**Video Title: "Advanced Markdown: Introduction to Pandoc Conversion"**
https://youtu.be/IOxGQErK1Zw

**Transcript:**

This video provides an introduction to Pandoc conversion. As we learned in the previous video, titled "Your First Markdown Webtext," one of the advantages of markdown is that it can be converted into other things. In that video we used StackEdit to convert markdown to HTML, and then we used Dropbox to host that HTML as a single page webtext. In this video we will be learning about Pandoc. Pandoc is a conversion tool that converts from one document type to another. And of course, the primary document type is markdown. As the diagram shows on the homepage, there are many document types that it converts to and from—which is its primary use. John McFarlane, who is discussed quite a bit on the "History" page of the site, is a history professor from the University of California Berkley. And he's the creator and maintainer of Pandoc, and Pandoc is open source and free to use by all. Of course, this video cannot provide a comprehensive tutorial for Pandoc. There are many videos and tutorials on the Internet for all the different sorts of things that Pandoc can be used for. However, here in the resources page I have given a couple of examples. One of them is from Pandoc itself on its website, and it shows how to create an ebook with Pandoc. Using multiple markdown files, with each file being a chapter, it is brought together with Pandoc using CSS to create the styling for the ebook. Another example of what can be done with Pandoc is using Pandoc to create full websites. This website first gave me the idea of building my own person website with markdown and Pandoc. And then finally, the last example for Pandoc that I provide on the resources page is Pandoc for academic webtexts. When I first started to learn markdown and started to learn Pandoc. I came across this great tutorial on the *Programming Historian* site. The *Programming Historian* is peer reviewed publication for tutorials and lessons in the humanities. This site is great for introducing markdown and Pandoc for creating what is often referred to as "reproducible" academic scholarship.

With the remaining part of this video I am going to provide a demonstration of some of the basic conversion functions that make Pandoc so useful for those interested in markdown. First of all, Pandoc is a command line program. Meaning, you run it from the command line, within a terminal. Now, my operating system is Linux, but Mac or OSX users also have this same option to use a terminal. Now, Window users have a very limited terminal space, and often it is recommended to install some Bash terminal or command line for Windows users. I am not going to get into that here. I am assuming that if users will go this far that they will take the time to learn that part. What I am going to provide is just a demonstration. For those that are interested in learning Pandoc or interested in an advanced understanding of markdown, using Pandoc, and what is possible were you to do so.

So, the first thing we want to do is replicate the conversion from the previous video, where we used StackEdit to turn history.md into a styled HTML page. Of course, in this case, now we are going to use Pandoc to do that. So, the first thing we need to do is tell the terminal to move into the Desktop where the files are located. To do that, we type "cd Desktop." Now we're in the Desktop directory, and if we type "ls" we can see its contents. And we can see we have the terminal pointed at the Desktop where our files are. Now, to call Pandoc, since it's a command line program, we just write Pandoc at the command line. And then, "space" we give it the input, which is history.md. After that we designate an output: history.html. Now, by giving it the extension of HTML Pandoc knows to convert it to the HTML format. Next, we want to give it our CSS file, which is "screen.css" as we see on the Desktop. And the

last thing is a command for Pandoc called “standalone” which is designated with “-s.” Now we can open this and see how it looks. And, success! We have replicated the same page output that we had from our StackEdit conversion.

So, the next thing I want to display for you using Pandoc is how to use bash scripting to do more complicated things with Pandoc. For this example I am going to do a very simple bash script, and I am going to give a short description of what that is. All that we plan to do with this bash script is take history.md, and go from one input—as we already had here previously—and have 3 outputs. We will have the HTML file still, and then 2 additional ones: a Word Doc file and a PDF file. And so, we’ll use Pandoc to go from 1 input to 3 outputs. So we’ll do this with the basic text editor in the terminal. You could use any text editor, such as the basic text editor that comes with OSX. I often use the basic text editor that comes with Linux called Gedit. I also really like GitHub’s new text editor called Atom. But for this video I’m just going to use Nano, which is a basic text editor included with the bash terminal on Linux. So, if I do “Nano” with a space after it, whatever word I type next will be the file name. So we’ll use the term for this bash script called “multiout.sh.” And there you have it. A very basic text editor. It’s an empty file. We’re going to write it now. So, the first thing we want to do is give it a basic command at the top that tells the system that this is a bash script. The next thing we want to write is that command that we just wrote previously. Where we use Pandoc to take the markdown and create a webpage. And that’s the same command we wrote before. Let me check it here. Yes, that looks correct. Now, right after this, we’re going to write two more commands. We’re going to give it the input again, and this time we’re going to tell it “output=history.docx.” And again, since I gave it the “.docx” extension, Pandoc knows to turn it into a DOCX file. And now, I see I forgot to add output up here, so I’m going to add that in. And then I’m going to create one more. So there you have it. We have 1 input and 3 outputs. Now, there are ways to write a single command that does this all one line. I’m using this—replicating three separate commands to show you have to use bash script to create multiple commands. Now I used this same logic to actually create my whole website as I showed earlier in this video. I don’t use this exact logic that you see on the screen here, but the basic logic of using a bash script to do multiple Pandoc commands to create a complex set of documents or pages. Now in this case we are taking 1 input and creating 3 outputs. In the case of my website I take multiple markdown files, convert them to multiple HTML pages, and add navigation to all of them using Pandoc.

So, in this example, we’re going to go ahead and save it. And we’ll call it “multiout.sh.” So, there’s one more thing we need to do. Now, you see it appear on the Desktop which is good. So the next thing we need to do is just make this an executable file. We do that with the “chmod +x” command, and then we type the name of the file. And now we can call the file. And all this is going to do when I hit “Enter” here—it’s going to take the bash script, and it’s going to run those 3 commands, and from 1 input create 3 outputs. Oh, I’m sorry, I forgot to type the “multiout.sh.” And there we go. So, now, even though you didn’t see it. It actually rewrote the HTML, and now we have 3 outputs from 1 input. All using markdown. So, let’s open the PDF and look at it. And you can see it is nicely styled. Now in this case I didn’t use CSS. That’s for a specific reason. Pandoc uses a markup language called Latex for doing its PDFs. Now, you can create templates in Latex for Pandoc as well. Now I’m not going to show that in this video—it’s another complex step that’s worth learning if you’re going to do a lot with Pandoc and PDFs. But for this I just wanted to show you how nice the PDFs look without any additional styling. And again, we have the Word file. Also generated by our markdown. Now this is in Libre Office, so Libre Office doesn’t look as good with Word files, as if I opened it in Microsoft Word. I don’t use Word because I

prefer open source software. But I just wanted to show you that it effectively takes markdown and turns it into a Word file. So, if you're someone like me who is interested in moving to markdown for all of your writing and drafting needs, and for your initial writing and freewrites, it's useful to turn it into a Word file for when you're ready to turn it into a finished document for a full first draft.

So, the final thing I want to show you is the logic behind my own personal site using Pandoc. As I showed earlier in the video, my own personal site is made totally from markdown and then converted with Pandoc and an HTML template. Here are the files on my GitHub page, which is totally open to the public. You can come in and look at these and see how I construct my site using Pandoc. And I have a file here called "compile"—this is a bash script. It's the same as the one we made before, but it is a little more complicated. You can see that I have put my Pandoc commands in block style. So rather than having them in a single line, I have these little indicators that say that the line continues. This allows me to organize my code in a nice visual style. So you can see with my homepage. I create this by calling my markdown from my markdown folder that holds my markdown files. So I call "home.md," I style it with my CSS, I add the template for the body of the HTML, I designate the output, I tell it "smart"—"smart" specifically adds a few in-text styling that makes text more readable. And then I have my header and footer templates that I do with the "include-before-body" and the "include-after-body" Pandoc commands. And then I also add a title prefix to the page. And I do this for all the pages of my site. I have 3 pages of my site and I replicate the same thing 3 times to create all the pages of my site. So each one created with the Pandoc commands shown on this page. And if we come back here you can see. All of the markdown files are very simple. So whenever I want to edit something on my page I just come in and edit the markdown, and then I run my bash script—just like we saw in the previous video. And it creates the site. Now, it took me a while to figure that out and learn Pandoc, and it's certainly an investment. But I think an investment that's worth it. Having started with something approachable like markdown, I was able to slowly figure out more and more about CSS and HTML. Slowly become more comfortable with the command line and with scripting. And eventually I have moved into learning HTML and CSS directly now. However, I still prefer markdown, and think that it is a very writer friendly way to write for the Internet and to write webtexts.