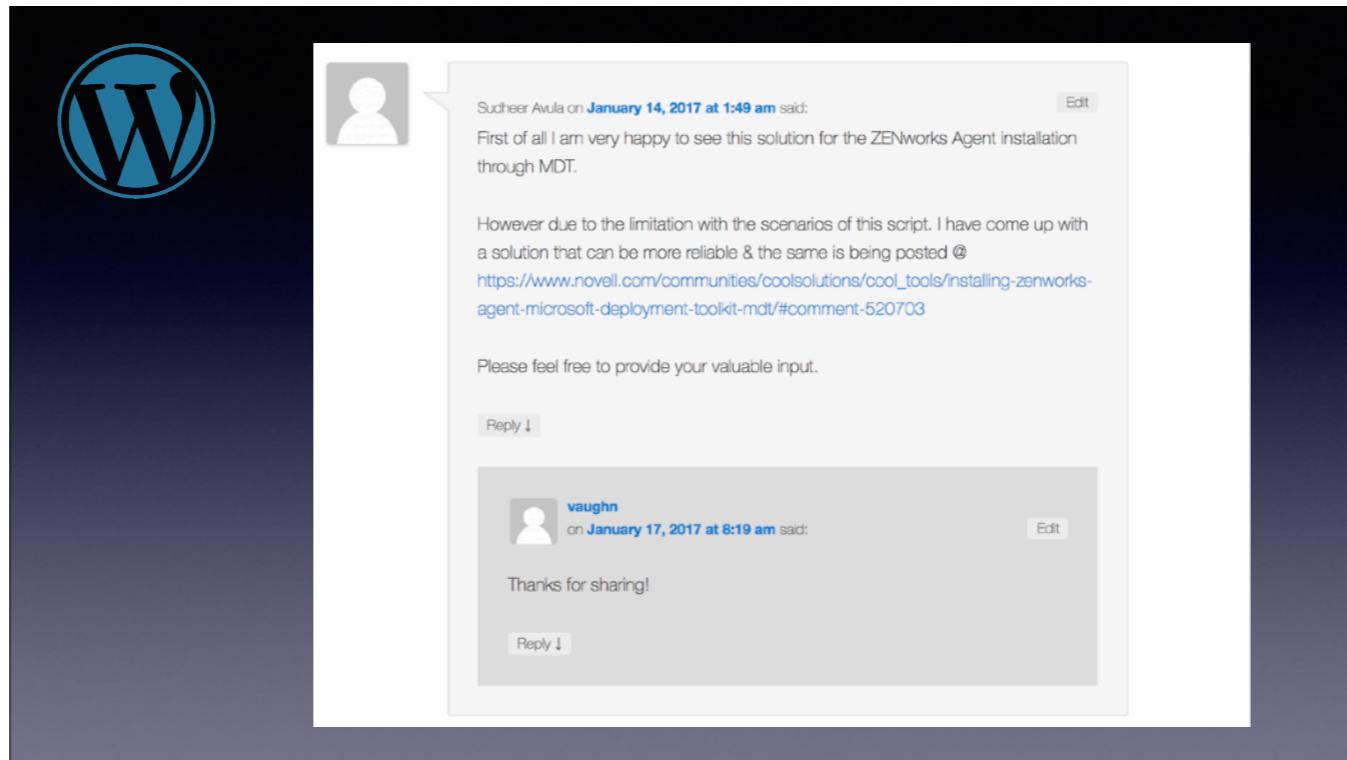


## Comments

Comments –

Depending on your view, comments can be a good or a bad thing. While I often subscribe to the "don't read the bottom half of the internet" view, I have found comments to be useful on my site. –Example–

You will want to use a plugin to help filter spam.

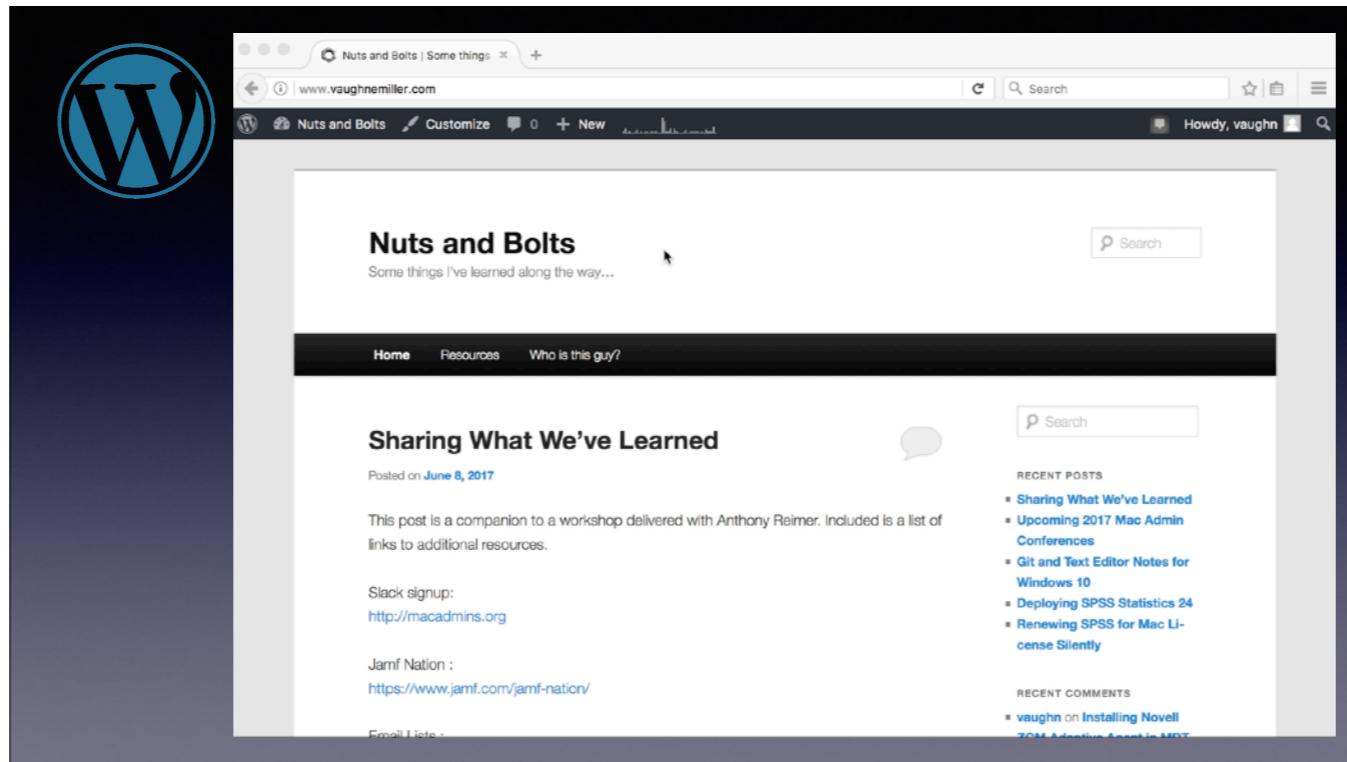


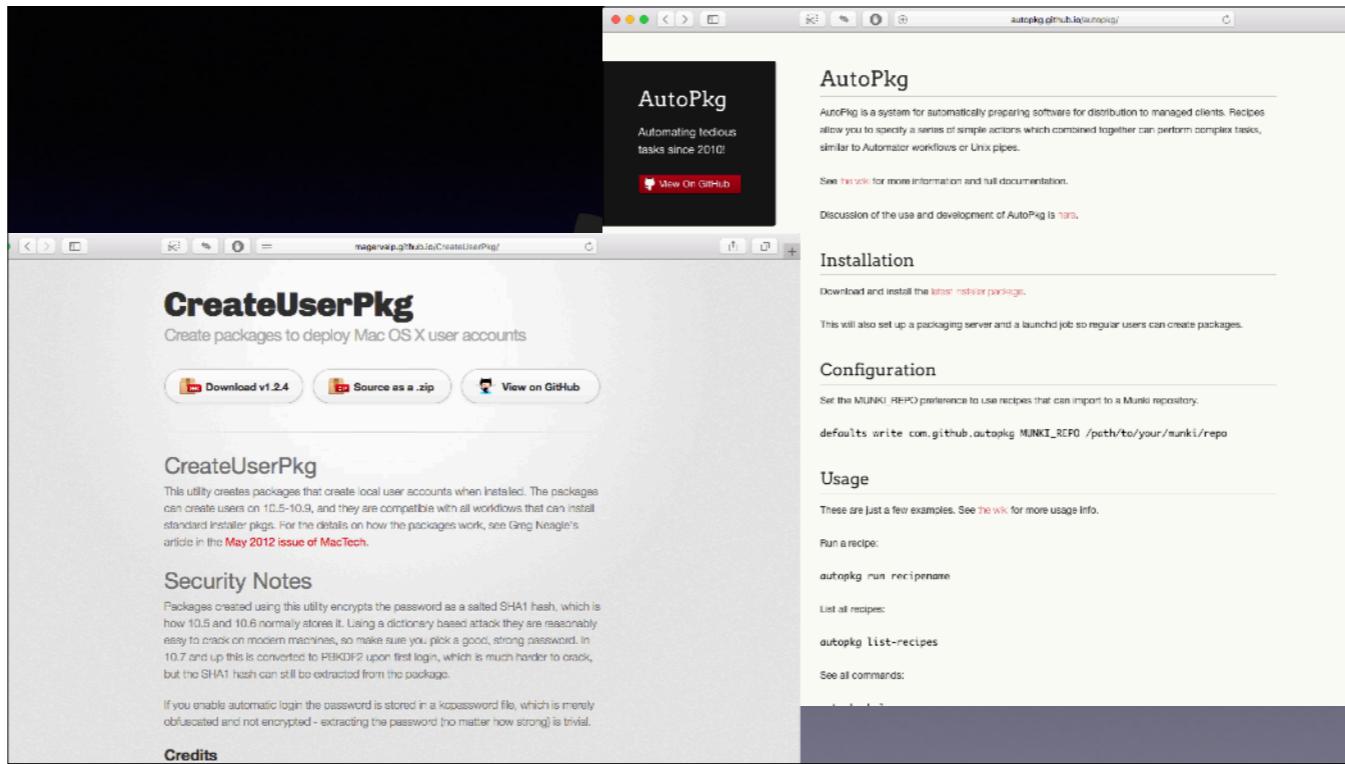
I've had some good discussions come from comments, including someone dropping a link to an updated version of a script that I no longer maintain. New visitors to this post will be better served by this newer script.



## Posting Content

Now a quick demo of posting content on a WordPress powered site.





Show of hands, how many people use GitHub right now? Is anyone already using GitHub Pages? • You may have seen project sites where the author offers a home page for basic information and project downloads. Those often use GitHub Pages. But GitHub Pages is not just for single pages like that; if you like the Git workflow and are willing to get your hands a little more dirty than with WordPress, then you might like blogging with GitHub Pages >



# Blogging with GitHub Pages

Navigating the Confusion, Harnessing the Power

I had a lot of stumbles along the way, so I'll focus on conceptual things so that navigating the documentation will be easier if you try this at home.



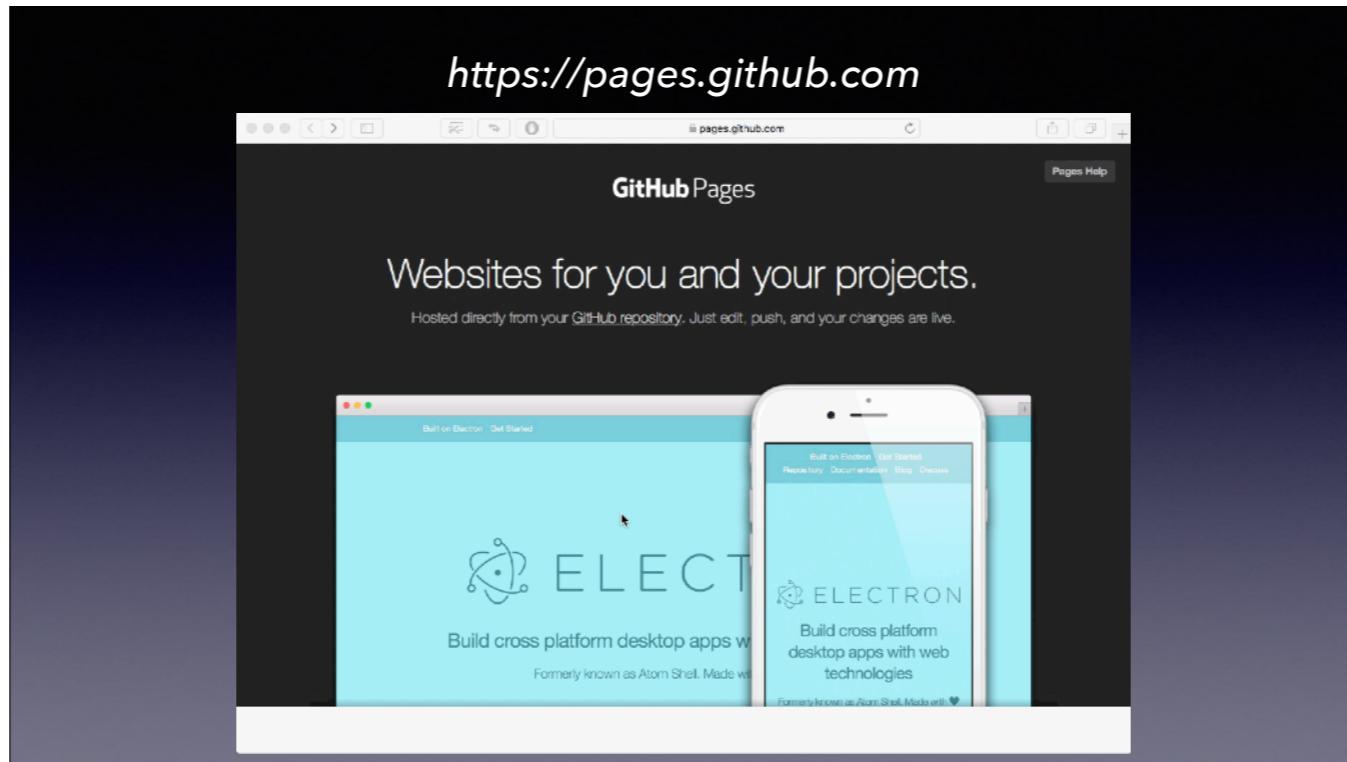
# What You'll Need

- GitHub Account
  - ▶ <https://github.com/join>
- GitHub Repo named *username.github.io*
- GitHub Desktop for Mac (or command line)
  - ▶ <https://desktop.github.com/>
- Text Editor
  - ▶ e.g., BBEdit (Bare Bones Software, [barebones.com](http://barebones.com))

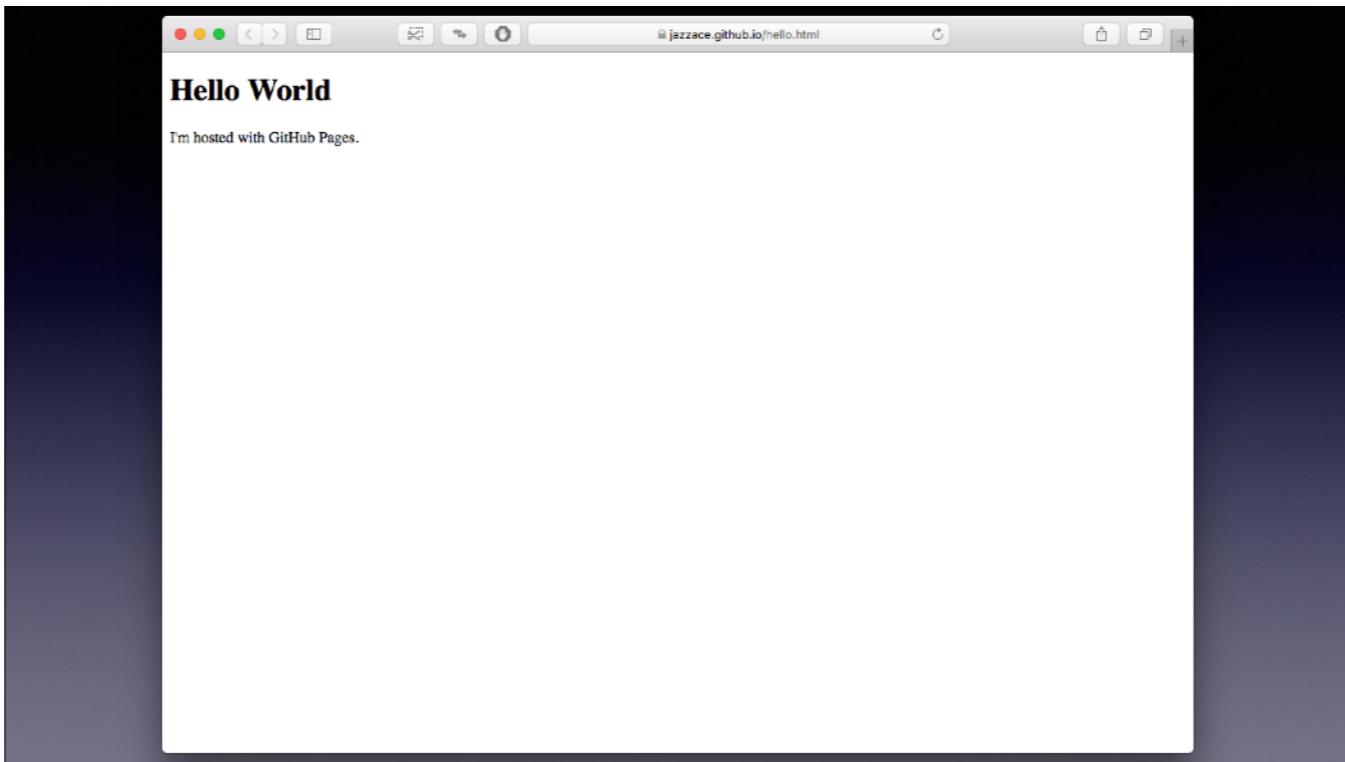


Concept 1

# GitHub Pages ≠ Blogging



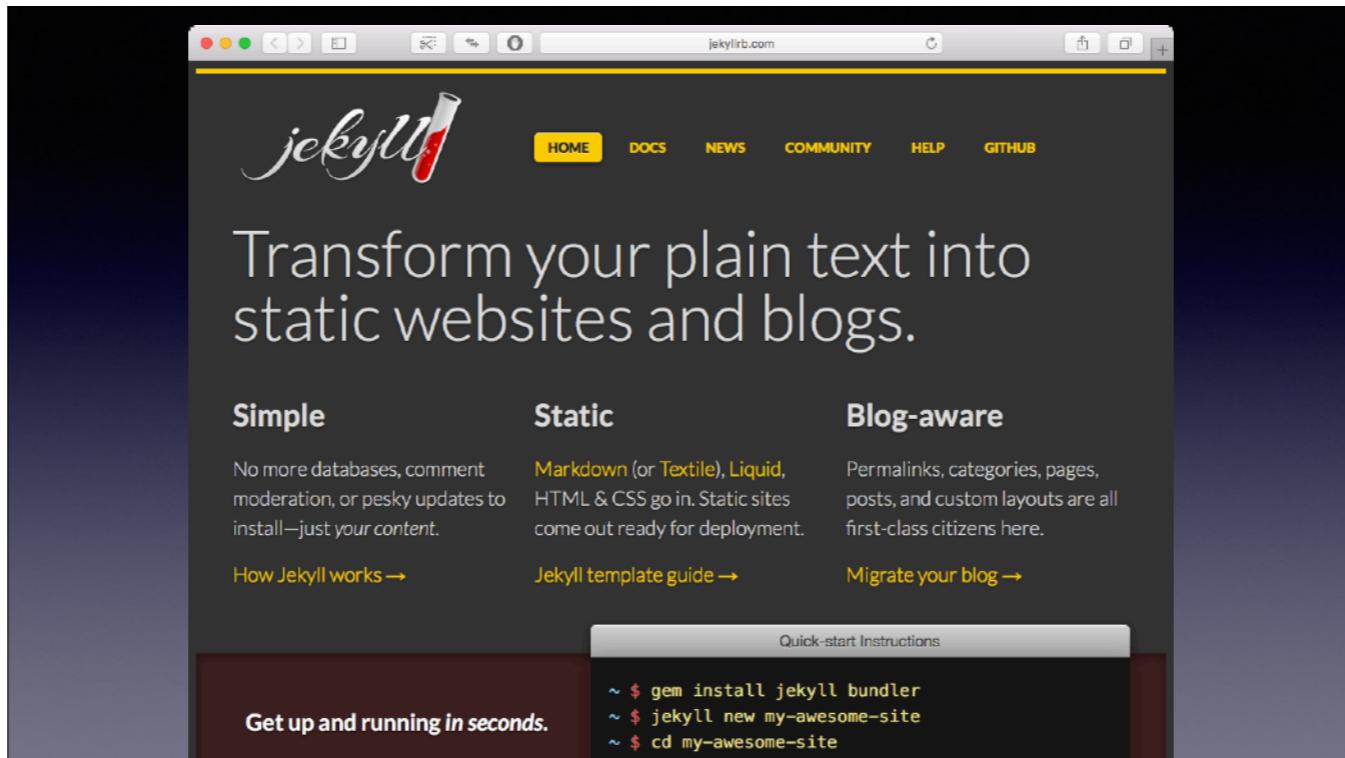
[autoplay] GitHub Pages is a way to host a web site on GitHub. We create a repository called username.github.io which has special properties that allow you to web host. If you commit raw HTML files to the repo, like the Hello World file it suggests here, >



You get exactly what you would expect. But we want to add easy blogging.



That's where Jekyll comes in. So what is Jekyll? Let's visit the project web site and find out.



If you know HTML or even Markdown, it has tools that make sense of your posts and build a static site. You upload, tell it to rebuild, and voilà. • It's integrated with GitHub Pages. When you commit a post to GitHub, the rebuild request happens automatically, so even easier.



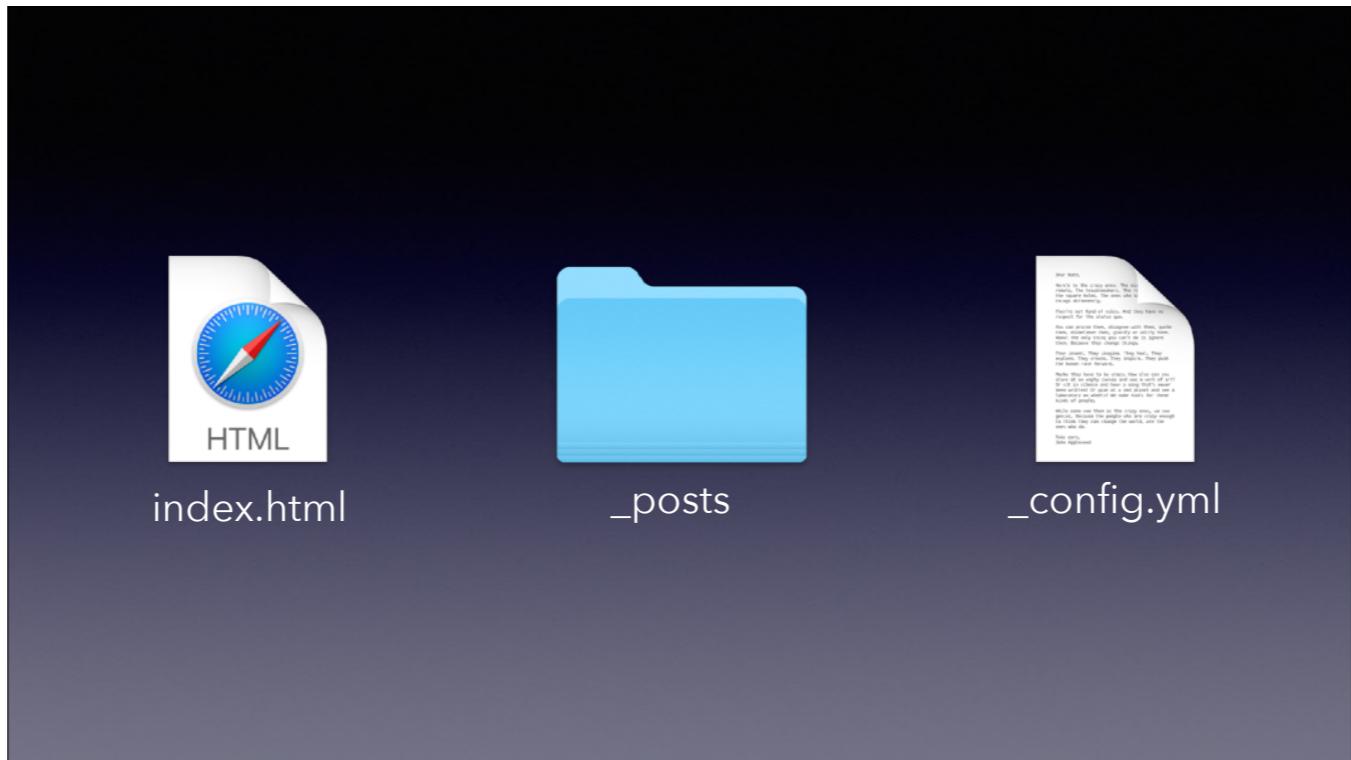
The biggest problem I had was separating which did what, since they are two separate things and can be used independently of one another. (AutoPkg/Munki example.) They're also documented separately, so you may find yourself flipping back and forth trying to learn how to proceed. But basically, GitHub Pages is like a web host with some features pre-installed and Jekyll is like a compiler, where you can “code” at a higher level and have Jekyll assemble it in such a way that it can be hosted on a web host that supports it. You can also leverage the work of others when it comes to using one of the provided Themes for Jekyll, just like you might use someone's AutoPkg recipes.



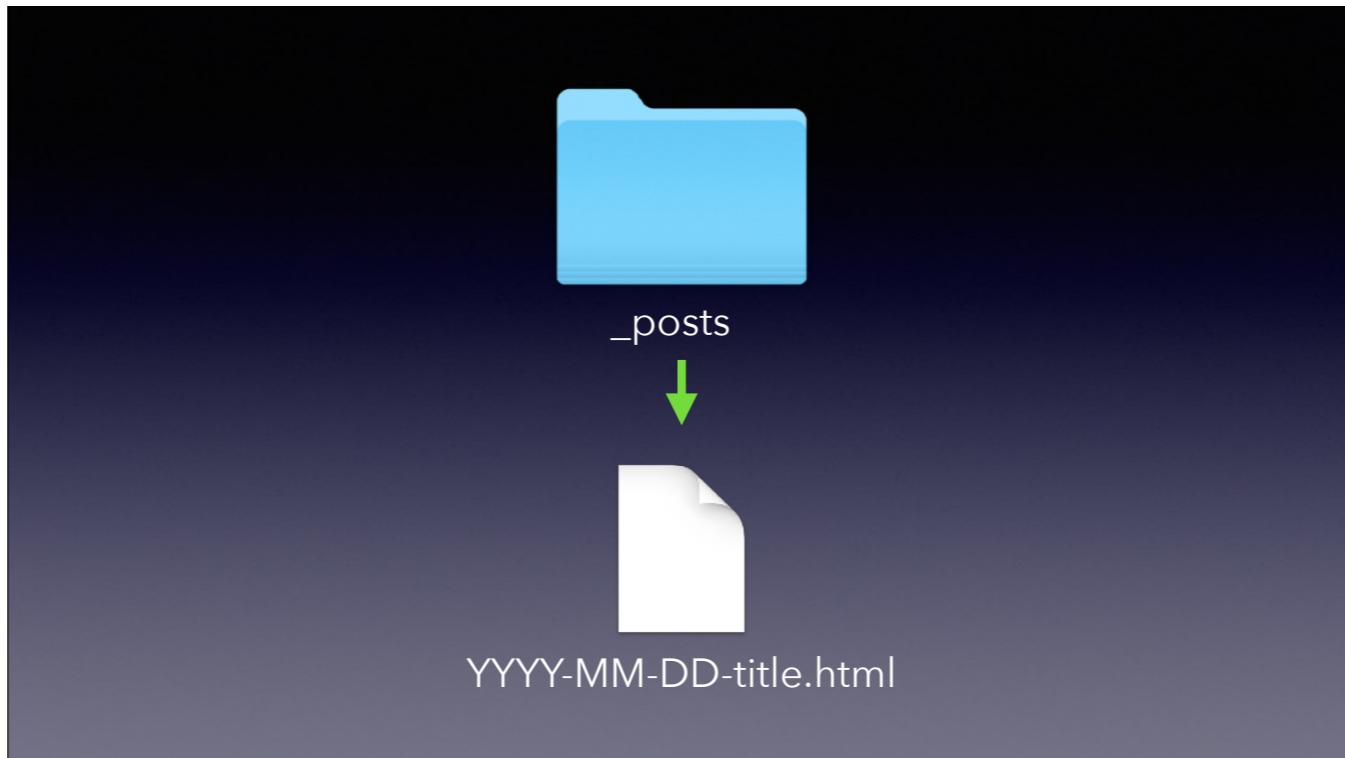
Concept 2

## Blogging Requires Structure

When using Jekyll, we need to adhere to a structure...



Jekyll requires you place certain files in certain places so that when it goes to compile your site, it knows where to find it. The good news is that with the recent versions of Jekyll, these might be all the structure you need, with the theme providing the rest. Let's look at the Posts directory



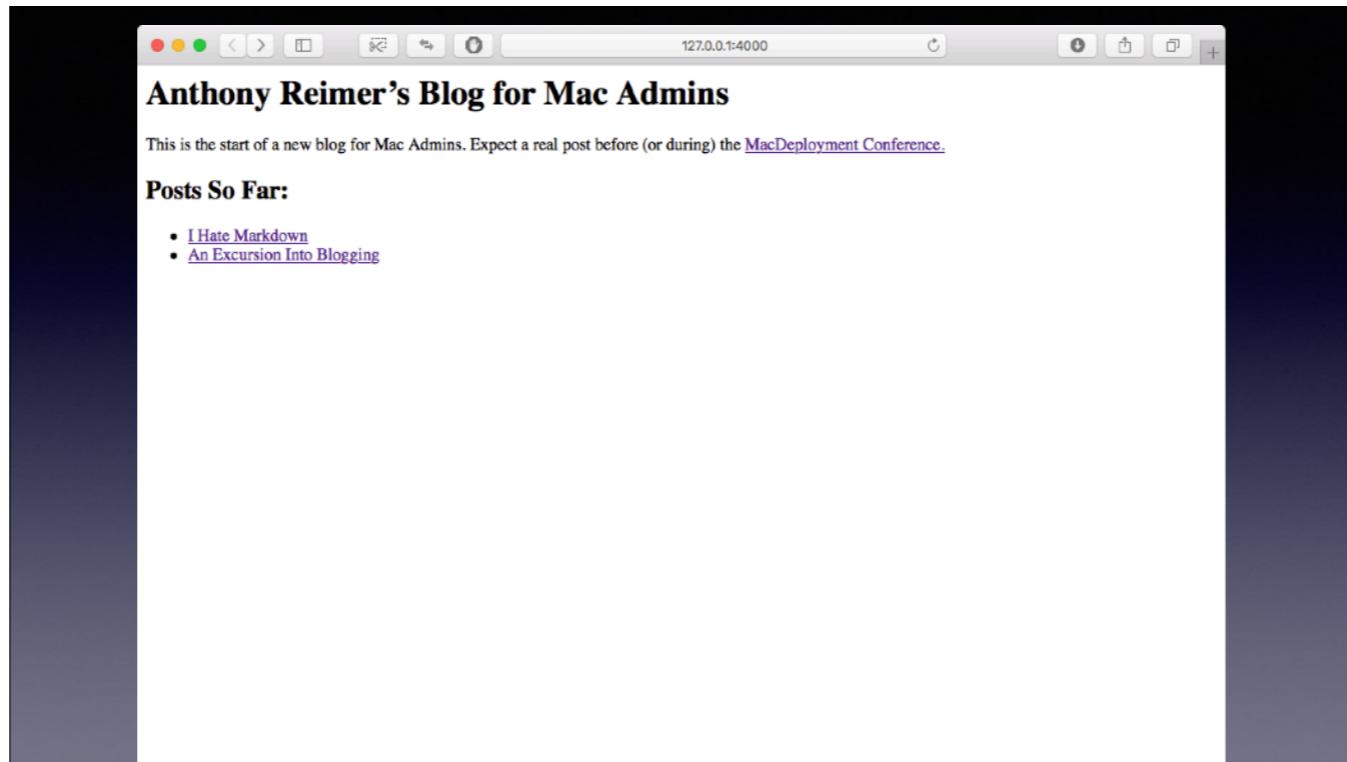
All blog posts need to go there and use a particular naming format. If you use Markdown, you use the .md extension instead of .html. Pretty simple so far, especially if you already know how to write HTML. But if I just write the post like a plain HTML file, I'm going to get the same plain text web page as before.



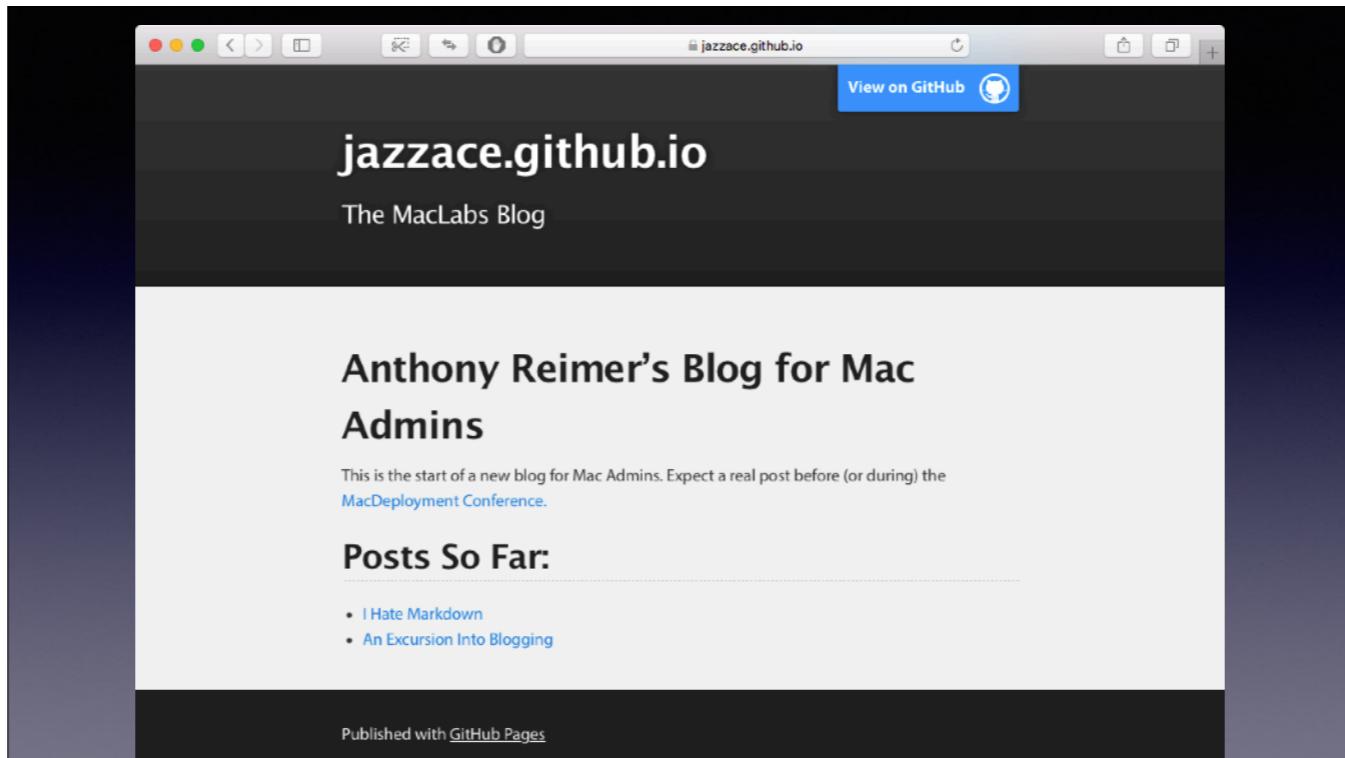
Concept 3

## Understanding Themes

So if we want “pretty”, we need to select a theme. Let’s break down what makes a theme in Jekyll.



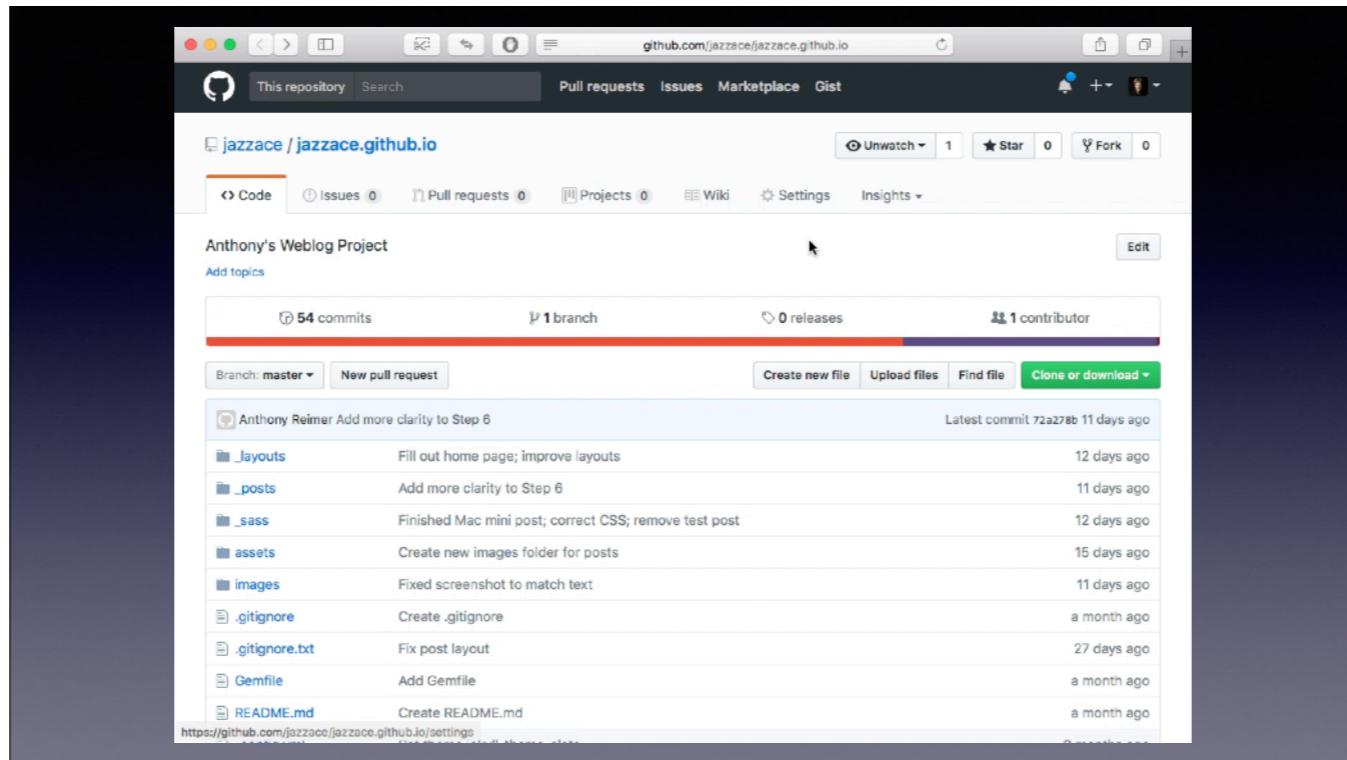
Let's say we've got this. >



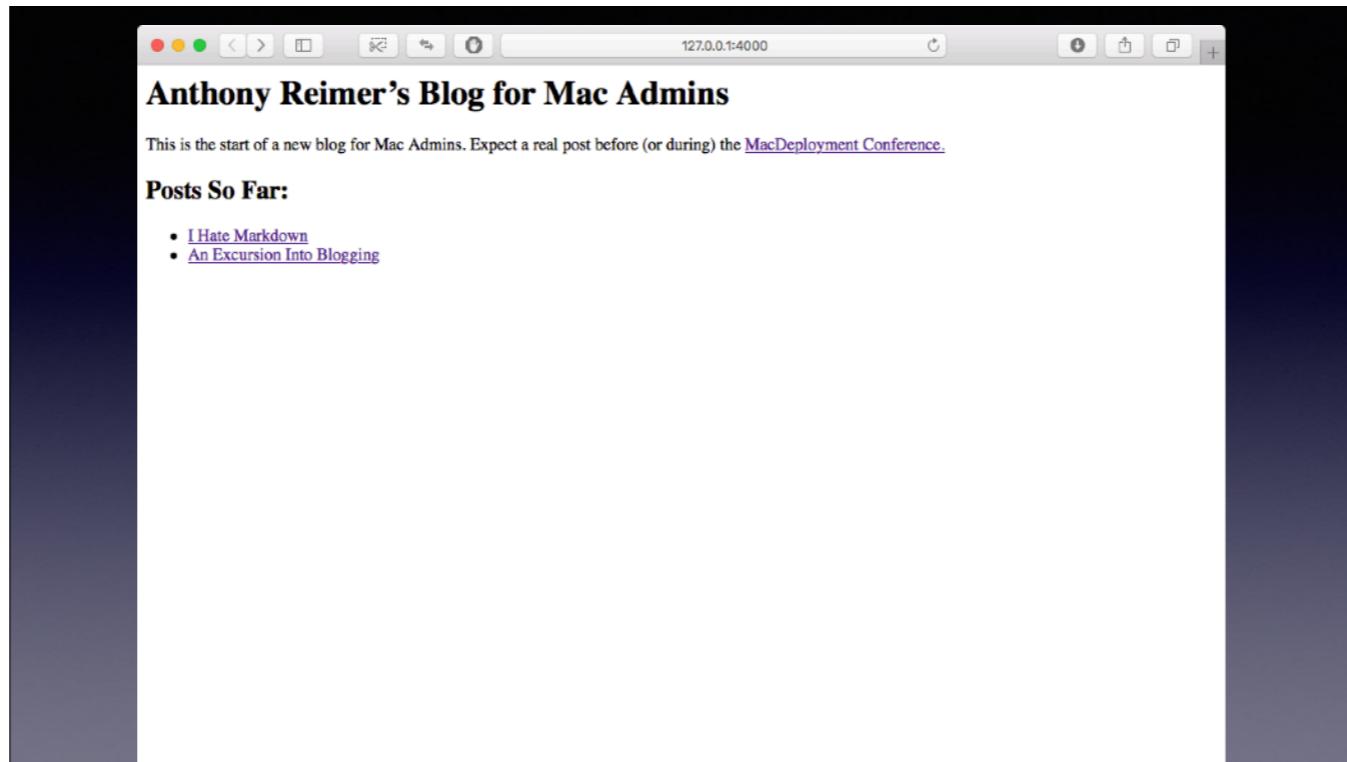
We want to turn it into this. Anyone know how we would do this on a normal web site?

# CSS ❤ Template

Right, Cascading Style Sheets. This is where too much knowledge got me into trouble. I assumed that a theme was just a friendly way of saying CSS. • But it also allows you to use something akin to a mail merge template, where you just create the content for your blog post and signal which type of page you are creating. • So a Jekyll Theme is actually both CSS and a template. Let's go pick a theme for our blog.



In order to do that, we're going to login to GitHub. I've already created the [jazzace.github.io](#) repo, so to select the theme, • [video plays] I click on the Settings tab, scroll down to the options for GitHub Pages, and then select Change Theme. [variety of themes, content stays the same] I'll keep the Slate theme. Only one problem at this point: it's not like CSS where a change to the CSS makes a change to the appearance. >



It's still going to look like this. We need to tell our pages and blog posts to use the template. So let's look under the hood at our web content.

```
<!DOCTYPE html>
<html>
<head>
    <title>The Hello World Blog</title>
    <link rel="stylesheet" type="text/css"
          href="/style.css">
</head>
<body>
    <h1>Hello World</h1>
    <p>I have something to say.</p>
    <p>&copy; 2017 Anthony Reimer</p>
</body>
</html>
```

Here is the absolute simplest HTML page I can write that has the essential elements we need for this discussion and that you can read on screen. It's a slightly more complete version of the Hello World page from before. If I post this on a GitHub Pages site, it will render, even without Jekyll.

## ■ Layout

```
:
: <!DOCTYPE html>
: <html>
:   <head>
:     <title>The Hello World Blog</title>
:     <link rel="stylesheet" type="text/css"
:           href="/style.css">
:   </head>
:   <body>
:     <h1>Hello World</h1>
:     <p>I have something to say.</p>
:     <p>&copy; 2017 Anthony Reimer</p>
:   </body>
: </html>
```

The text in white is going to be the same for every page on our site, or pretty close. So we can extract it and put it into what Jekyll calls a layout.

Here's the markup split into a layout and a page. If you now open `helloworld.html` in a Jekyll-enabled site, it should render exactly like the previous single HTML file. But now I can re-use this layout throughout my site, just writing the body content and the special Front Matter to tell Jekyll which layout to use, the title of the page, and other variables that you can specify using Liquid, the markup you see in this example with the curly brackets (more on that in a moment). Let's look more closely at the simplified page source.

<https://jazzace.github.io/helloworld.html>

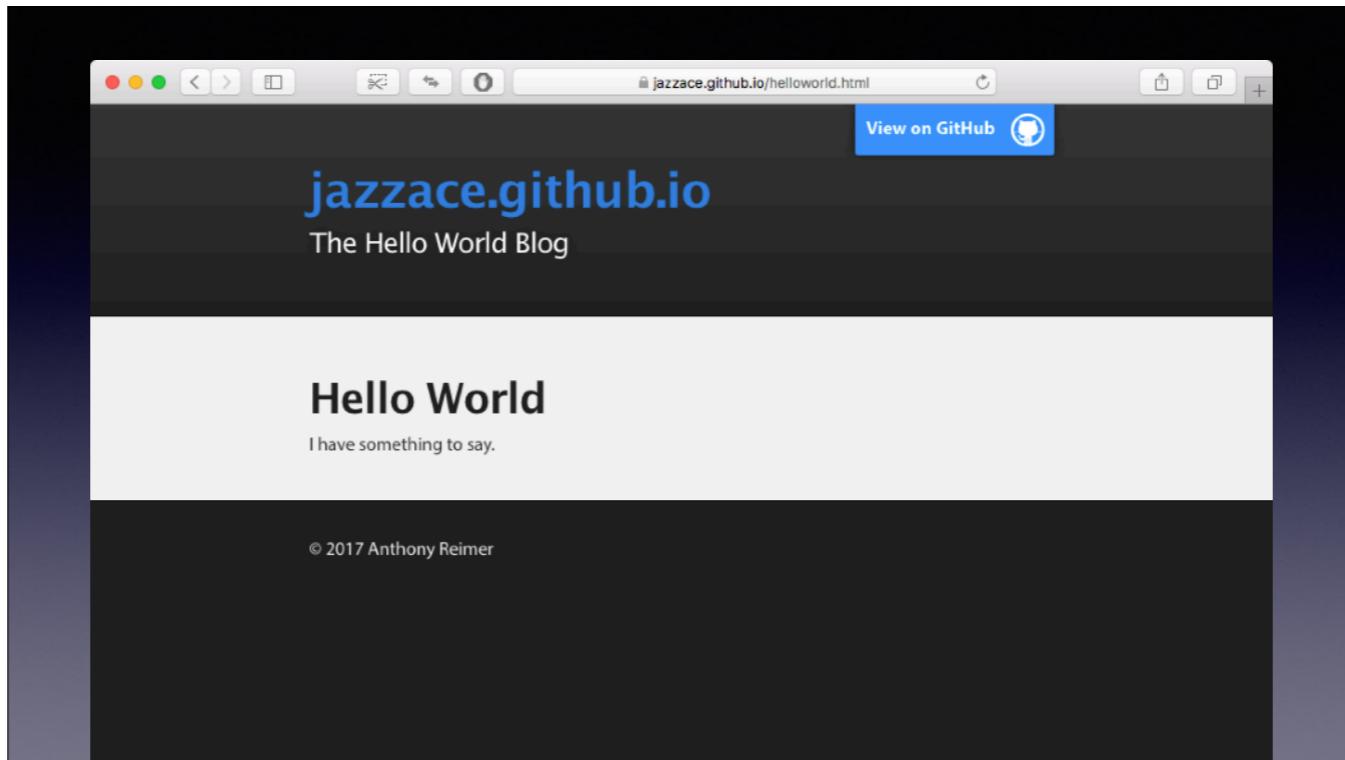
```
---
layout: default
title: The Hello World Blog
---
<h1>Hello World</h1>
<p>I have something to say.</p>
```

Those top 4 lines are the YAML Front Matter. (YAML Ain't Markup Language.) The lines with three dashes that enclose the Front Matter are mandatory. For all the Themes available on GitHub, there will only be a default layout, although you can create your own and even override the default layout if you so choose. The title is the page title — on your home page, that's likely the name of your blog; on a blog post, the name of your post. There can be more items specified, but those two you will want on every post. Of course, if you prefer Markdown, you can get the same result like this: >

## *hello-world.md*

```
---
layout: default
title: The Hello World Blog
---
Hello World
=====
I have something to say.
```

You could also use the Markdown hash syntax for headers as well, but you get the idea. Since Jekyll has to process the file to bring in the items from the layout, it's not a big deal for it to convert Markdown to HTML as well.



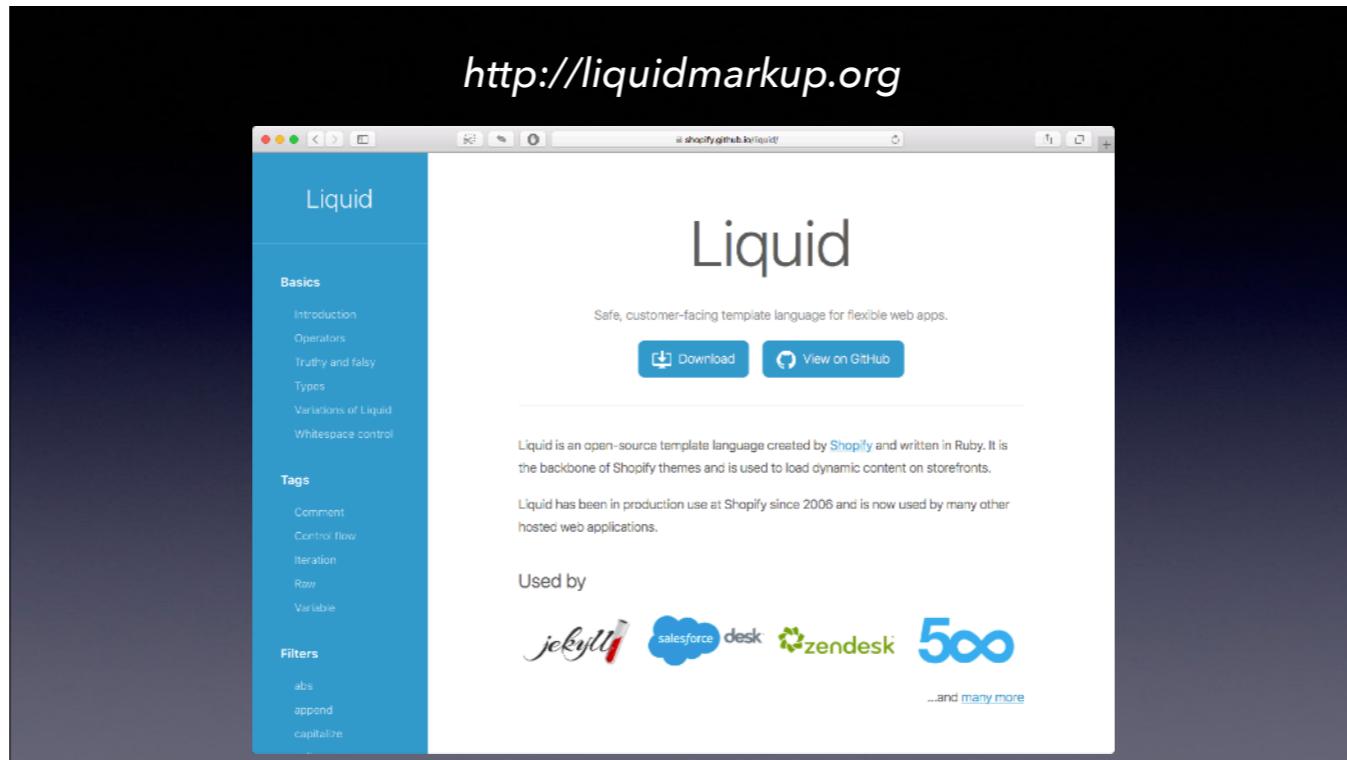
So if we did everything right, we hope to see something like this. Could all of you write a blog post in this simplified format? The good news is that whichever Theme you choose, you'll get a great base of styling with CSS and a default layout provided for you, which you can modify to your heart's content. For my site, I made one layout for my home page and another for my posts.



Concept 4

## Liquid Objects, Tags, and Filters

At this point, I've glossed over some details in order to give you the basics of writing your first blog posts. So let me talk more about Liquid >



Liquid is a template language developed by Shopify from my nation's capital, Ottawa. It's open source. Jekyll leverages objects, tags, and filters to do the magic I was showing with the layouts.

Static Web Site generator  
Themes (via include)  
“Blog aware” (Liquid)  
Add to any web site (Ruby)



Let's go back to what I said earlier about Jekyll and talk about the relationship between themes and Liquid.



From what I showed previously, I found out that I could use variables and conditionals of a sort using Liquid in my markup that would show a list of blog posts without any Jekyll theme. But you can also use a Jekyll theme without using any Liquid; those home pages for GitHub projects I showed earlier are an example of that. We want both of these functions for our blog.

*http://jekyllrb.com/docs/posts/*

```
<ul>
  {% for post in site.posts %}
    <li>
      <a href="{{ post.url }}>{{ post.title }}</a>
    </li>
  {% endfor %}
</ul>
```

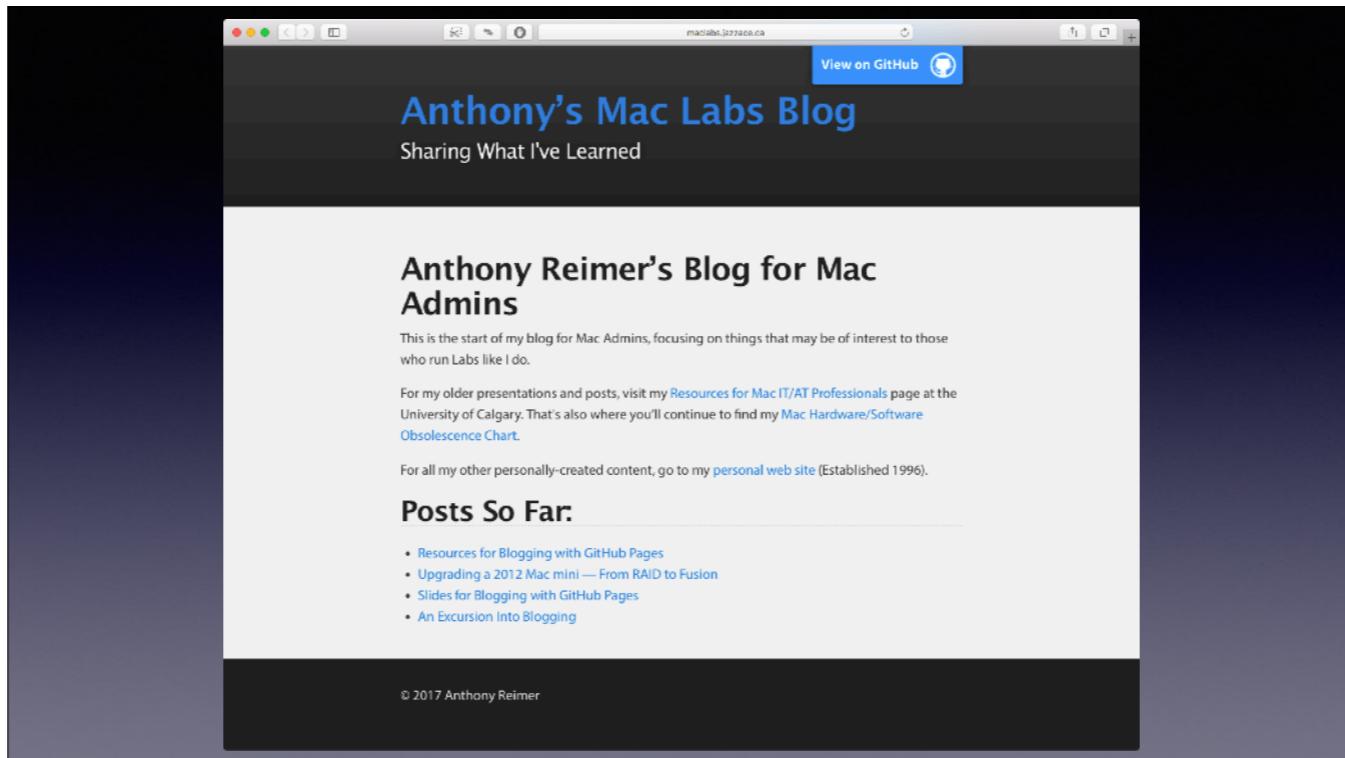
Here's a nice snippet of markup provided in the Jekyll docs that you can insert in any layout (or page, for that matter) to list links to all of your posts from newest to oldest. Let's look at how it's constructed: >

```
<ul>
  {% for post in site.posts %}
    <li>
      <a href="{{ post.url }}>{{ post.title }}</a>
    </li>
  {% endfor %}
</ul>
```

Tags

Objects

Red: Objects, a kind of variable (OO); Green: Tags for conditionals. **site.posts** holds the entire list of posts, so the **for** loop extracts the posts one by one.



[auto-animate] So here's an example of how it would look at the bottom of the home page on my blog.

```
<!DOCTYPE html>
<html>
<head>
  <title>{{ site.title }}</title>
  <link rel="stylesheet" type="text/css"
    href="/style.css">
</head>
<body>
  {{ content }}
  <p>&copy; 2017 Anthony Reimer</p>
</body>
</html>
```

*\_layouts/post.html*

The other thing I wanted to do that is common on blogs is to show the date as part of the post. Here's the layout from earlier. I can leverage Liquid to do this automatically, since the date is part of the post. >

```
<!DOCTYPE html>
<html>
<head>
  <title>{{ site.title }}</title>
  <link rel="stylesheet" type="text/css"
    href="/style.css">
</head>
<body>
  <p>Posted {{ page.date | date: "%Y-%m-%d" }}</p>
  {{ content }}
  <p>&copy; 2017 Anthony Reimer</p>
</body>
</html>
```

*Reference:*

<http://alanwsmith.com/jekyll-liquid-date-formatting-examples>

By adding this one line of markup, the date will be displayed at the top of the page, just before the content I added in the post file. Let's exam the Liquid markup. >

```
<!DOCTYPE html>
<html>
<head>
    <title>{{ site.title }}</title>
    <link rel="stylesheet" type="text/css"
        href="/style.css">
</head>
<body>
    <p>Posted {{ page.date | date: "%Y-%m-%d" }}</p>
    {{ content }}
    <p>&copy; 2017 Anthony Reimer</p>
</body>
</html>
```

Objects

Filter

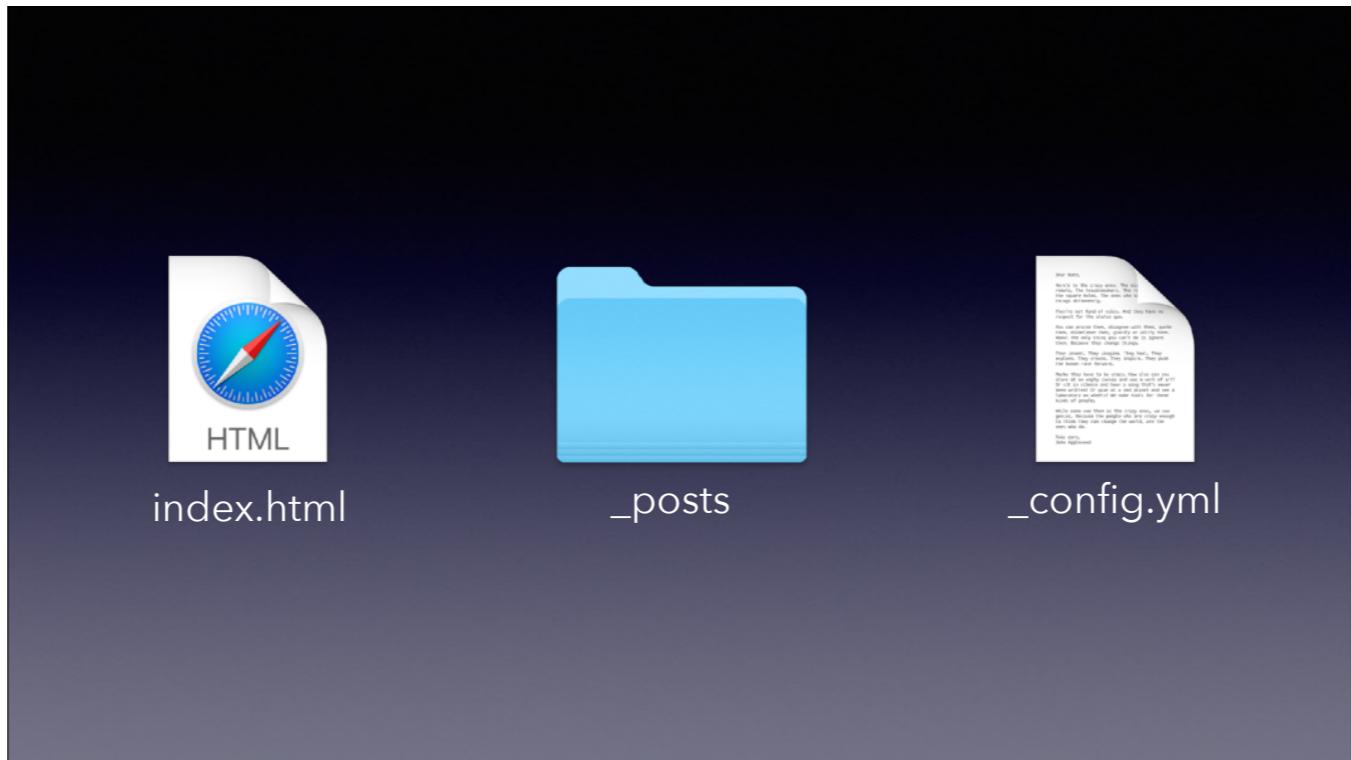
Page.date is another object that you can use. What comes after the vertical bar is what's called a filter. In this example, I've opted for the ISO 8601 date format. [Own blog: Year, Month in words, date. Also, can have more than one filter, will be processed left to right.]



Concept 5

## Site-wide Settings and Resources

A couple of final details.



Remember this structure I showed earlier? Let's look at the `_config.yml` file.

```
theme: jekyll-theme-slate
```



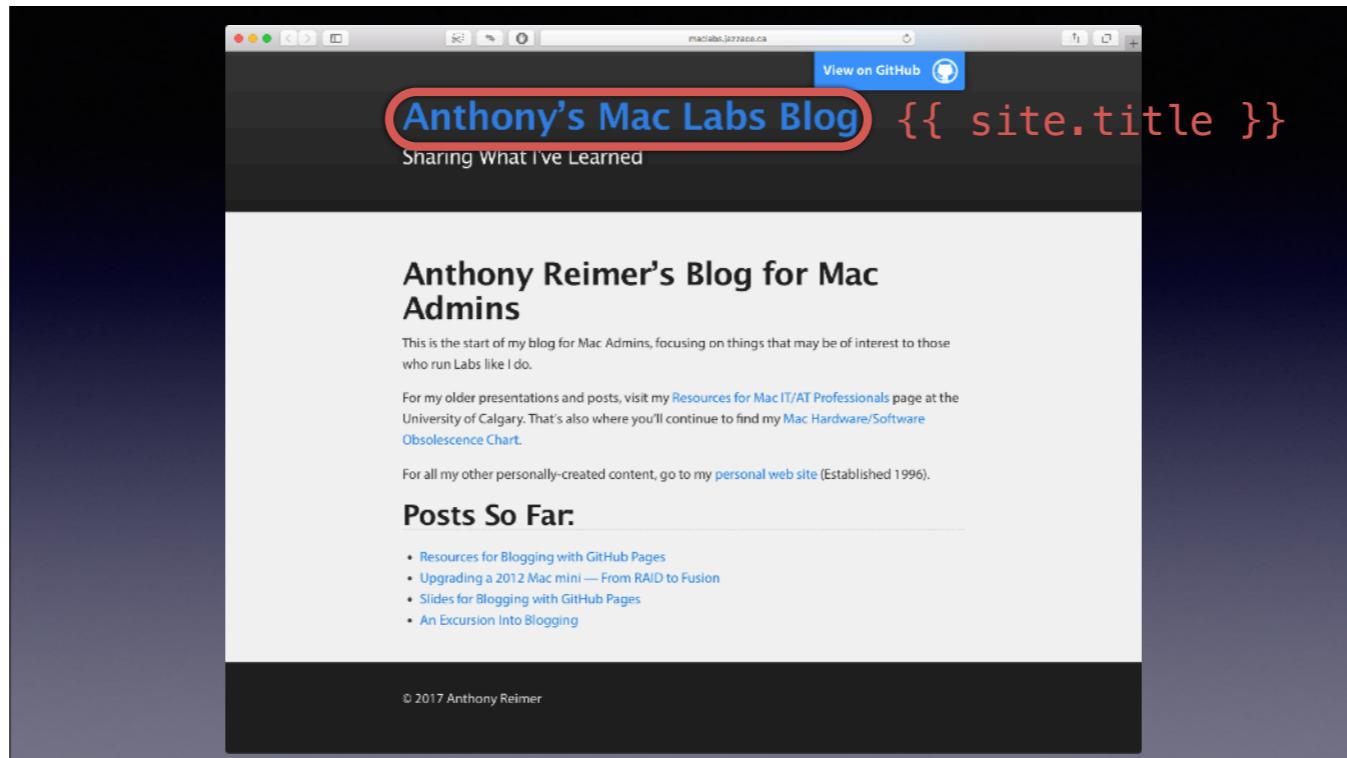
```
_config.yml
```

When you established the theme via the web interface on GitHub, it wrote this line to `_config.yml`. That's how it knows to reference your theme. But this is also a place where we can put other site-wide variables that can then be referenced by object.

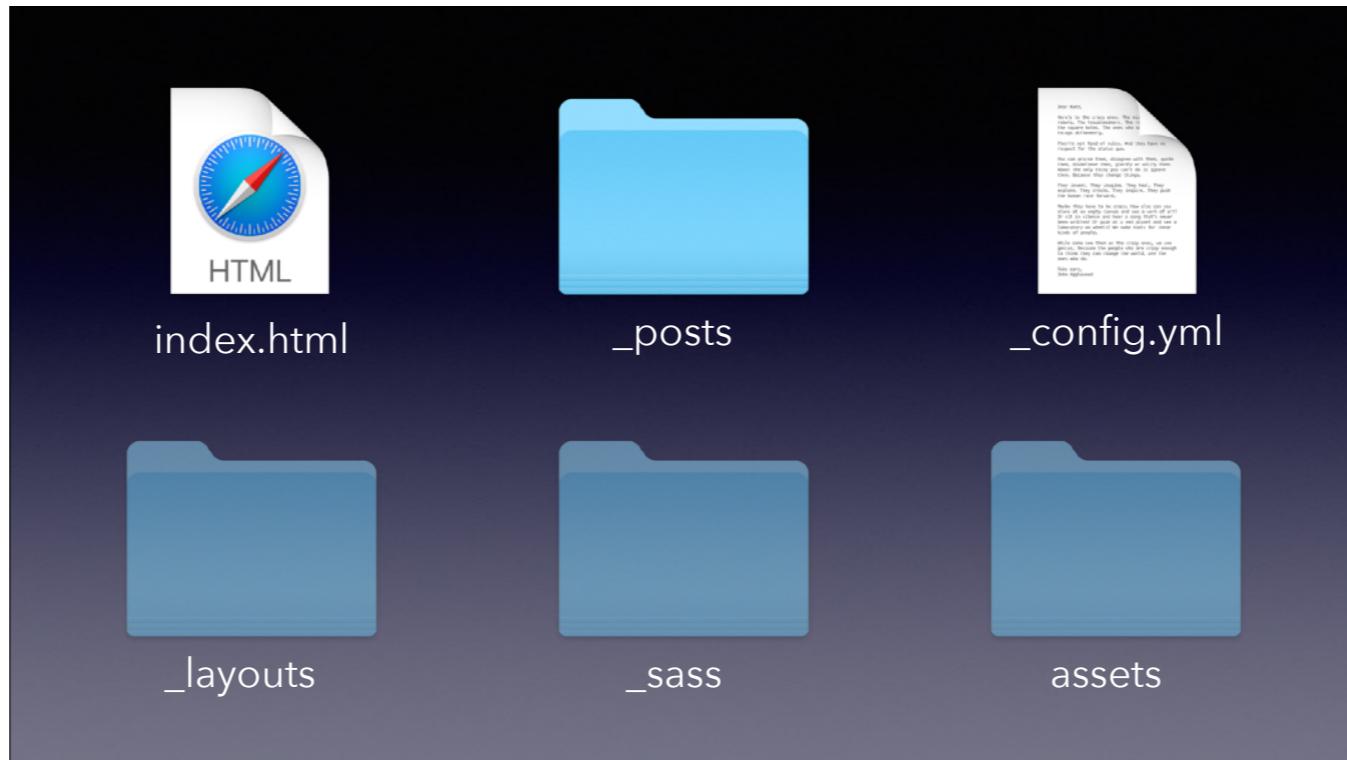


\_config.yml

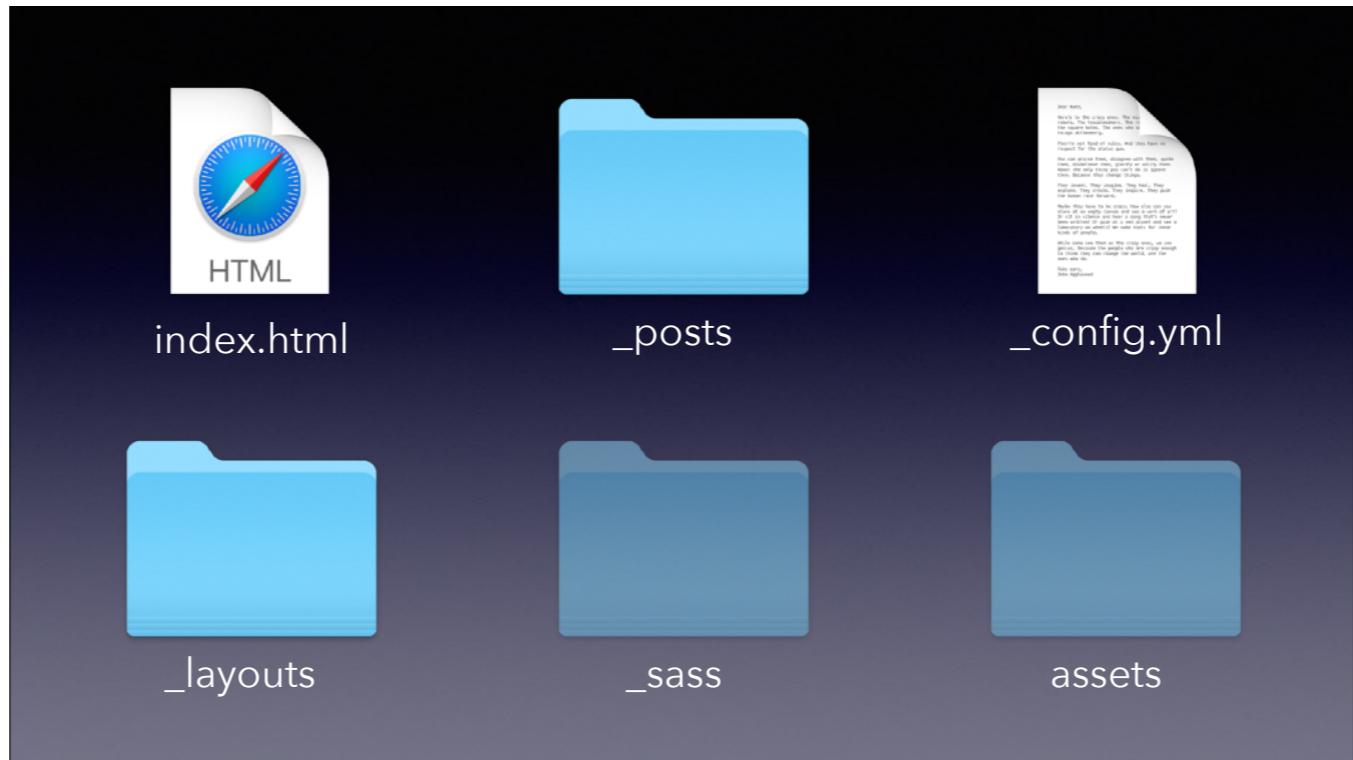
Using my site's theme, the term is "title." In some themes, it's "name". So, if I have a layout that asks for the object site.title, it will insert "Anthony's Mac Labs Blog". >



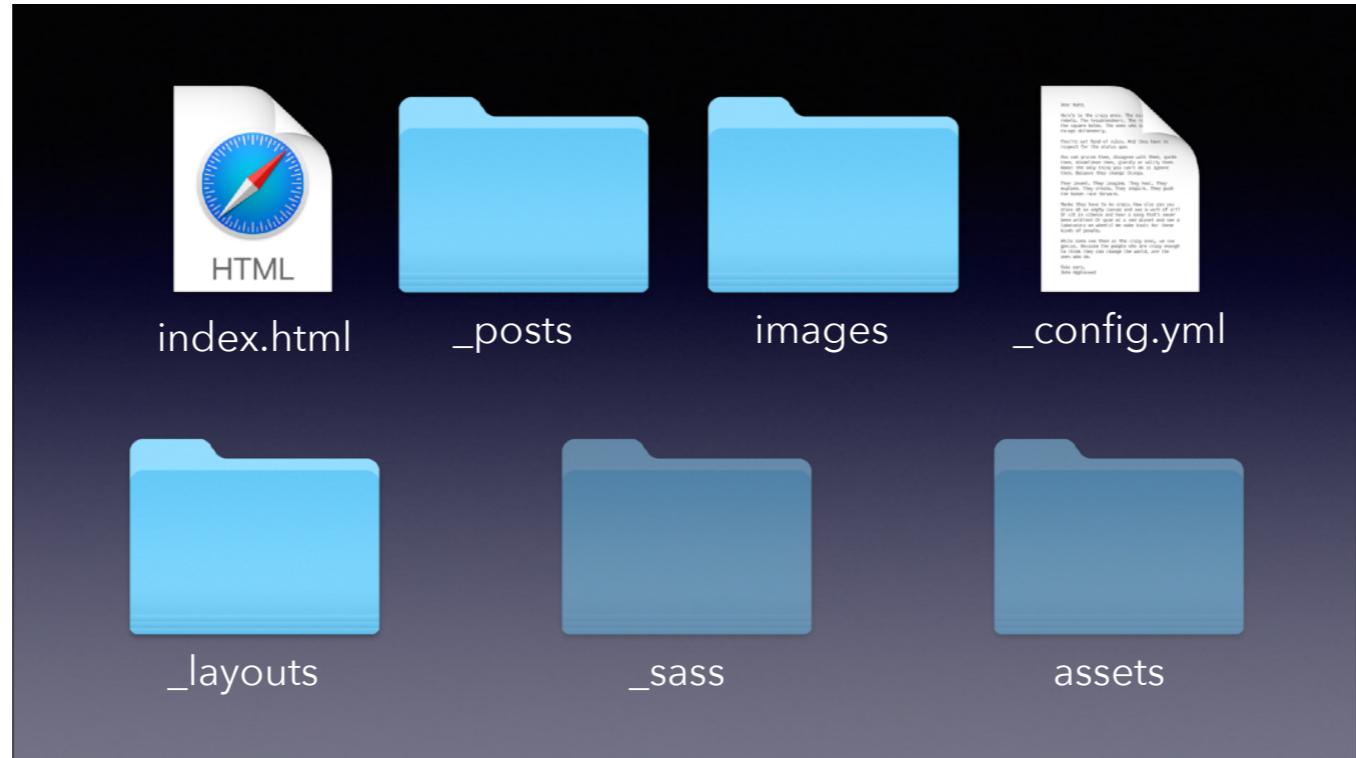
If I don't put anything in there, most themes will be smart enough to default to the repo name. That's why you saw [jazzace.github.io](https://jazzace.github.io) on some of the earlier screenshots; I hadn't set that value in the config file yet.



The other thing I want to mention is overriding aspects of the theme. Every theme comes with a bunch of goodies for free: CSS, layouts, images for display.  
• If you wish, you can override or add to what the theme already has by creating one or more of those folders in your own repo. It will then use what you have supplied rather than the one included in the theme. >



The most common one you'll override is the `_layouts` folder. For my blog, since I knew HTML, I ripped out a bunch of complexity in the default layout and added a layout just for posts. • The other thing you'll probably need is a place to put those screenshots you will inevitably include in your blog posts. >



You can create any folder at the root level of the drive, and as long as you're not using one of the directory names needed for the theme, Jekyll will know that it should be included without processing. When you give the path to those images in your markup, just start the path with a slash so that it knows to look at the root of the site.



## Resources

- <http://maclabs.jazzace.ca/2017/07/05/resources-for-blogging-with-github-pages.html>

I said there was a bunch of documentation, so I've laid that out on my blog, naturally.

# Sharing Code

```
#!/usr/bin/python
"""
CLI Application to streamline the creation of PKGInfo files
for printer deployment in Munki.

Created by Tim Schutt for Syracuse University, 2014 - tschutt@syr.edu
Bug squashing assistance from Vaughn Miller

Much code reused from Printer PKG deploy scripts by:
Walter Meyer, SUNY Purchase, 2008
Nick McSpadden, 2013
"""

import os, sys, subprocess, shlex, string, re, plistlib
from optparse import Option
dirname,filename = os.path.split(os.path.abspath(__file__))
os.chdir(dirname)

## Modify the following to match your environment.
## These are used to set driver dependencies.
## The dictionary is configured as "Human readable":'munki pkgname'
driverCollection = {'Hewlett Packard':'HewlettPackardPrinterDrivers', \
                    'Canon - Commercial Copiers':'Canon_MP_II_Installer', \
                    'Canon - Consumer Printers':'CanonPrinterDrivers', \
                    'Lexmark':'LexmarkPrinterDrivers', \
                    'Epson':'EPSONPrinterDrivers'}

## Enter an example of your house naming convention
nameExample = "printer-as-psy-hp-e551-430hh-grq83"
```

# Sharing Code

- Collect scripts
- Public collection
- Anonymize

Many of us would not call ourselves programmers, but find ourselves needing to write scripts to solve problems. Having a collection or repository of scripts will help you write future scripts.

Why not make it a public collection? If it is helpful to you, why would it not be helpful to others?

Caveat – may need to sanitize your scripts. Be careful with institutional information. Replace real server names with example names. Include comments in the code for which items need to be customized per org.

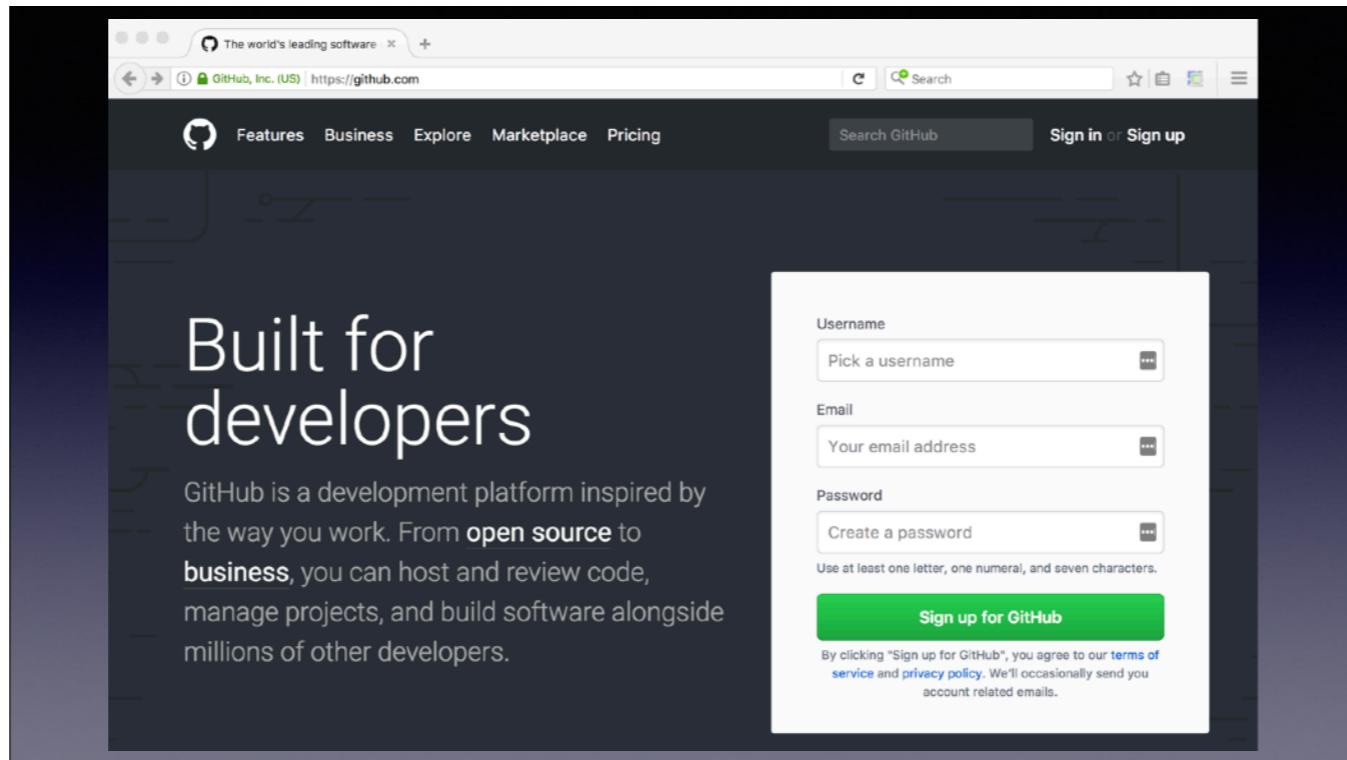
# GitHub

- Social Coding
- Feedback
- Git?

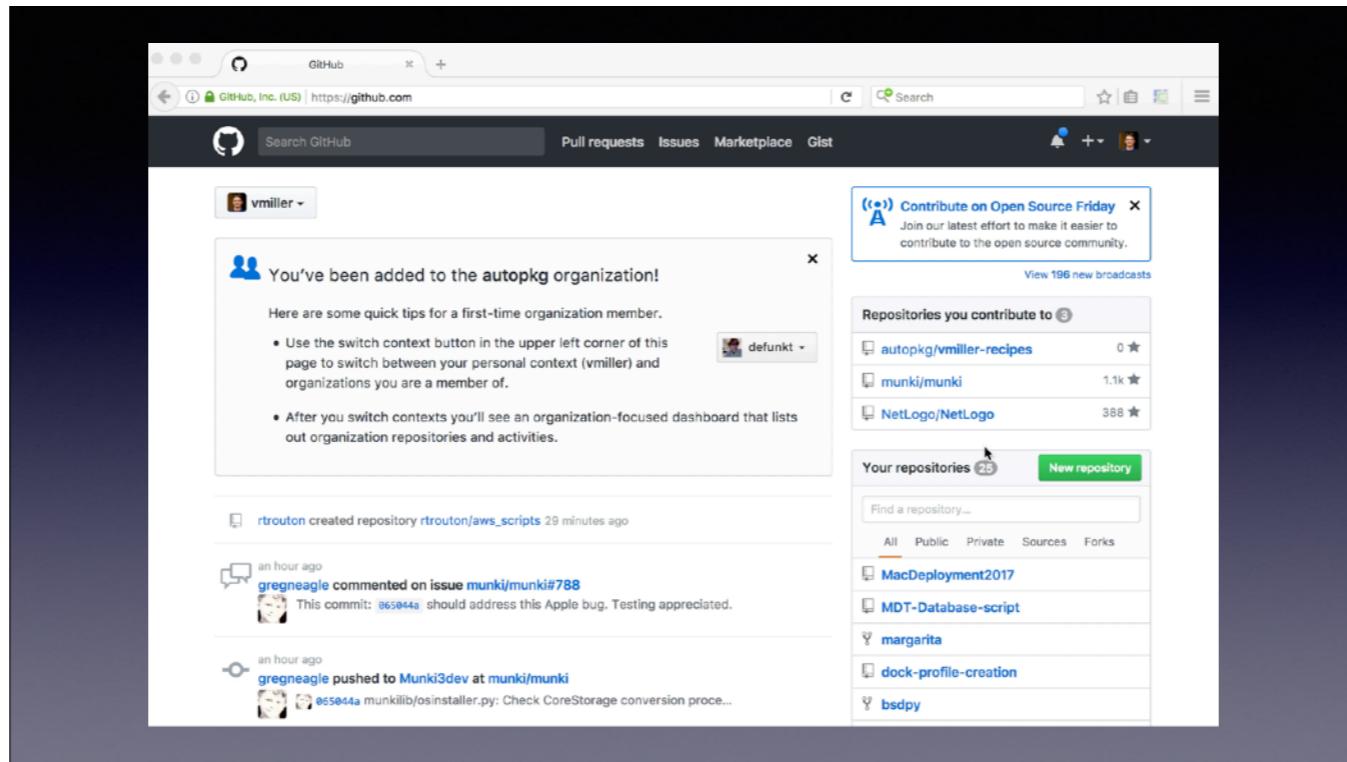
There are other platforms for sharing code, such as GitLab and Bitbucket. GitHub is the most popular for Mac Admins with many sharing scripts.

Get feedback from others on your code. Others may send you improvements for your script.

Don't worry – you do not need to know Git to share via GitHub.



Signing up for a GitHub account is as easy as going to [github.com](https://github.com). Pick a username, enter your email address and set a password.



A quick demo of adding a script to GitHub without using Git



**But...**

**But I don't have time and I  
want a life**

## Time...

- Professional development
- Dedicate a block of time
- Documentation
- Share what you are already doing

Treat community involvement as professional development. Consider dedicating a portion of your work week if your employer will allow.

Save your future self time by documenting

No one expects you to spend all of your free time doing this. Simply making some of what you are already doing public so that others can benefit.



# I have no idea...

- Imposter syndrome
- Teaching == learning
- More or Less

Imposter syndrome is very real for many people.

Trying to teach an idea is a great way to learn more about it. Often we learn just enough to solve a problem. If we try to write or speak about it often we need to add breadth and depth to our knowledge.

No matter how much we learn, there is always someone more knowledgeable. Consider the inverse – there is also someone less knowledgeable.

\_\_\_\_\_ is not my thing!

# Coding is not my thing

- Shell scripts are code
- Complexity optional

You may not identify yourself as a "Programmer" If you have ever written a couple of lines of shell script to solve a problem, you have written code.

Having a collection of scripts/code that has helped you in the past is incredibly useful. I have referred to code from a previous job.



# Writing is not my thing

- Help yourself
- Keep it simple

Remember, do it help yourself. You needn't write a novel. If you can write notes that help you, it is likely someone else will find it helpful too.



# Speaking is not my thing

- Audience != adversary
- Start small
- Safety in numbers?
- Feedback

Remember, the audience wants you to succeed.

Keep in mind you can start small– perhaps presenting at work or at a local meetup.

There can be safety in numbers, consider pairing up with another speaker. For the first time you can pair up with a more experienced speaker.

There is a skill to handling feedback. It may be wise to throw out both the most positive and the most negative response and take something constructive out of the middle.

Be willing to change. If a style or subject matter doesn't work for you (maybe it even bombs) try changing it up and trying again.



**Pick what works for  
you!**

You do not need to do any of this, and certainly you do not need to do all of it. If it truly isn't your thing, share one of the other ways. We need not do all – we can pick and choose.

# Resources

- <http://wp.me/p2wRHj-9L>

# Workshop Feedback:

<https://bit.ly/psumac2017-157>



## Anthony Reimer

[maclabs.jazzace.ca](http://maclabs.jazzace.ca)

 [@AnthonyReimer](https://twitter.com/AnthonyReimer)

 [@jazzace](https://github.com/jazzace)

 <https://github.com/jazzace>

## Vaughn Miller

[vaughnemiller.com](http://vaughnemiller.com)

 [@Vaughn\\_E\\_Miller](https://twitter.com/Vaughn_E_Miller)

 [@vmiller](https://github.com/vmiller)

 <https://github.com/vmiller>