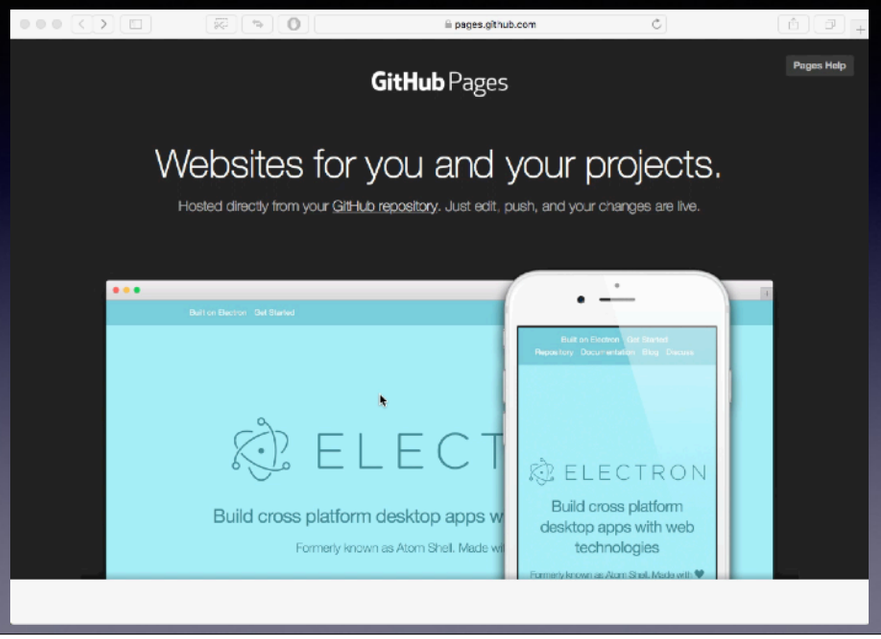
Blogging with GitHub Pages

A Step-by-step Guide on How to Fail

Step 1

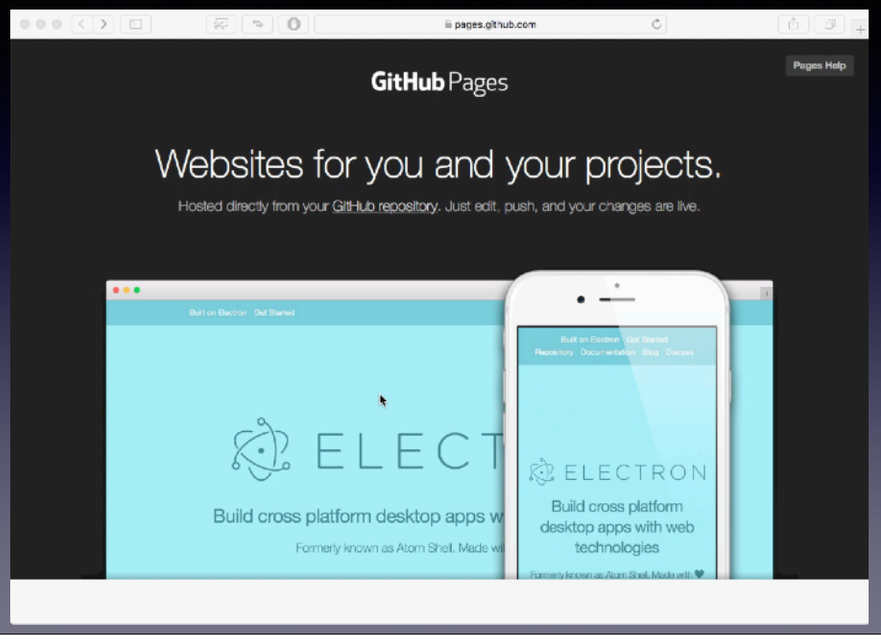
Follow the instructions

https://pages.github.com



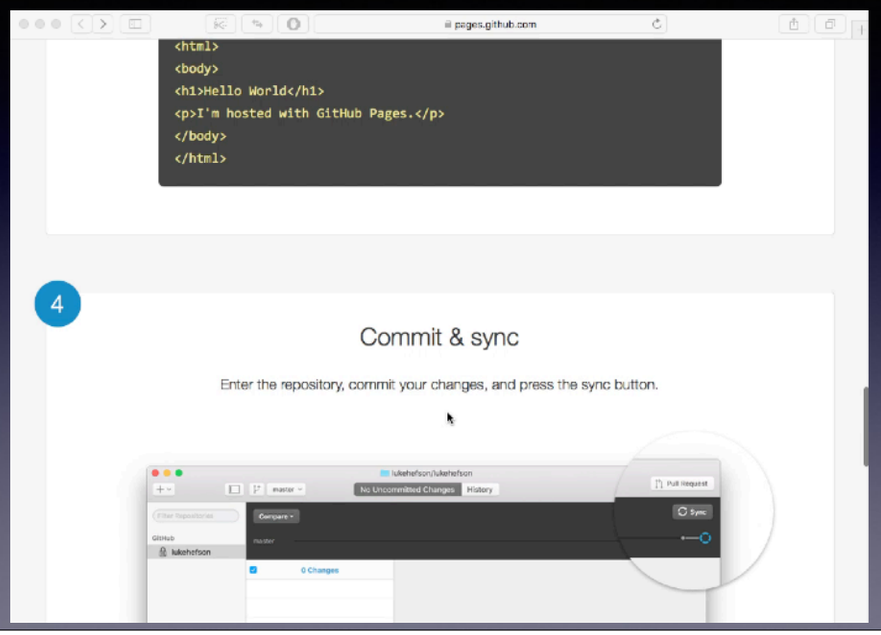
Great, so a simple guide. We create a repository called username.github.io… Hey, a very simple HTML file.

https://pages.github.com



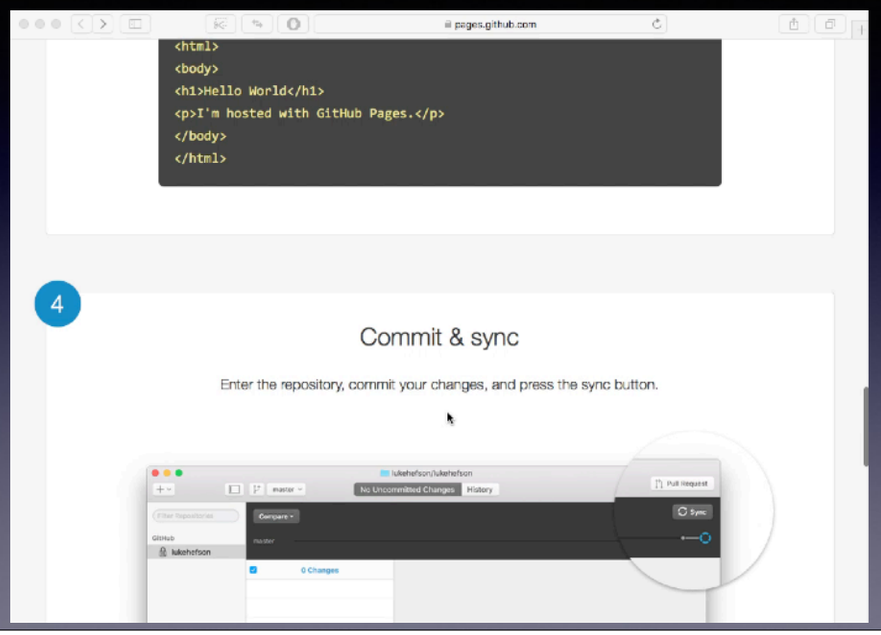
Great, so a simple guide. We create a repository called username.github.io… Hey, a very simple HTML file.

https://pages.github.com

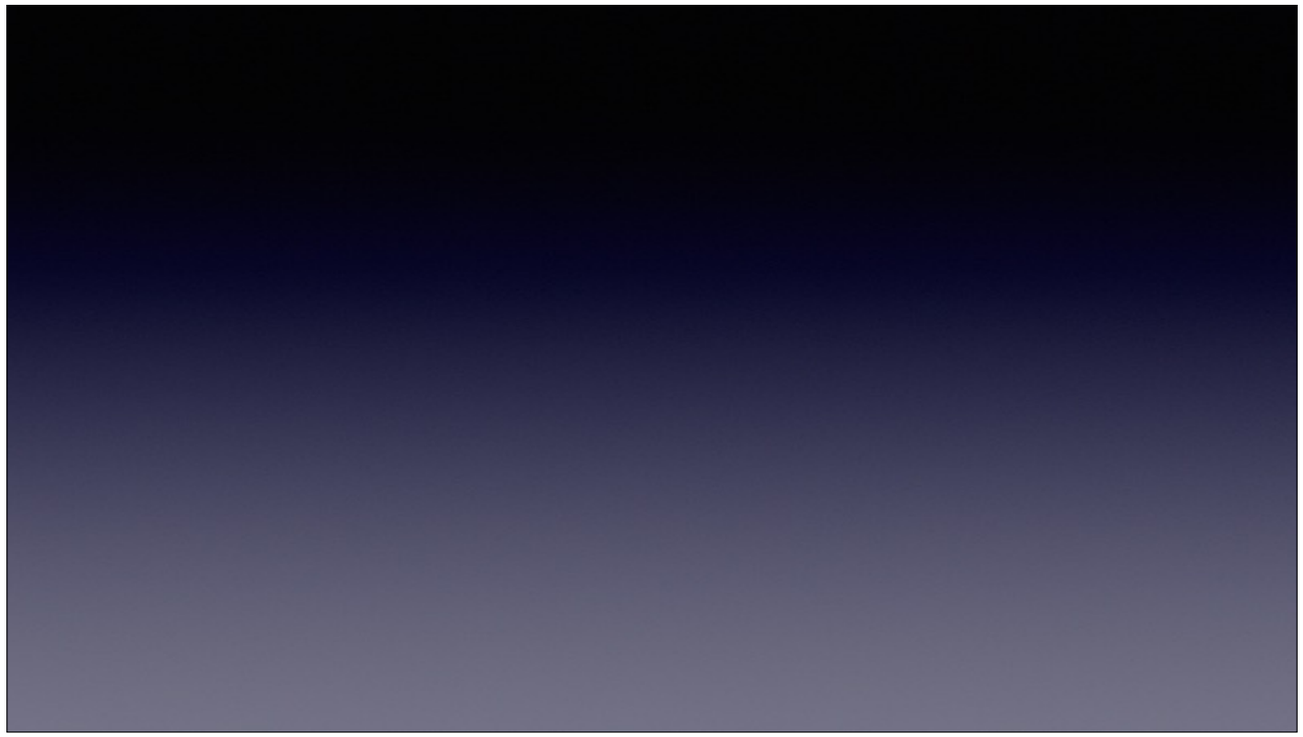


So we sync and our page is posted. …Blogging, that's what we're looking for. …Hey, we can add a theme without using CSS. This Jekyll must be awesome. And, it's integrated. Great! So here's the result when we use their sample HTML. >

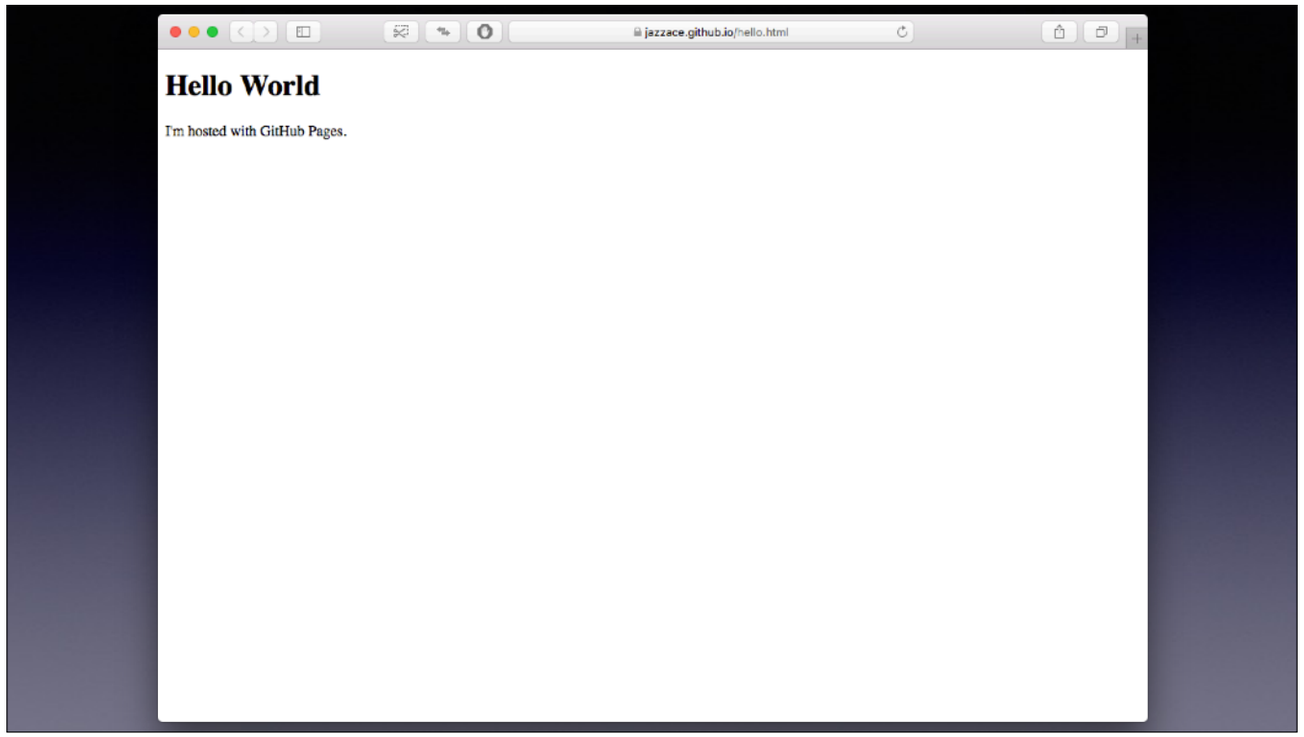
https://pages.github.com



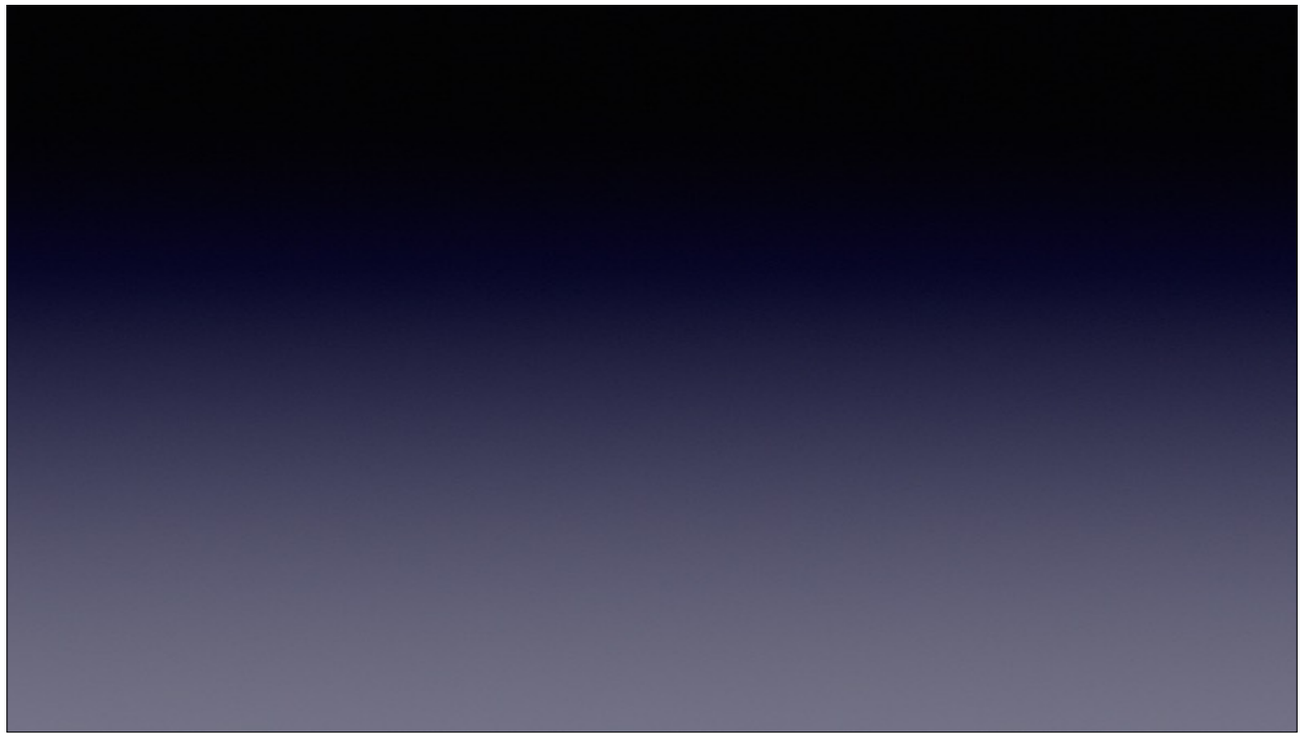
So we sync and our page is posted. …Blogging, that's what we're looking for. …Hey, we can add a theme without using CSS. This Jekyll must be awesome. And, it's integrated. Great! So here's the result when we use their sample HTML. >



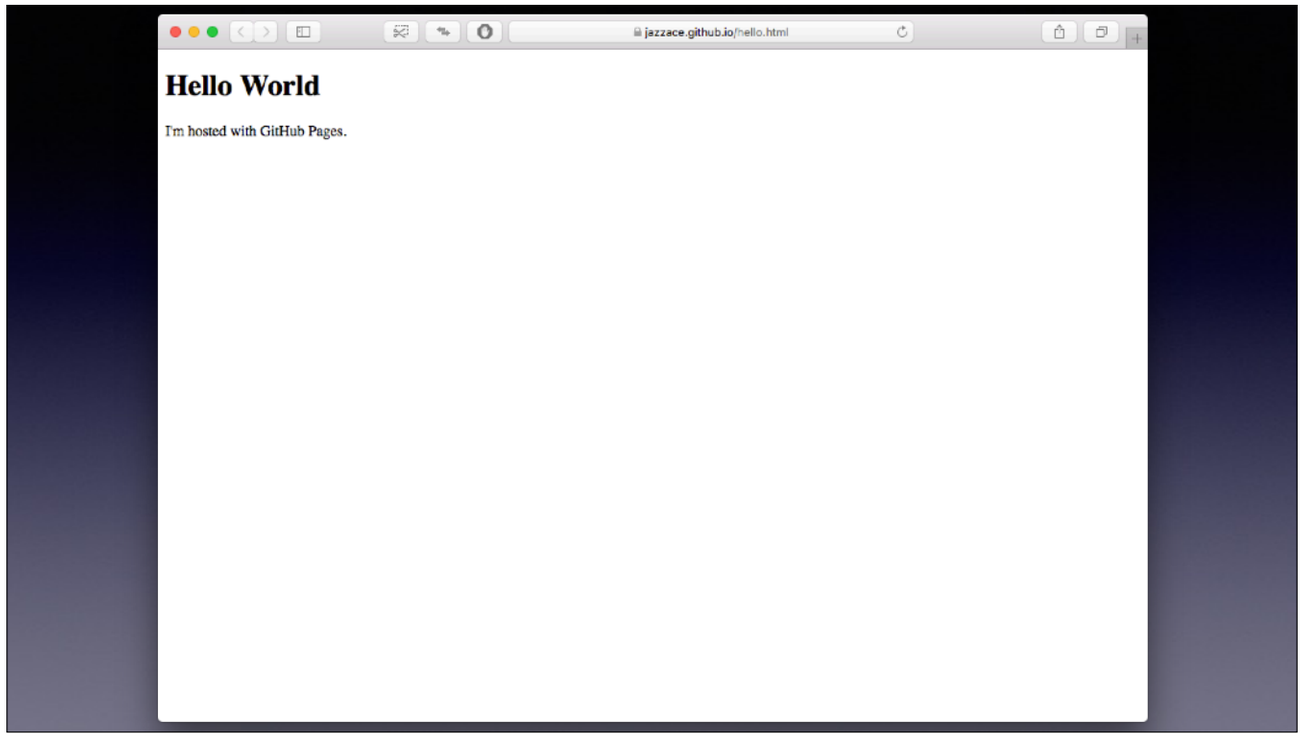
And here’s what it looks like after we follow the instructions to add a Jekyll theme to our site preferences >



And here's what it looks like after we follow the instructions to add a Jekyll theme to our site preferences >



Yup.


jazzace.github.io/hello.html
Hello World
I'm hosted with GitHub Pages.

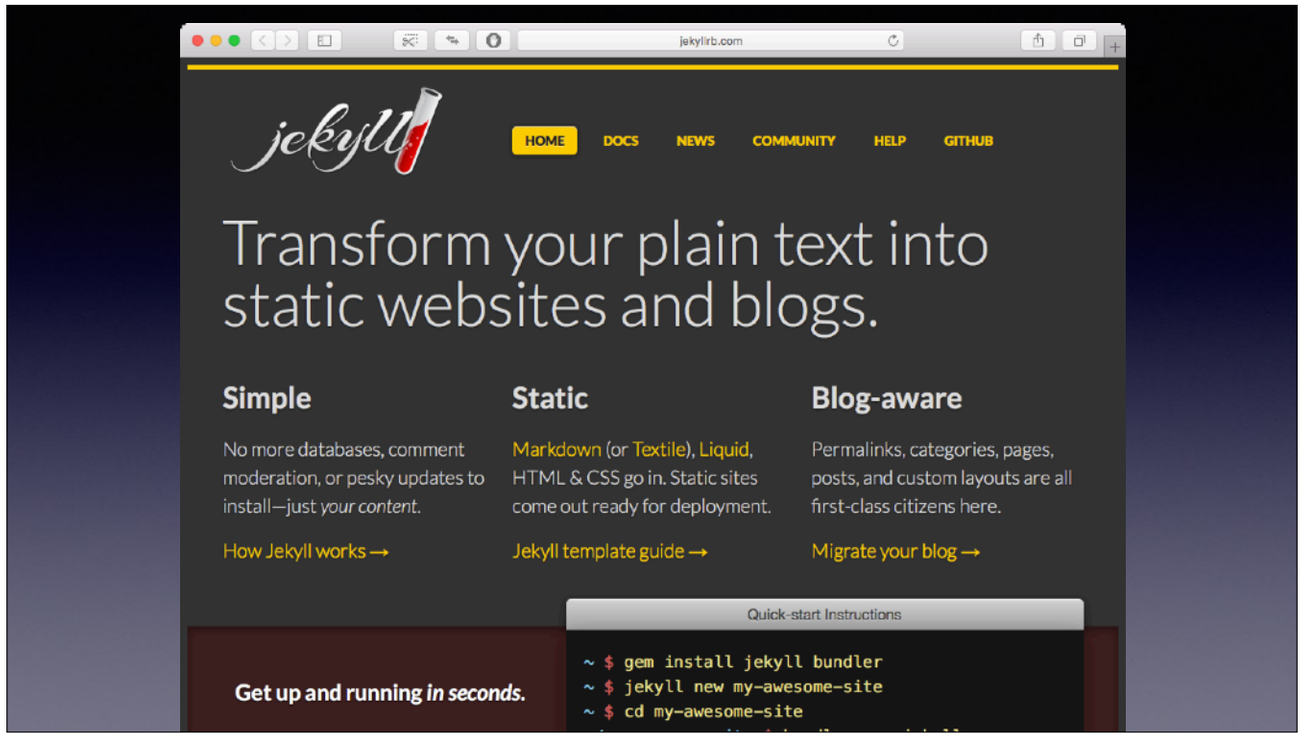
Yup.

Step 2

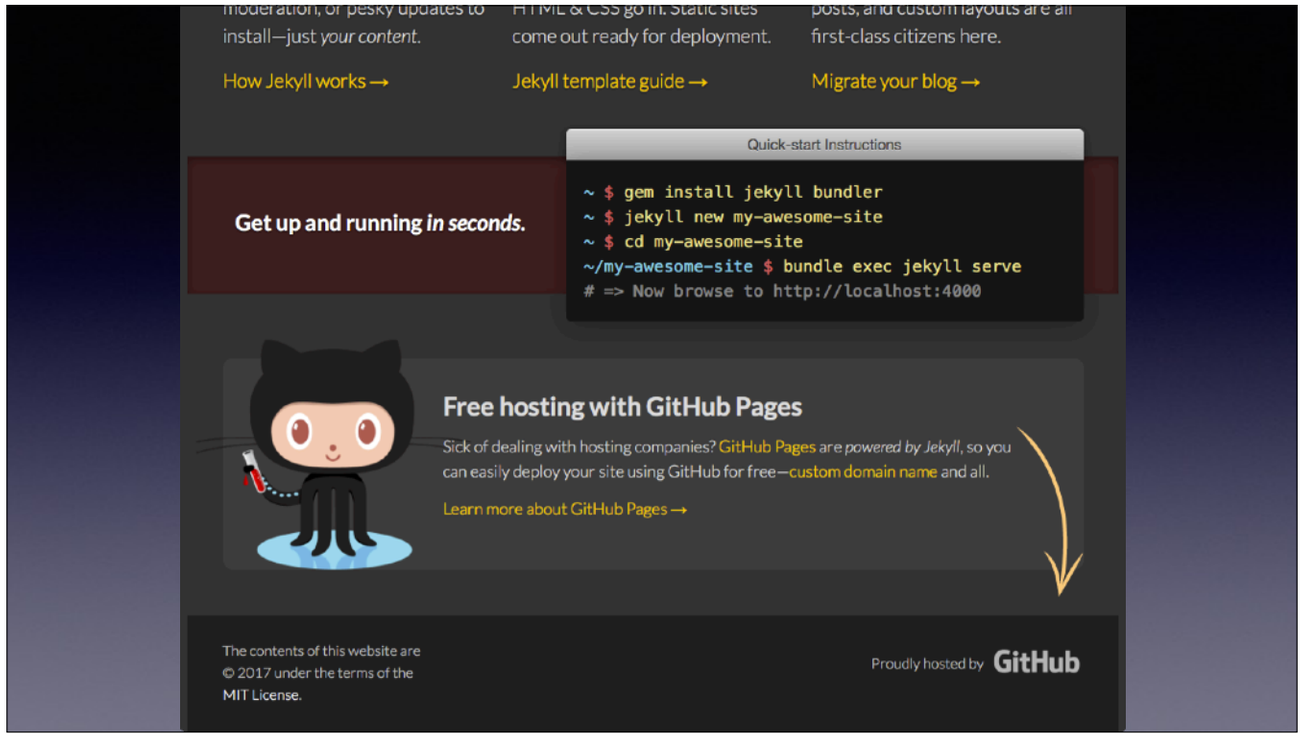
Conflate GitHub Pages and Jekyll

What our first exercise taught me was that if I post the files for a web site in the magical repository of username.github.io — html, css, images — GitHub will serve it. This does not require Jekyll, as I found out later.

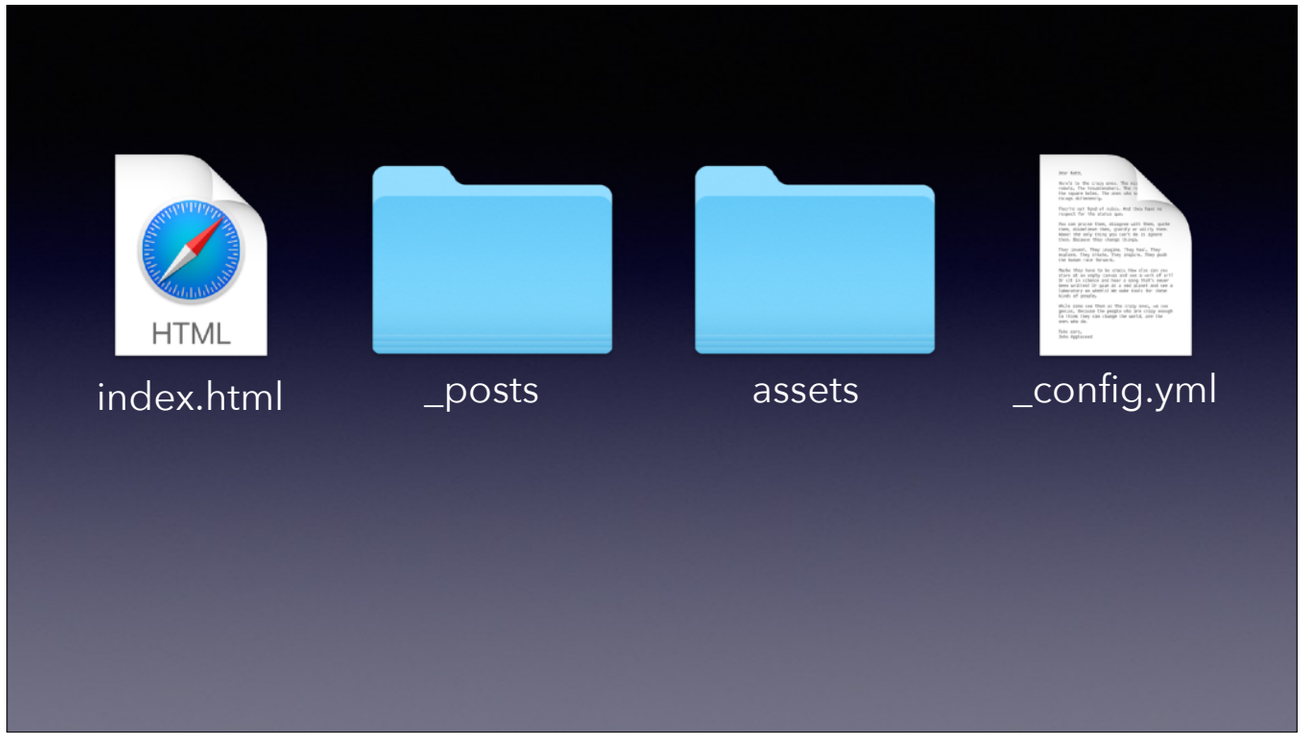
So what is Jekyll? Let’s visit the project web site and find out.



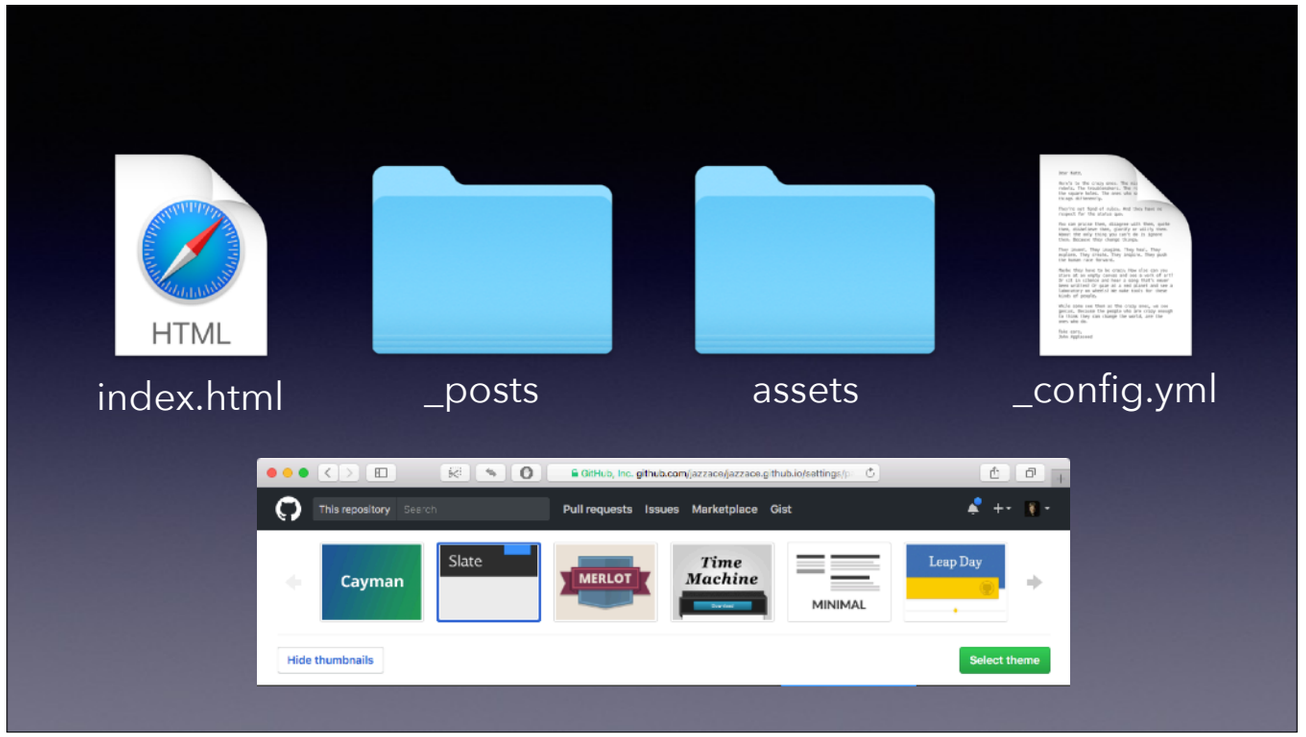
Gee, that sounds like what we're looking for. If we know HTML or even Markdown, it has tools that make sense of our posts and build a static site. We upload, tell it to rebuild, and voilà. • Hey, and it's integrated with GitHub pages. When you commit a post to GitHub, the rebuild request happens automatically, so even easier.



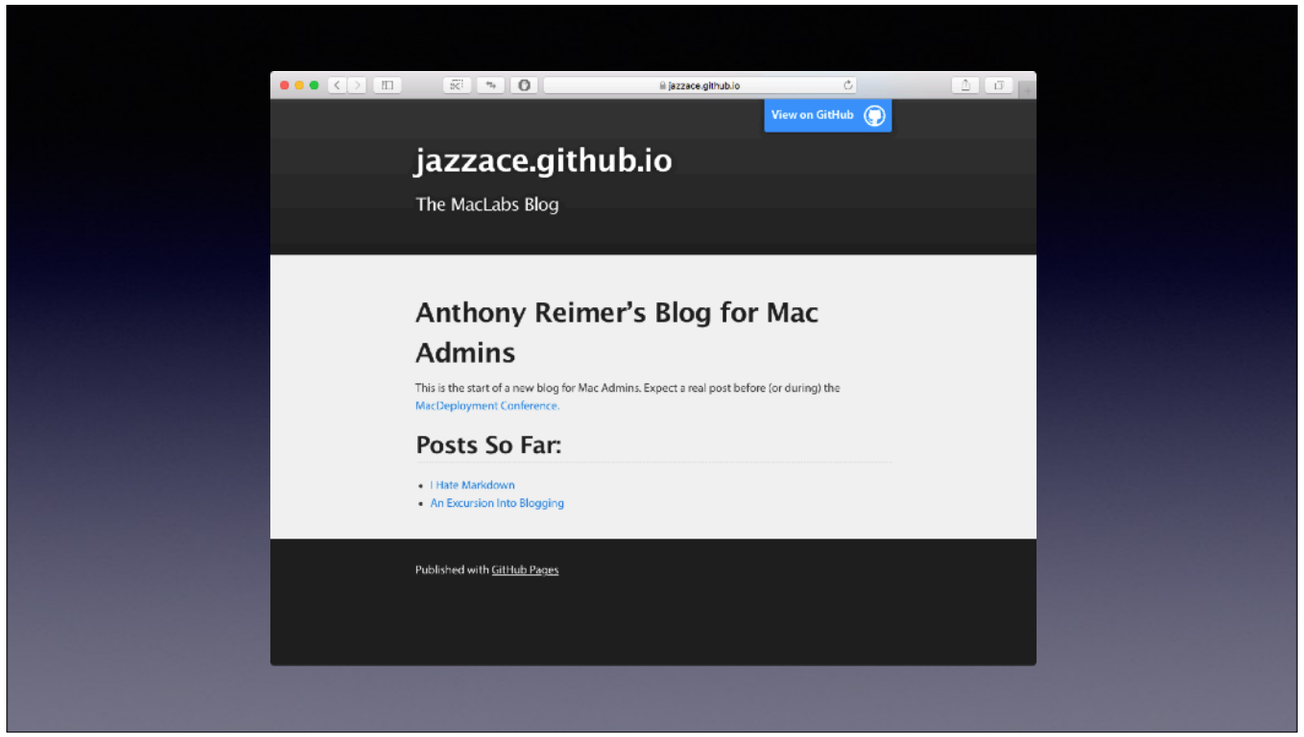
Gee, that sounds like what we're looking for. If we know HTML or even Markdown, it has tools that make sense of our posts and build a static site. We upload, tell it to rebuild, and voilà. • Hey, and it's integrated with GitHub pages. When you commit a post to GitHub, the rebuild request happens automatically, so even easier.



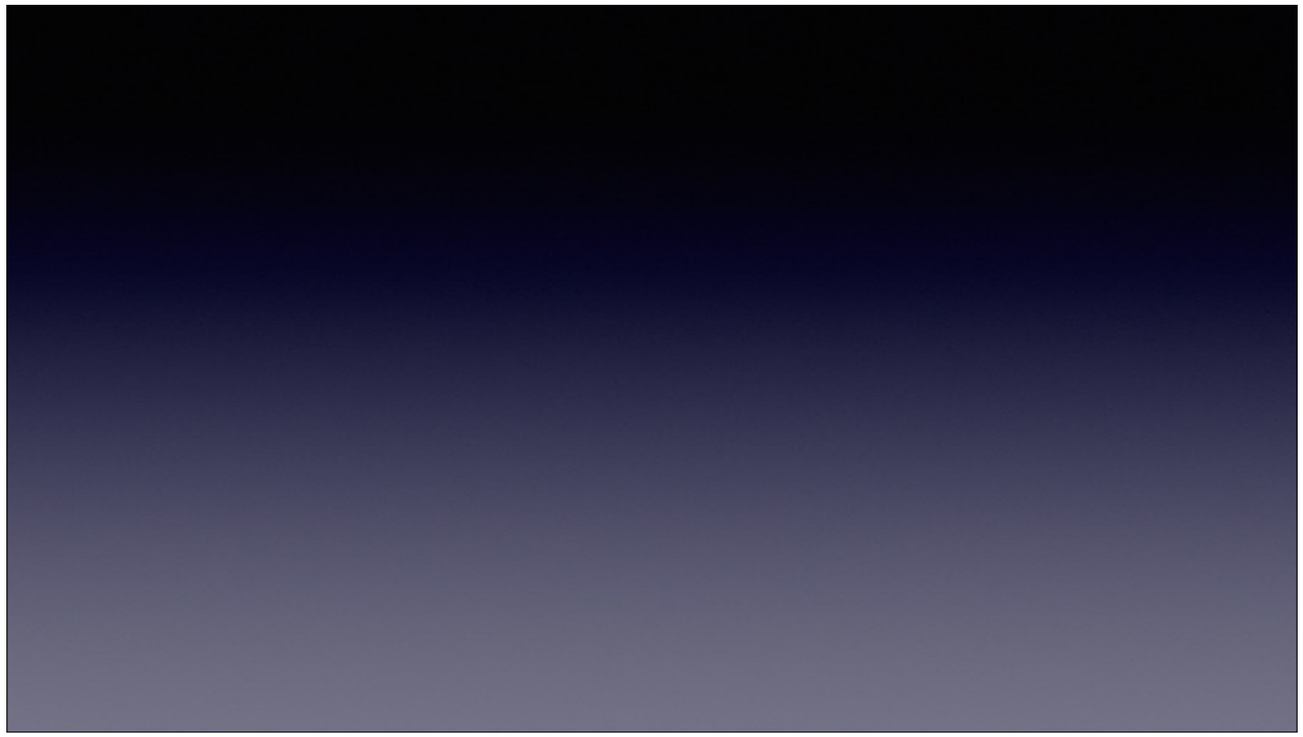
So we supply the markup, adhere to a directory structure with some helper files • pick a theme and we should get something that looks like this… >



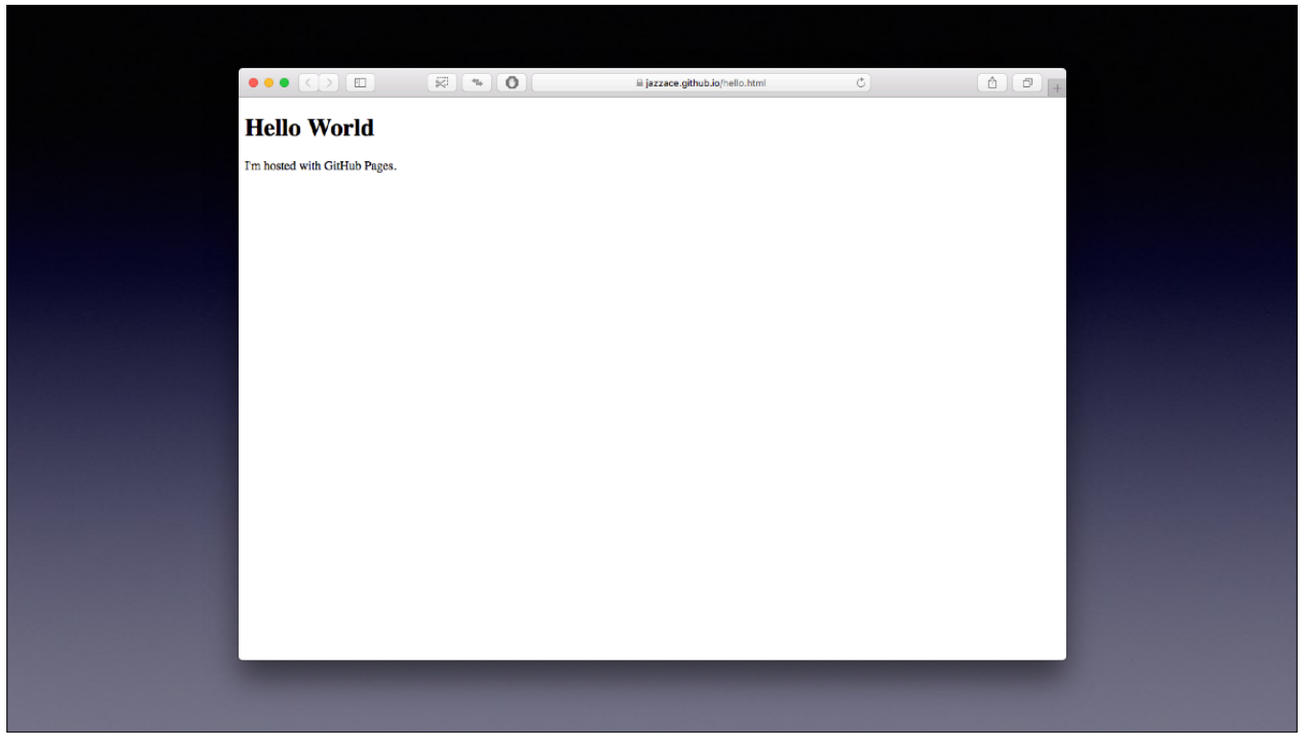
So we supply the markup, adhere to a directory structure with some helper files • pick a theme and we should get something that looks like this… >


jazzace.github.io
View on GitHub
jazzace.github.io
The MacLabs Blog
Anthony Reimer's Blog for Mac Admins
This is the start of a new blog for Mac Admins. Expect a real post before (or during) the MacDeployment Conference.
Posts So Far:
I Hate Markdown
An Excursion Into Blogging
Published with GitHub Pages

[pause, then >]



I’m still getting _this_. Let me skip over a lot of my pain and tell you what I learned.



I’m still getting _this_. Let me skip over a lot of my pain and tell you what I learned.

Static Site Web host

Regular GitHub features
(branching, version control)

Jekyll support on server

Static Web Site generator

Themes (via include)

“Blog aware” (Liquid)

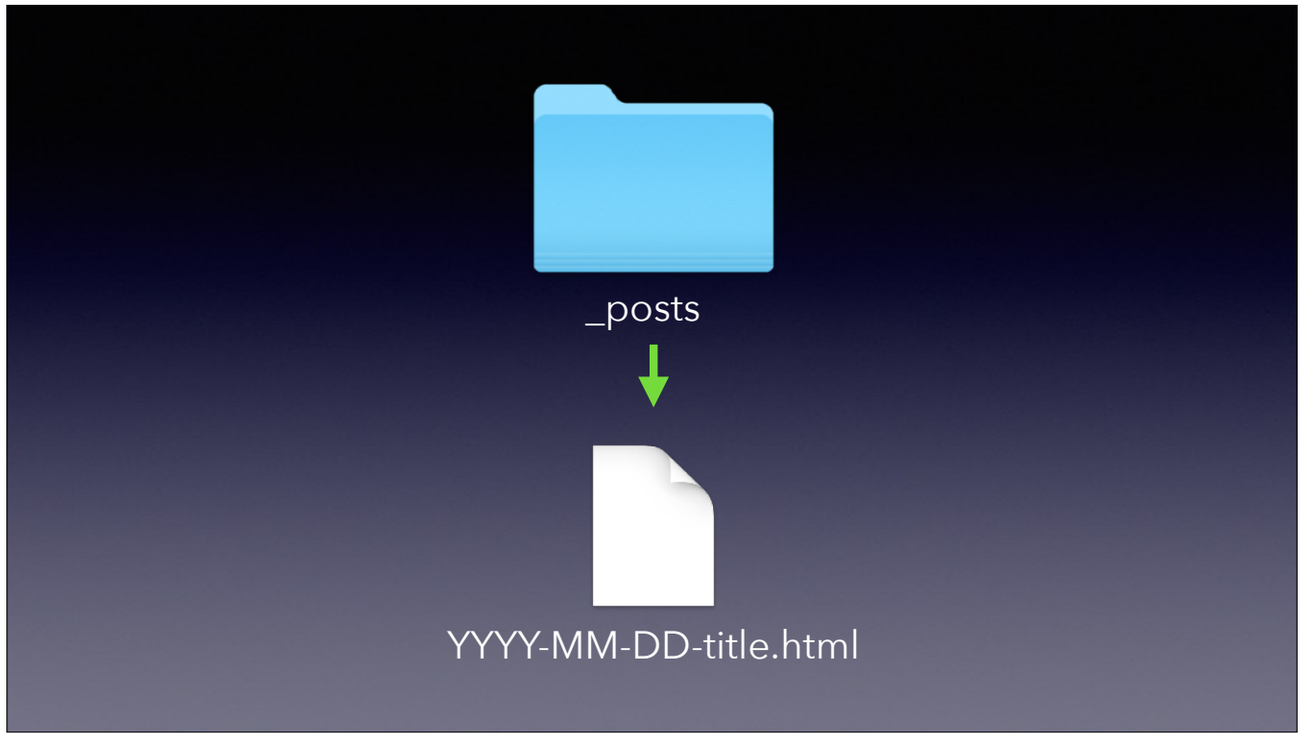
Add to any web site (Ruby)

[go through points]. So my problem is that I haven’t found the glue that joins the two together.

Step 3

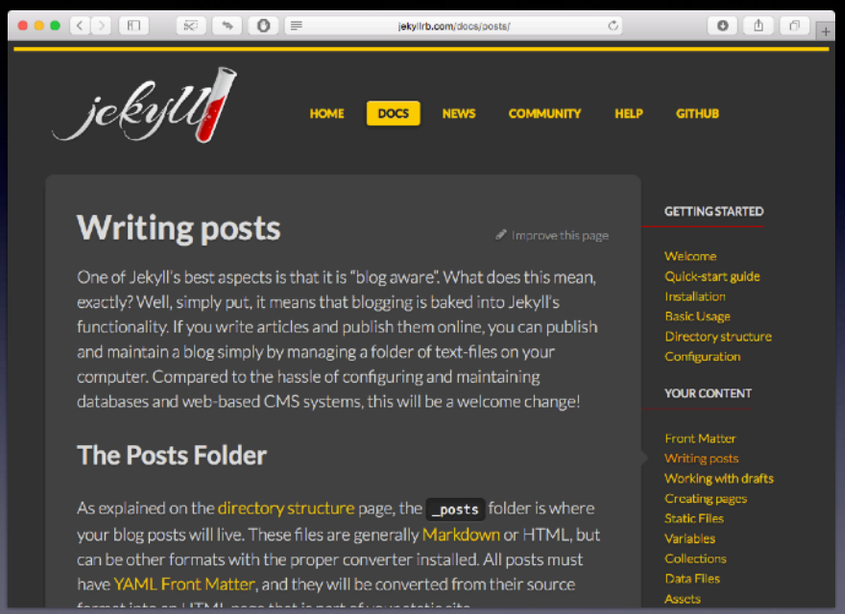
Focus on the Blogging Part

So I thought, hey, maybe it’s only that Hello World page that’s a problem. What if I write a blog post or two to see what happens. This part was not actually a failure, although it felt like it.



All blog posts need to go in the _posts directory and use a particular naming format. If you use Markdown, you use the .md extension instead of .html.

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

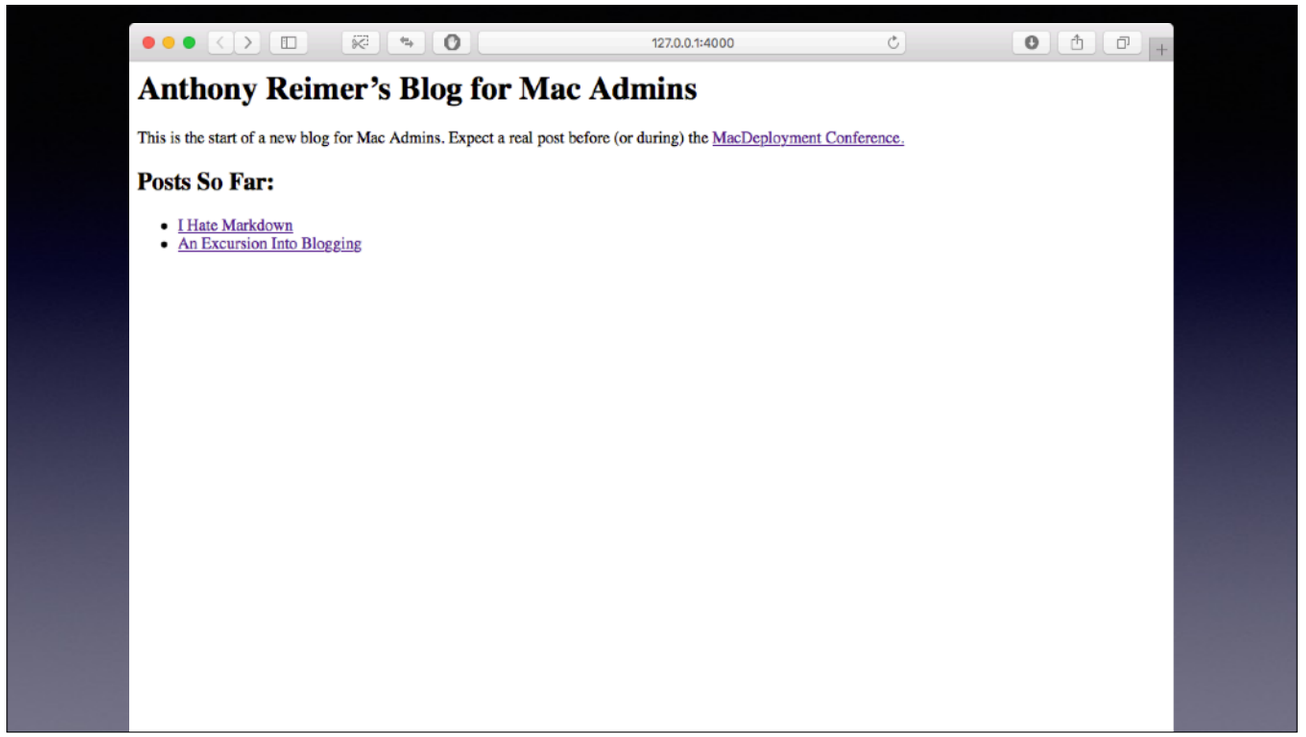


So the Jekyll site has a nice instruction page describing these technical details, the beginnings of describing YAML Front Matter, and then >

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

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

this nice snippet of markup that you can insert in your index (or home) file. (Reference to Liquid)

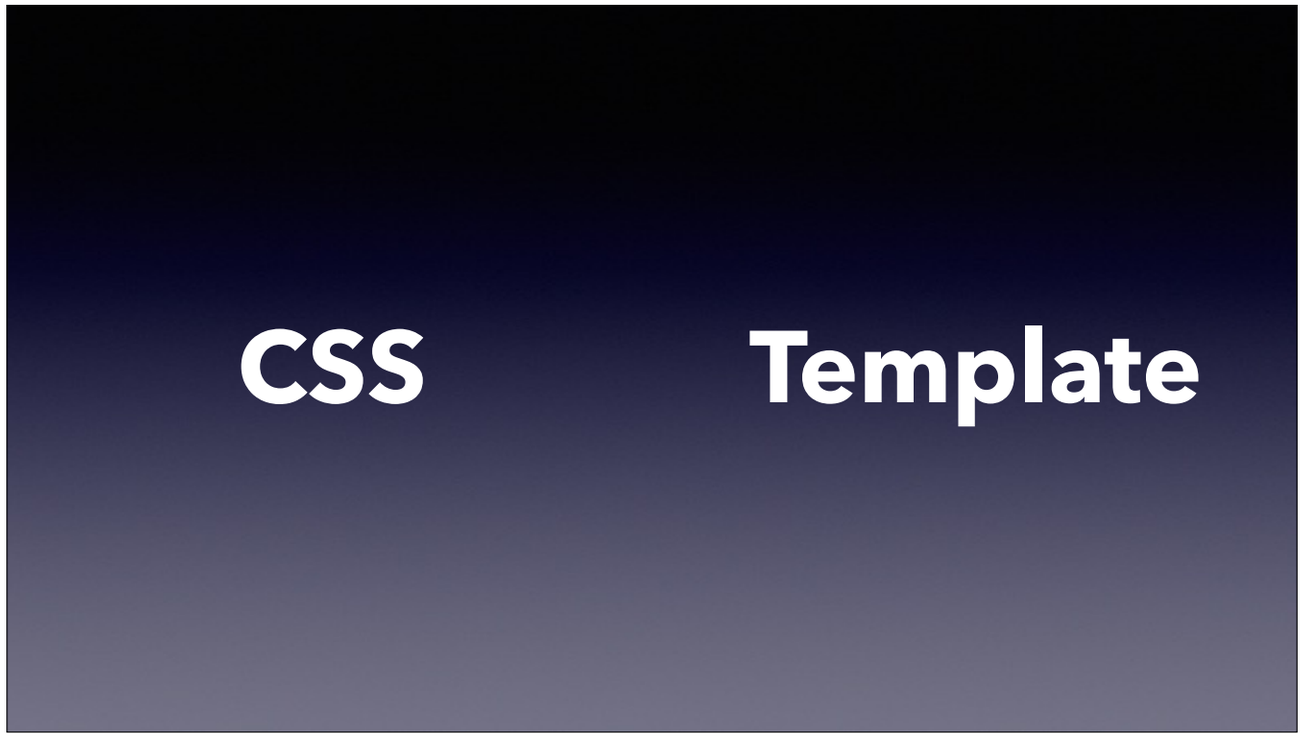


Hey, that's an improvement! The blog posts don't have formatting either, but the content works.

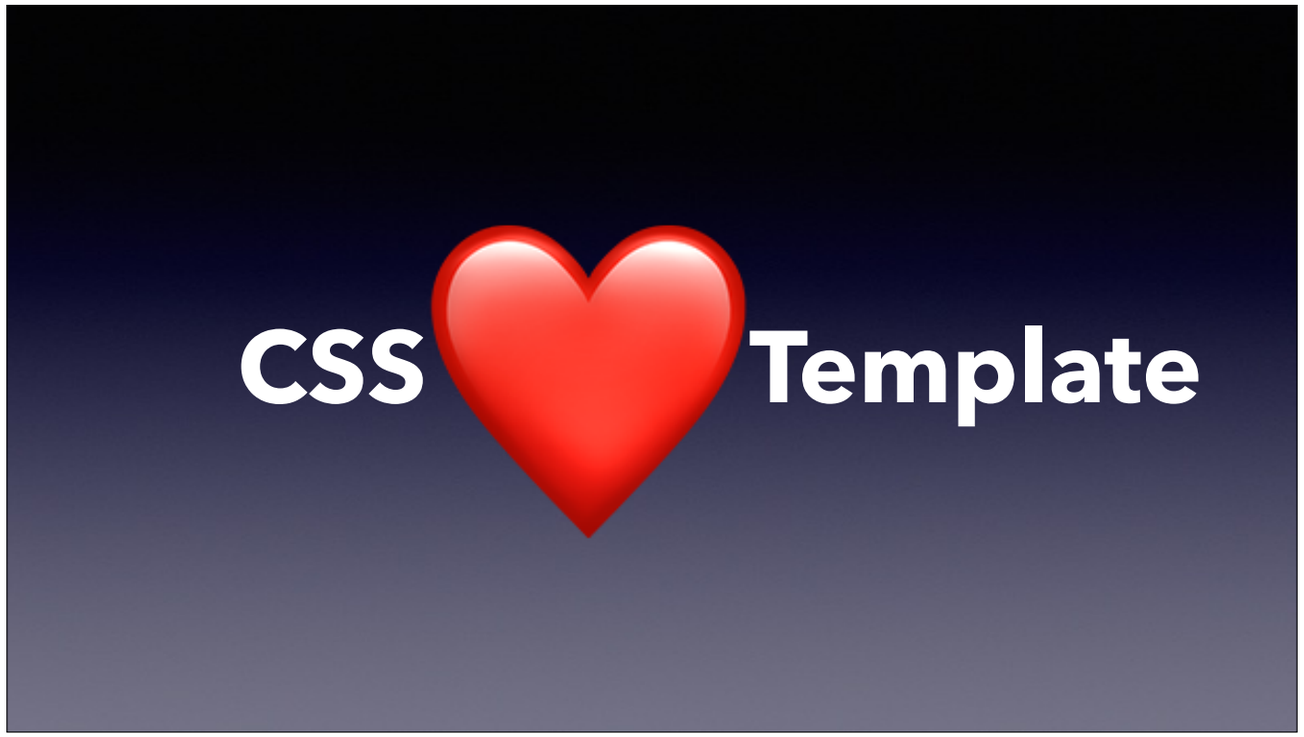
Step 4

Presume How Jekyll Themes Work

Since web sites are formatted using Cascading Style Sheets, I assumed that a theme was just a friendly way of saying CSS. • But it also does some of its magic by including markup like site headers and footers, much like a mail merge template does. • So a Jekyll Theme is actually both CSS and a template. But I didn’t know this yet.

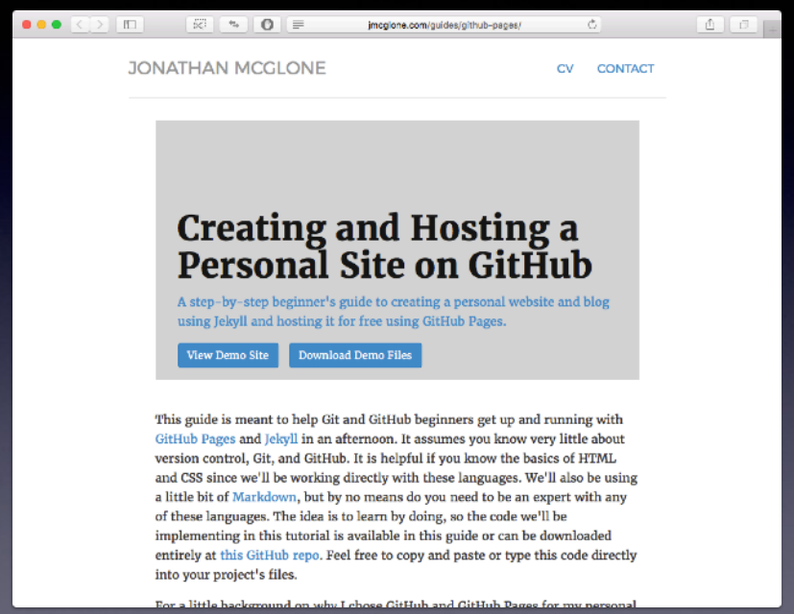


Since web sites are formatted using Cascading Style Sheets, I assumed that a theme was just a friendly way of saying CSS. • But it also does some of its magic by including markup like site headers and footers, much like a mail merge template does. • So a Jekyll Theme is actually both CSS and a template. But I didn’t know this yet.



Since web sites are formatted using Cascading Style Sheets, I assumed that a theme was just a friendly way of saying CSS. • But it also does some of its magic by including markup like site headers and footers, much like a mail merge template does. • So a Jekyll Theme is actually both CSS and a template. But I didn’t know this yet.

http://jmcglone.com/guides/github-pages/



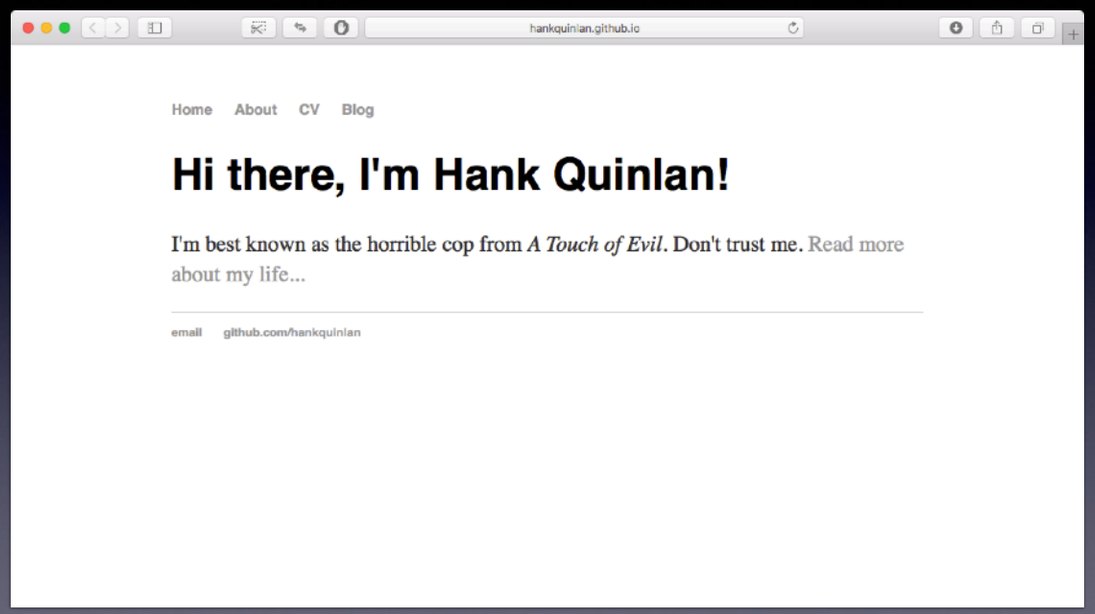
So I found this guide by Jonathan McGlone, which is good for those who know a little HTML and CSS. It used Jekyll mostly for the Liquid-based variables and the menu structure, not for the CSS. So if you come from a traditional web publishing background and can write your own CSS, you could actually do it this way. That was my first success, but it led me astray on how Jekyll Themes worked.

http://hankquinlan.github.io/

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hank Quinlan, Horrible Cop</title>
    <link rel="stylesheet" type="text/css" href="/css/main.css">
  </head>
  <body>
    <nav>
        <ul>
            <li><a href="/">Home</a></li>
            <li><a href="/about">About</a></li>
            <li><a href="/cv">CV</a></li>
            <li><a href="/blog">Blog</a></li>
        </ul>
    </nav>
    <div class="container">
        <div class="blurb">
            <h1>Hi there, I'm Hank Quinlan!</h1>
        <p>I'm best known as the horrible cop from <em>A Touch of Evil</em>
           Don't trust me. <a href="/about">Read more about my life...</a></p>
        </div><!-- /.blurb -->
    </div><!-- /.container -->
```

Here is the first 2/3 of the page he used for his home page. What does this look like? (A: Fully functional HTML page.) Add the CSS he provided and we have a working, functional web page >

http://hankquinlan.github.io/



So if you didn’t want to use Jekyll themes, you could stop here. But we want to Blog, so he then goes into that. In retrospect, if I had not been in a hurry, I might have been confused by the “why” of what he was doing, but it should have worked.

Static Web Site generator

Themes (via include)

“Blog aware” (Liquid)

Add to any web site (Ruby)

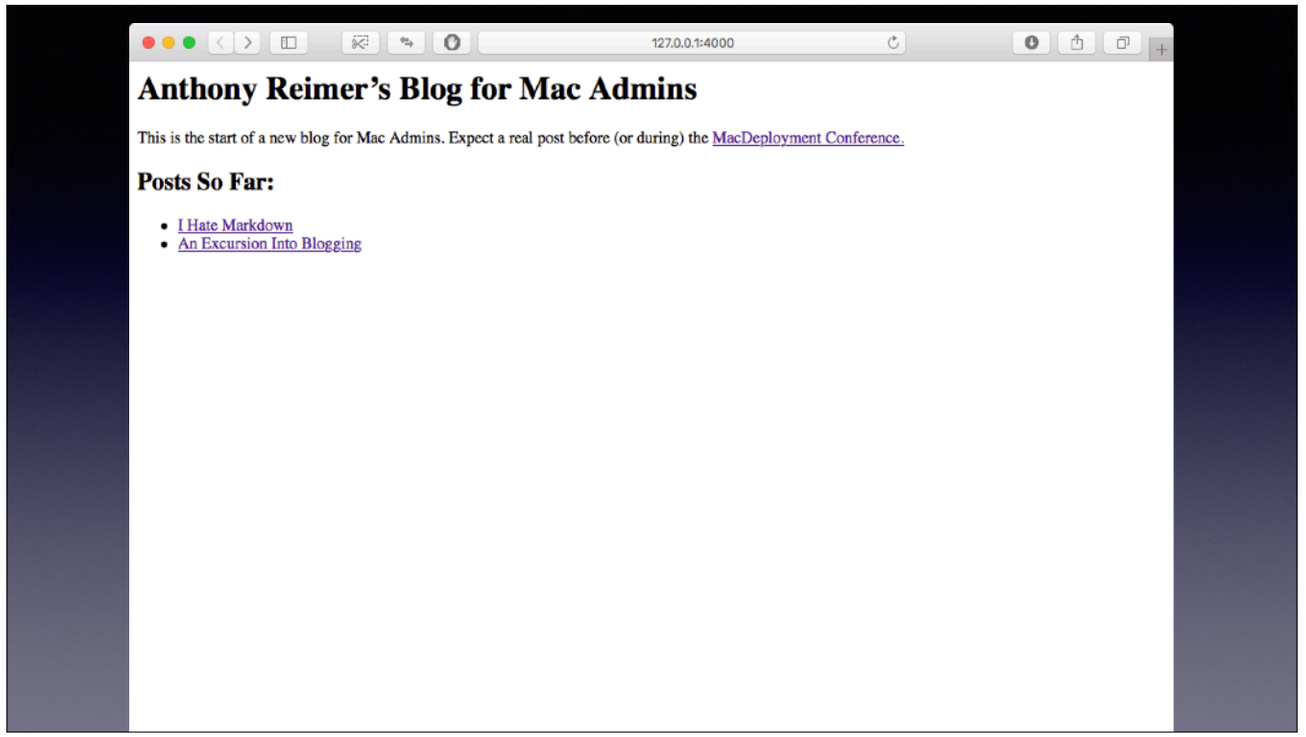
So let’s go back to what I said earlier about Jekyll. We need to break this down further now to understand what’s going on.

From what I did in Step 3, I found out that I could use variables and conditionals of a sort using Liquid in my markup that would work without any Jekyll theme. As it turns out, you can also use a Jekyll theme without using any Liquid.

https://jazzace.github.io/

```
---
layout: post
title: The MacLabs Blog
---
```

So then I learn I can just add lines like this to the top of any page and it should activate the theme. This is the YAML Front Matter that was mentioned on the Jekyll site. Great. Perhaps that is the glue between GitHub Pages and Jekyll. >



Still not working.

Step 5

Skip Reading Critical Instructions Because Reasons

I've been jumping amongst Jonathan McGlone's tutorial, the Jekyll documentation, and the GitHub Pages documentation to try to find the missing link. I later determine the information is there but not all in one place. Let me summarize for you what I learned.

```
<!DOCTYPE html>
<html>
<head>
    <title>The Hello World Blog</title>
</head>
<body>
    <h1>Hello World</h1>
    <p>I have something to say.</p>
    <!-- Footer -->
    <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.

■ Layout

```
<!DOCTYPE html>
<html>
<head>
    <title>The Hello World Blog</title>
</head>
<body>
    <h1>Hello World</h1>
    <p>I have something to say.</p>
    <!-- Footer -->
    <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. Layouts have names. You can have multiple layouts in a theme.

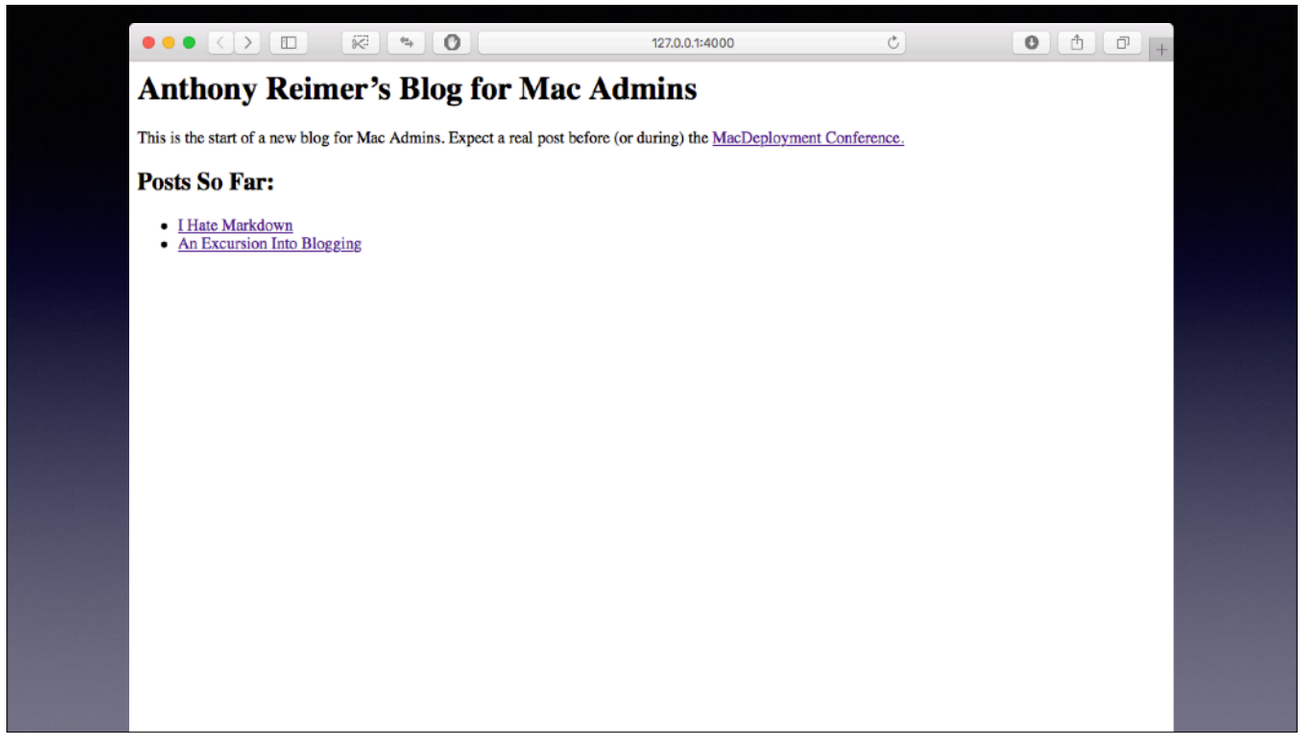
_layouts/
default.html

```
<!DOCTYPE html>
<html>
<head>
    <title>{{ site.title }}</title>
</head>
<body>
    {{ content }}
    <!-- Footer -->
    <p>&copy; 2017 Anthony Reimer</p>
</body>
</html>
```

hello.html

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

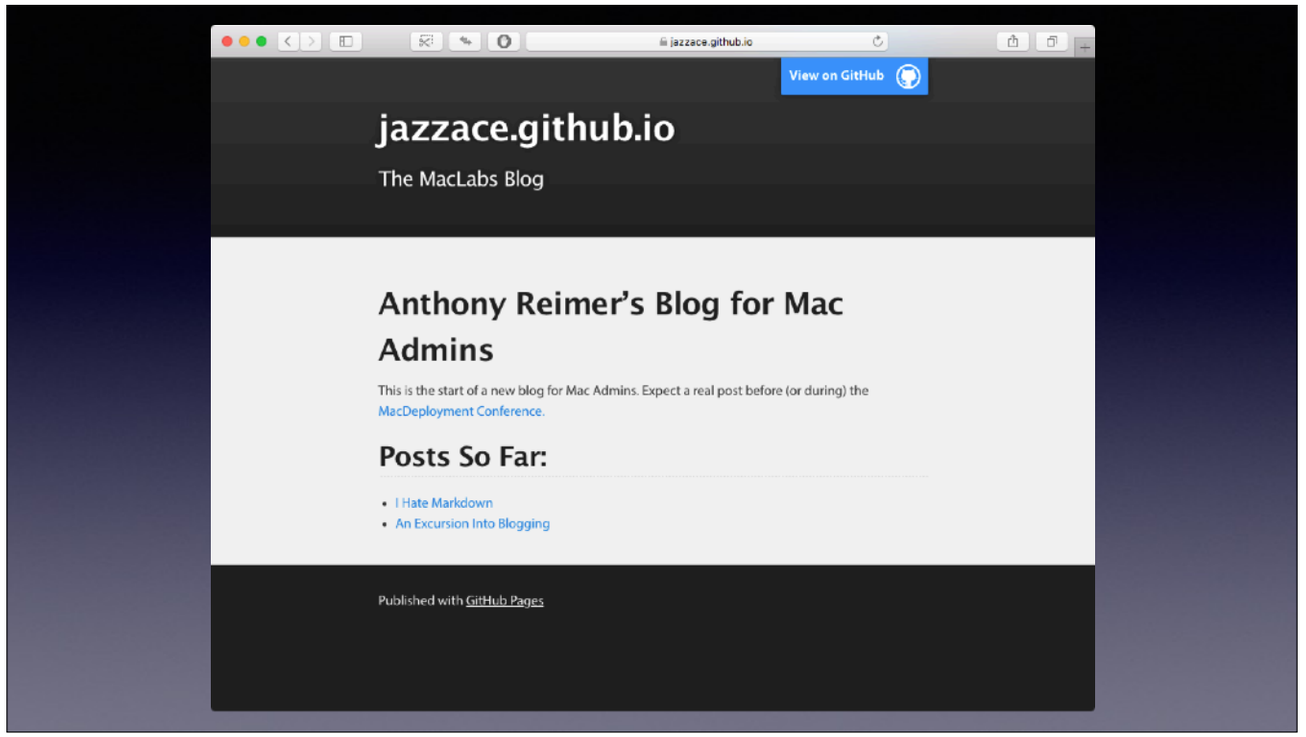
So I can split this into a layout and a page. If you now open hello.html in a Jekyll-enabled site, it should render exactly like the previous single HTML file. But now I can re-use this layout throughout my site, just writing the body content and the special YAML Front Matter to tell Jekyll which layout to use, the title of the page, and other variables that you can specify using Liquid. SO I finally understand this, apply it to my site using the theme I had chosen >



And it still doesn't work. In the end, it was a stupid mistake.

```
---
layout: post
title: The MacLabs Blog
---
<h1>Hello World</h1>
```

In some of the documented examples, they showed themes with more than one layout, usually default and post. I, of course, could not pick the default theme. I picked one with only one layout: default. Yet I kept typing “post” in as a layout. Once I changed it to default >



I got this. Same when I went to the blog posts. Alternately, I could have created a layout named post, which I did in the end to list the past blog posts in the footer rather than where they show up on the home page. I now know a lot more about how this works, so if you take my adventure as a cautionary tale and start with the understanding that I now have, you should be able to follow the documentation and get blogging using GitHub Pages and Jekyll. But perhaps you want a more conventional solution. Vaughn will now talk about blogging with WordPress.