ENUG 2021 Day 3: TIL: Creating Open Documentation Through Public Note-taking and Literate Programming (Jay L. Colbert)

Thursday, 10/28 1:00 PM

Transcript generated by Zoom Live Transcription

00:00:00.000 --> 00:00:00.000

So yes hello Thank you. My name is jL Colbert you pronounced it right Yay, you know, sorry if I disappoint anyone for my last name not being cold bear.

00:00:00.000 --> 00:00:00.000

I am the metadata and discovery strategy librarian at the University of New Hampshire, and today I will be presenting TL or today I learned, creating open documentation through public note taking and literally programming.

00:00:00.000 --> 00:00:02.000

If you want to follow along.

00:00:02.000 --> 00:00:03.000

Hello.

00:00:03.000 --> 00:00:10.000

You can go to this link here, these slides are just plain text there or dmo documents.

00:00:10.000 --> 00:00:18.000

And you can follow along the slides here on an HTML file, or you can even look at the plain text if you want.

00:00:18.000 --> 00:00:31.000

So let's get started. Briefly the presentation I'm going to briefly talk about who I am, since this is sort of begins as a little bit of an auto ethnography, and why that's relevant to this topic.

00:00:31.000 --> 00:00:43.000

And then I will actually go into the sort of problems I had while learning premium ve and the solutions and that's what inspired this talk today.

00:00:43.000 --> 00:00:58.000

Then I will go into the importance of documentation and your Why is it so important that we do this, and then I will sort of go through this framework for making documentation and that is learning in public.

00:00:58.000 --> 00:01:00.000

Then I'll go through some methods of doing that.

00:01:00.000 --> 00:01:04.000

And that will include a small demonstration.

00:01:04.000 --> 00:01:09.000

And then I will go. I'll leave time for some questions at the end.

00:01:09.000 --> 00:01:16.000

Before I get started, is there anything anyone needs to ask or anything.

00:01:16.000 --> 00:01:20.000

No, I'm just looking at the chat. Okay.

00:01:20.000 --> 00:01:36.000

So who am I, why, why is this relevant to what I'm talking about today. So my name is Trey on Colbert as I said, I use him pronouns, and I am the metadata and discovery synergy library and at the University of New Hampshire, and most importantly for this

00:01:36.000 --> 00:01:49.000

talk, I want to point out that my bachelor's degree is in literature. I do not have any sort of formal training in computer science or programming.

00:01:49.000 --> 00:02:00.000

The only sort of tech stuff I know has been self taught, or when I was focusing on metadata in my master's degree, but I have an English degree.

00:02:00.000 --> 00:02:05.000

I, I am not a computer science person at all.

00:02:05.000 --> 00:02:09.000

Yet I am in a technical job.

00:02:09.000 --> 00:02:19.000

So how does this play into learning free movie and why was this so difficult for me, so apologies if anyone here is actually from actually bris.

00:02:19.000 --> 00:02:24.000

I'm about to insult your documentation a little bit so I apologize.

00:02:24.000 --> 00:02:42.000

So in September of 2019 I was hired as the University of New Hampshire's metadata and discovery strategy librarian, technically of September 30 so really it was October, and my position responsibilities as faculty tend to be less hands on.

00:02:42.000 --> 00:03:05.000

They're more about vision and strategy and coming up with, you know, standards and projects. But one of the sort of main hands on day to day things of my job is that I maintain our pre movie discovery layer, which Umh had migrated to from millennium.

00:03:05.000 --> 00:03:23.000

In July of 2019. I want to make sure you look at those dates there. So they implemented me like they went live with Primo carries in the, in the chat that's, that's good to know that I'm not the only one that finds the documentation inadequate Okay good.

00:03:23.000 --> 00:03:44.000

I'm among friends but I'm so una had migrated and went live with Primo ve and Alma Angela terminate teen, which means they went through the entire implementation and set up the migration process before I was even hired before I even started.

00:03:44.000 --> 00:03:52.000

So no one who would actually be working on pre movie was part of setting it up.

00:03:52.000 --> 00:03:57.000

I did not get to be a part of setting up the thing that is my job to to maintain.

00:03:57.000 --> 00:04:04.000

So that already sets us out with a little bit of a struggle here.

00:04:04.000 --> 00:04:21.000

So my challenges. I was new to this software, I this is only my second professional position, I'd only ever used Primo as a user, before, but I had never had to manage the discovery like back end, aspect of it.

00:04:21.000 --> 00:04:33.000

But not only was I knew this awkward. Everyone at my institution was they had had Millennium for over a decade. I think I've never even used Millennium Millennium I have no idea what that looks like.

00:04:33.000 --> 00:04:49.000

And so I ended of the software, as well as everyone else at my institution. Like I said, I have no formal tech background beyond metadata and then, you know, I can tinker a bit but you know I've never been formally taught.

00:04:49.000 --> 00:05:00.000

And then we get into some problems with the documentation. So as I was going through some of the, the trainings on the actually press website and go through the documentation.

00:05:00.000 --> 00:05:16.000

I was getting very confused. Okay, what's the difference between Primo and Primo ve, what is the Primo in UI. What's the Primo back office does ve have a back office, just be have the new user interface.

00:05:16.000 --> 00:05:28.000

Can I look at the same documentation for either of these what's what's going on what acronyms that librarians love like Why is no one explicitly even mean right.

00:05:28.000 --> 00:05:32.000

The documentation is always clear about what's what.

00:05:32.000 --> 00:05:45.000

The most helpful thing is, is when you're in all my Primo, you can sort of go up into the house and be like, I want to help for this specific page, and it will take you to the right documentation page but if you're trying to find it on your own.

00:05:45.000 --> 00:05:49.000

I've found it to be very difficult to navigate.

00:05:49.000 --> 00:05:56.000

And the documentation isn't always clear about what you should be doing or what you're looking at, either.

00:05:56.000 --> 00:06:12.000

It also can be years out of date, including the official documentation, but also a lot of the sort of third party information like stuff you might find on GitHub, and especially that's what I'll be talking about today regarding the development environment

00:06:12.000 --> 00:06:18.000

as a lot of that sort of third party information is also years out of date.

00:06:18.000 --> 00:06:32.000

So luckily the instructions for the perfect Primo development environment on the official reverse GitHub repository do contain some information for for Primo TV, you can get it up and going.

00:06:32.000 --> 00:06:49.000

But most of the third party packages. Don't. And that's largely because the, the libraries, developing those packages like the Boston University Libraries with like they have their own paywall, and the Help menu, shout out if you're one of the people

00:06:49.000 --> 00:07:03.000

from Boston University Libraries who makes those that maintains them. They're great, but because they don't have access to free movie they can't test them on ve and so if there's any differences into how they should be set up, or not, that information

00:07:03.000 --> 00:07:12.000

isn't available, they're always like, well, we assume this will work but we haven't been able to test it on it.

00:07:12.000 --> 00:07:24.000

And then also many of the third party packages, use this browser by command line flag to compile the environment.

00:07:24.000 --> 00:07:40.000

And actually does not provide any official documentation for using that when I first saw that and I believe it was in the Boston University packages I was like what is this talking about how do I find out how to use this Why is this not working.

00:07:40.000 --> 00:07:49.000

And some of the packages won't work unless you use that so I just switched over to using that entirely.

00:07:49.000 --> 00:08:01.000

So how did I even, you know, manage to learn how to do all this stuff, considering the documentation is kind of lacking, including being out of date or even incorrect.

00:08:01.000 --> 00:08:19.000

The first was that I explored a lot of GitHub repositories, including the official actually rest development environment one, but also of people who had their institutions development environments on GitHub, or people who develop those sort of third party,

00:08:19.000 --> 00:08:31.000

and Primo explorer packages that you can install. So explore those repositories and if you are doing this yourself some of the places that you might look our commits.

00:08:31.000 --> 00:08:35.000

That might not be the most obvious place you might look.

00:08:35.000 --> 00:08:52.000

But a lot of times people are doing version control and get hub. Well, they will normally, type in the like the message when they do a commit, of what did they change and why, or it will show the difference between what they changed and why.

00:08:52.000 --> 00:08:59.000

So that can always be helpful. I'm looking at the issues both open and closed in the repositories.

00:08:59.000 --> 00:09:14.000

I, you know, one thing my, my father who worked at an internet company was that when I was a kid, taught me when he was teaching me how to use a computer by myself is that I am never the first person to have the problem that I'm having, right, I am not

00:09:14.000 --> 00:09:27.000

the first person in the world to ask this question, or to have this error code or to have this thing break on me. And so going into the issues and seeing if anyone else has raised this issue before or if there's any other useful information I might be

00:09:27.000 --> 00:09:50.000

able to find that was another helpful thing. And the same thing with pull requests. One of the best things about these sort of open, open source code repositories is that people can contribute to them, and that I can include submitting their own

00:09:50.000 --> 00:09:56.000

suggestions or changes to the, to the code.

00:09:56.000 --> 00:10:00.000

And I think what I did was that I watched conference presentations like this.

00:10:00.000 --> 00:10:06.000

However, a lot of conference presentations I ran the same problems that I did with the documentation.

00:10:06.000 --> 00:10:25.000

And that sometimes they were years out of date, or that just that they were older, or they were sort of hyper specific to the institution prison. Dang, or it was still only relevant to Primo and not ve in that maybe the he hadn't even existed yet, or

00:10:25.000 --> 00:10:32.000

the methods that they were using like relying heavily on back office or not applicable to be at all.

00:10:32.000 --> 00:10:47.000

So a lot of messing around and finding out both and just tinkering and saying what I could find out but also like breaking things like I messed around I found out what happens when you mess around and break things and you have a lot of sleepless nights

00:10:47.000 --> 00:10:59.000

until you try to figure things out where you think you have it, and you tell your colleagues, oh yeah, you know, this should take me about five minutes to install this and put it into the package and then two weeks later when you're still trying to figure

00:10:59.000 --> 00:11:01.000

out why it's not working.

00:11:01.000 --> 00:11:06.000

That kind of messing around and finding out happens a lot as well.

00:11:06.000 --> 00:11:23.000

And I eventually that's kind of what I spent the entire first year of my job doing, which happened in quarantine very fun time to start a new job was just trying to figure out the Primo development environment.

00:11:23.000 --> 00:11:34.000

So this leads me to the importance of documentation, and I want to put a qualifier on that and the importance of good and open documentation.

00:11:34.000 --> 00:11:53.000

So the first is that a single source of truth saves time and energy, there's only one place that people have to look for something that saves them time trying to hunt around to see if it might be part of me to see if it might be somewhere else.

00:11:53.000 --> 00:12:12.000

documentation is essential to quality control and process control. So if you want everyone to be doing the exact same thing and the exact same way and if you want to be able to reproduce that time and time again, and not sort of introduce any, any variation

00:12:12.000 --> 00:12:21.000

and how people do something, you need to document that workflow, and how you might check to make sure it is following that workflow.

00:12:21.000 --> 00:12:25.000

Documentation also cuts down on duplicative work.

00:12:25.000 --> 00:12:37.000

If someone else has already done this before. There's no use and you trying to reinvent the wheel you should just be able to follow the instructions that someone else has already written or say someone else already has a code snippet, why should you have

00:12:37.000 --> 00:12:55.000

to rewrite that code snippet, if someone else already has a great documentation makes hiring and onboarding easier. So say you know it's not unheard of, say something were to happen to me, and someone else, suddenly had to manage the University of New

00:12:55.000 --> 00:13:16.000

Hampshire's Primo the discovery layer. If I have done my job right and documented, what I do and how I do it and why. That makes hiring someone else or onboarding someone to do that job way easier than them having to start from scratch again.

00:13:16.000 --> 00:13:34.000

Documentation also increases collective knowledge. so if we make our documentation, open, and even collaborative that everyone who works with a movie or really any library software or anything at all, really, we can sort of increase the amount of knowledge

00:13:34.000 --> 00:13:49.000

we all have about it because the way I go about something might be different than the way someone else might go about something and we might discover that our approaches are complimentary or that their approach is better or that mine is better or oh I've

00:13:49.000 --> 00:13:56.000

been looking for a way to do that I'm so happy someone else's figured that out

00:13:56.000 --> 00:14:06.000

with documentation we are able to share our knowledge with others both internally to our own organization and externally within a wider community of practitioners.

00:14:06.000 --> 00:14:11.000

I know the Digital Library Federation's metadata assessment.

00:14:11.000 --> 00:14:27.000

Working Group, I believe it is there this amazing website that's the thing it's like the library workflow clearing house or something like that. And there's so much useful information on there are people who have just shared their documentation and the

00:14:27.000 --> 00:14:30.000

workflow or their projects.

00:14:30.000 --> 00:14:36.000

And so that others can see how this specific institution to this specific thing.

00:14:36.000 --> 00:14:43.000

But without documentation, knowledge is lost over time consistency is lost and quality control as lost.

00:14:43.000 --> 00:15:08.000

So we recently ran into this at the University of University of New Hampshire. when, as part of both a push for improving, you know, reducing our budget, as well as just to help ease the burden of coping financially, their university offered these retirement

00:15:08.000 --> 00:15:20.000

packages to people if they were maybe nearing retirement age or something like that but if they retired when they weren't planning on it. They got all of these sweet perks that they wouldn't have otherwise.

00:15:20.000 --> 00:15:23.000

And that meant that we lost.

00:15:23.000 --> 00:15:29.000

About half, maybe of our tech tech services department.

00:15:29.000 --> 00:15:45.000

And so there was this huge scramble to make sure we did not lose that institutional knowledge that you know this person who had been doing this job for this many years to make sure that all of the knowledge that they had about doing that job all of their

00:15:45.000 --> 00:15:48.000

expertise about doing that job.

00:15:48.000 --> 00:16:03.000

We had to make sure that was all written down somewhere. We're also moving from box to an entirely Microsoft ecosystem. And so we had to go, Oh, where is this documented and box and how and we have to transfer that over anyway.

00:16:03.000 --> 00:16:13.000

So save yourself some, some grief and having to do that all the last minute and make sure you're doing that right as you're learning something.

00:16:13.000 --> 00:16:29.000

And so when I was doing research for this for this talk a few months ago, especially in the realm of technical communication, there seems to be two different kinds of documentation, at least for more technical documentation.

00:16:29.000 --> 00:16:34.000

There's system based, which is basically here's what the system does.

00:16:34.000 --> 00:16:45.000

This is very useful for people who might be creating or maintaining the tool or system itself, you know, this is how it does this thing if it breaks and you need to fix it.

00:16:45.000 --> 00:17:03.000

You know, whatever. But then there's also task based. And this is mainly for people who might be using that system and that's not necessarily like library users and the history movie, this is for the people who will be implementing Primo at their universities

00:17:03.000 --> 00:17:22.000

or institutions here's how you do your job, using the system. So for the people at actually birth who create and maintain Primo those more system based documentation is probably more useful and applicable, but for all of the rest of us, and not that.

00:17:22.000 --> 00:17:41.000

Not that knowing how it works, isn't important. But we are much more concerned with task ways okay how do I, I don't know how do I make sure that students who are off campus during the pandemic.

00:17:41.000 --> 00:17:55.000

How can I make sure that when they're using our discovery layer that they can get what they need with as few clicks, as possible. And a lot of documentation sort of forgets this aspect of it, a lot of documentation is more focused on this.

00:17:55.000 --> 00:18:03.000

This system based of how this is how this thing works and here's what it is, and not necessarily on and here's then how you do your job. Using this thing.

00:18:03.000 --> 00:18:12.000

So we need a mix of both but I would say that we probably need to move more towards doing task based documentation.

00:18:12.000 --> 00:18:31.000

So sort of the framework or paradigm, I want to suggest for bettering our documentation both an official levels of, you know, again sorry there's anyone from actually breasts at this talk, but also for those of us who might be having to write our own

00:18:31.000 --> 00:18:42.000

documentation and workflows for the specific things that we do at our institutions. And that is a framework that I came upon recently called learning in public.

00:18:42.000 --> 00:18:57.000

And I like this, this quote about it. So learning the public is scary for many reasons people can find and cling to outdated information and users are exposing their knowledge during a vulnerable time in the project, when they don't yet have all the answers.

00:18:57.000 --> 00:19:02.000

However, during this part of the process is when learning can actually be most valuable.

00:19:02.000 --> 00:19:14.000

This is very scary, but I promise, it has a lot of benefit to it. So what can you do with learning in public, what is it for, what are some of the perks of it.

00:19:14.000 --> 00:19:24.000

Well, the first is that it's a really good way to track your own learning process but also other people can track it or you can track the learning process of others.

00:19:24.000 --> 00:19:28.000

It acts as a public log of what you're learning.

00:19:28.000 --> 00:19:41.000

And so this is an interesting way to sort of see both with the public blog, and they're tracking the process of like how you have improved over time, or how your knowledge has changed over time.

00:19:41.000 --> 00:19:59.000

It's also a way to give continual feedback so other people are able to see what you're learning, they can give you feedback on it, so a lot of people I see that are into this whole like learning in public or working with the garage door up mindset is

00:19:59.000 --> 00:20:04.000

they might often like have a link to a GitHub repository.

00:20:04.000 --> 00:20:24.000

In whatever website, they're posting their sort of learning and public on so that people can submit pull requests or raise issues, using the language of GitHub, but it's a way for someone to go, oh hey you know I tried this on my own computer.

00:20:24.000 --> 00:20:38.000

And I found that this worked better or you have a typo here that actually changes how this function works, you might want to change it to this or. Oh, I learned this recently and I found this resource to be really helpful.

00:20:38.000 --> 00:20:41.000

So that is another perk of learning in public.

00:20:41.000 --> 00:20:52.000

And that leads into again tapping into this network of community knowledge, which is sort of the same thing as when we have our good documentation rate.

00:20:52.000 --> 00:21:11.000

If I have posted that I figured out how to do this thing with a development environment through hours and hours of tinkering and this is how I did it and why someone else who maybe has to do that at their institution might be able to find my little post

00:21:11.000 --> 00:21:12.000

about it.

00:21:12.000 --> 00:21:24.000

And they can go Oh, I'm so glad I didn't have to spend hours of my life that I could have used like spending time like doing things I really enjoy or, you know, spending time with my friends and family.

00:21:24.000 --> 00:21:32.000

I can spend more time on that and then not having to figure out this thing that someone else has already figured out.

00:21:32.000 --> 00:21:38.000

So what are some methods for learning in public What can it look like.

00:21:38.000 --> 00:21:57.000

So a very very beginner, easy, um, version of this is, you can do it in a get repository. And you could just do this as like a simple markdown or other plain text notes you could upload PDFs and GitHub, really you can upload anything and GitHub.

00:21:57.000 --> 00:22:09.000

A really good example of exactly the kind of thing I'm talking about is this til or today I learned repository that I found and where I got the name of this talk from.

00:22:09.000 --> 00:22:19.000

And so I've been wanting to create one of these for myself but I feel like this person's is a perfect way to just show you exactly what I'm talking about.

00:22:19.000 --> 00:22:28.000

So this person's, a til repository where it's a collection of concise write ups on small things they learned day to day.

00:22:28.000 --> 00:22:40.000

And they have like little categories here so I've been, you know, I've been sick at home this week with a cough. And so I've been trying to like, learn how to do generative art enclosure.

00:22:40.000 --> 00:22:49.000

So I wonder what this person has learned and written about the programming language closure.

00:22:49.000 --> 00:23:00.000

So, we can like hear all the little nerve notes, this person has written so like, let's say I want to load a file into the red ball.

00:23:00.000 --> 00:23:16.000

This person has made this very short little note that says, Okay, here's how you can load a file, and here's how I did it. And look, here's how you can find out more about this, or like this is where I learned it when I've gone through this person's your

00:23:16.000 --> 00:23:19.000

process where in the past they might be like.

00:23:19.000 --> 00:23:34.000

And this is where I learned this information or here's where you can find out more, very simple, very easy this is I think 1234 lines and I'm assuming, two more here to make this a code box.

00:23:34.000 --> 00:23:44.000

But yeah, and then that's all this person wrote about it, and that's already helpful to me because this is something I'm toying around on right now.

00:23:44.000 --> 00:23:56.000

But because this is public, you'll see that other people who aren't this person have created issues, and this repository so like what if you wanted an RSS feed.

00:23:56.000 --> 00:24:12.000

Like, I know this person wants an RSS feed of this repository to see every time this person posts and someone else who's not the owner of the repository went, Oh hey, how about you try, try this and you can subscribe to subscribe to it here, this person

00:24:12.000 --> 00:24:14.000

went great things.

00:24:14.000 --> 00:24:24.000

Other people have also raised pull request. This is where they have edited the code somehow and are asking it to be sort of pulled into the repository.

00:24:24.000 --> 00:24:43.000

This person was just improving a description for something and so you can see the the file that they change. This is literally just them changing the readme of this repository, but already you can start to see how this becomes like a community act of

00:24:43.000 --> 00:24:55.000

people sort of tapping into each other's knowledge and documenting what they know and helping to fill in the gaps of what other people might not know.

00:24:55.000 --> 00:25:10.000

Get repositories are very easy to set up. If you've never done it but if you've never done it, and have questions, please feel free to reach out to me after intermediate and this one you can use, not just for documentation that I thought hey, this would

00:25:10.000 --> 00:25:12.000

be a really cool way to sort of.

00:25:12.000 --> 00:25:30.000

I'm sort of up the ante on this whole learning and public and documenting and public in public note taking, venture. And this is something that's called the digital part and this is also something I learned last year and I've been getting really into

00:25:30.000 --> 00:25:53.000

my personal knowledge management standpoint. I actually finally recently made my own one and put it up. And basically digital gardens are kind of like they're not blogs as this post says, but they're kind of similar to a wiki, or even sometimes the the

00:25:53.000 --> 00:26:10.000

concept of a settled Kasten or slip box where people might write very incomplete unfinished thoughts or ideas and like a note, and it doesn't have to be a fully thought out blog post yet, it can be a seed, like you would play in a garden.

00:26:10.000 --> 00:26:28.000

And then over time you could edit that that post, and you can connect it to other posts and help them to grow and evolve into something bigger. But because it's a garden you also have to be purposeful and cultivate it, and you have to plant seeds and

00:26:28.000 --> 00:26:42.000

you have to pull weeds, you have to actively tend to the knowledge that you're putting out there, and seeing how you might need to change it or feed it or connected or trim it obey.

00:26:42.000 --> 00:26:57.000

So they're constantly shifting and they're sort of a very good encapsulation of the learning and public, and that the information that you put out there might not be the final state of where you are on that information.

00:26:57.000 --> 00:27:02.000

And that's where some of the vulnerability comes on.

00:27:02.000 --> 00:27:17.000

So all of the tools that are mentioned for digital gardening are free and open source. I have purposely omitted tools which are not free, such as problem research, research, and or not open source like obsidian I know I've used up city into and I love

00:27:17.000 --> 00:27:22.000

it but it is not open source and I only want to try to recommend things that are.

00:27:22.000 --> 00:27:29.000

If you want to know more about digital gardening and the tools to do it and even see some examples.

00:27:29.000 --> 00:27:42.000

This is a really good GitHub repository for it, including here's a list of tools here, again, not all of these are free and or open source.

00:27:42.000 --> 00:27:45.000

But then you can see examples.

00:27:45.000 --> 00:27:50.000

So probably the example is this guy Andy.

00:27:50.000 --> 00:28:10.000

And you can see how this person like connects all of their writing together, which I think is just the coolest thing in the entire world that's why I decided to make my own.

00:28:10.000 --> 00:28:14.000

See,

00:28:14.000 --> 00:28:18.000

you're seeing a little slow today, I apologize.

00:28:18.000 --> 00:28:25.000

So, this is a really good resource. If you want to learn more about digital gardening gardening.

00:28:25.000 --> 00:28:39.000

And some of the tools that you might see a lot of these are either work within Visual Studio code or their separate apps, but most rely on some sort of GitHub repository or some place to store markdown files.

00:28:39.000 --> 00:28:47.000

So didn't run is one that I've tried foam is like a free and open source version of Rome.

00:28:47.000 --> 00:29:07.000

You can just do it in and get repository with no fancy tools on top of that, log sack is another one that has just been coming out neuron is another one I use or grown because I'm a dork who uses Emacs, and then tidily wiki is another really cool one

00:29:07.000 --> 00:29:09.000

as well.

00:29:09.000 --> 00:29:24.000

Then then finally to bring everything together and this is where I'll do a small demonstration is the concept of literate programming. When I found out what literate programming was it kind of blew my mind and changed how I thought about writing code

00:29:24.000 --> 00:29:28.000

or documentation or anything really.

00:29:28.000 --> 00:29:37.000

And if you've never heard of it before literate programming is a. I'm going to take a sip of tea here.

00:29:37.000 --> 00:29:41.000

This style and paradigm of programming and documentation.

00:29:41.000 --> 00:29:48.000

And what it does is it emphasizes natural logic or natural language and human logic

00:29:48.000 --> 00:29:54.000

and how it does that, is that you write the documentation.

00:29:54.000 --> 00:30:14.000

And then within that documentation you actually embed your code snippets, instead of writing your code and putting sort of documentation as comments within your code, and through various tools, and it will actually generate the software or compile the

00:30:14.000 --> 00:30:33.000

full code from the documentation itself. So all of the code lives within the documentation, and then through whatever tool you use it will take all of those code chunks out and tangle them into the appropriate documents.

00:30:33.000 --> 00:30:41.000

But if you need to change anything you only need to change it and your main documentation files.

00:30:41.000 --> 00:30:49.000

And what's cool about this is you can actually you don't have to write the code snippets in the order that they will be in.

00:30:49.000 --> 00:30:53.000

In the final document or file.

00:30:53.000 --> 00:31:08.000

So you could emphasize how a human might walk through code to make sense instead of how a computer needs to walk through it, which I find to be really cool and you can comment, everything, all along the way.

00:31:08.000 --> 00:31:13.000

It's also encourages reproducible research, as well as open access.

00:31:13.000 --> 00:31:23.000

And so some of the tools and you may have heard of some of these are things like no web that's sort of the main syntax for

00:31:23.000 --> 00:31:35.000

for doing linear programming. There's also the tool literate. There's pi web. There's Emacs or demoed which is what I use. And what I will show you but you don't have to use that if you decide to put this route.

00:31:35.000 --> 00:31:54.000

There's code braid. and then there's of course, Jupiter notebooks. A lot of people doing any sort of Python programming these days are doing in Jupiter notebooks which, if you've never seen a Jupiter notebook before, basically what it is is it is alternating

00:31:54.000 --> 00:32:16.000

chunks of markdown code like markdown chunks and Python code chunks. So you can sort of talk about and this is how you do this and this is what, and these are the results you might expect, and here's more resources, and then the code chunk itself.

00:32:16.000 --> 00:32:30.000

And instead of that code chunk just being like an example like you might see in like a GitHub read me. You can actually run the code within that code chunk, and that is sort of what the point of literate programming is, is this isn't just here let's put

00:32:30.000 --> 00:32:48.000

code snippets within our documentation it's know the code within our documentation is the code itself, and we can run it from within the documentation, and we can compile it from the documentation it's really, it's a really cool thing.

00:32:48.000 --> 00:33:02.000

And what you can do is you can actually combine these methods and tools. So you could host it is no garden or multiple gardens and you can use whatever tool you want in a get repository or even just as a website if desired.

00:33:02.000 --> 00:33:20.000

And you can include any sort of literate programming documents in that digital card and so you can sort of for like it. Today I learned things like you could have those be seeds and your digital garden or, you know, notes on your website and you can link

00:33:20.000 --> 00:33:33.000

them together and you could even make those literate if you wanted to, but then you can also be like, Oh, I wrote this documentation for work. And here, here it is and I'm going to share it with people so that they can see.

00:33:33.000 --> 00:33:50.000

All right, so before I open up for questions. I am going to do a small demonstration of what exactly I mean, So let me get my, my dorky Emacs up and going.

00:33:50.000 --> 00:33:54.000

So, what I have up here

00:33:54.000 --> 00:34:03.000

is I have the oat my cats going to come say hi. Hi Arthur, my cat likes to be in presentations when I do them.

00:34:03.000 --> 00:34:09.000

Oh, sharing his paws I need to bring my sharing window to the front. That's right.

00:34:09.000 --> 00:34:13.000

I'm just going to share my desktop. There we go. Okay.

00:34:13.000 --> 00:34:19.000

So, you should be able to see my, my Emacs here.

00:34:19.000 --> 00:34:42.000

And this is something that's like Visual Studio code. So, and in my repository right now for the University of New Hampshire's Primo explore views. And so let's say in my little test explore view that I have that I, I wanted to

00:34:42.000 --> 00:35:06.000

maybe install the unpaid wall extension or something so let me create a new document, quote, and because this is just like a markdown file it's a it's an org mo document that it's kind of the same thing as a markdown it's a plain text.

00:35:06.000 --> 00:35:13.000

Here is where I will install. The

00:35:13.000 --> 00:35:22.000

install the unpaid extension or our explored.

00:35:22.000 --> 00:35:37.000

And I could actually run the shell, like the the command line code from within here so let's see, and I want to tangle.

00:35:37.000 --> 00:35:47.000

So let's say I want to I'm using your so yarn ad.

00:35:47.000 --> 00:35:55.000

And I believe it's Primo explore page. Hey wall.

00:35:55.000 --> 00:36:10.000

And if I think I need to do first is actually do like

00:36:10.000 --> 00:36:18.000

random code so let's see, that's going to work, like computers being a little slow.

00:36:18.000 --> 00:36:35.000

So you'll see this package dot JSON, that wasn't there before, is now there and look at that. It includes Primo explore and pay wall. I did not write that I didn't go to my terminal once it, it did it for me.

00:36:35.000 --> 00:36:38.000

So we have there.

00:36:38.000 --> 00:36:44.000

So now let's say I want to configure that.

00:36:44.000 --> 00:36:55.000

Now we need to put our code in our main.

00:36:55.000 --> 00:37:04.000

js, because I'm using browser by JavaScript.

00:37:04.000 --> 00:37:15.000

So what I'm going to do is, I am actually going to take it from

00:37:15.000 --> 00:37:19.000

the one from our main view.

00:37:19.000 --> 00:37:24.000

So, I can.

00:37:24.000 --> 00:37:29.000

And then I want to make sure I tingle this to the right file.

00:37:29.000 --> 00:37:47.000

And this is just relevant to the current file path. So I know that I am going to need to import Primo explore and hang on.

00:37:47.000 --> 00:38:13.000

And I am going to me too. Oh god bless you Arthur, my cat sneezing, I'm sorry, import some configuration for that I will copy over, and that I need to

00:38:13.000 --> 00:38:24.000

sort of import it here really quick. So then it's new lab, and pain.

00:38:24.000 --> 00:38:27.000

Yeah.

00:38:27.000 --> 00:38:47.000

And then finally, we have to do our app dot constant and paywall can fig.

00:38:47.000 --> 00:38:49.000

Okay.

00:38:49.000 --> 00:38:54.000

So, If I

00:38:54.000 --> 00:39:00.000

save this.

00:39:00.000 --> 00:39:03.000

And we go into my JavaScript.

00:39:03.000 --> 00:39:06.000

You'll see.

00:39:06.000 --> 00:39:19.000

Would you look at that, it wrote it for me, which I find cool oh I did I totally not write it right.

00:39:19.000 --> 00:39:36.000

But you'll see like this wasn't in here before. And this was compiled from that other document that I that I had, which, you know, if I wanted to change this.

00:39:36.000 --> 00:39:46.000

I don't actually have to change it in that document, I could,

00:39:46.000 --> 00:39:49.000

I could change it here.

00:39:49.000 --> 00:39:56.000

So I'm just added a little comment there.

00:39:56.000 --> 00:40:05.000

They see, when I saved it tangled it, and it created a little comment, within the document there.

00:40:05.000 --> 00:40:16.000

And you could use this with the reason why I like this over something like a Jupiter notebook is that this is not just limited to one language I can do this for all of the JavaScript in my development environment.

00:40:16.000 --> 00:40:29.000

I can do it for all the HTML, for the CSS for the CSS for any Docker files, I might be doing because I Docker eyes, our environment, because I followed with some other people had done.

00:40:29.000 --> 00:40:42.000

I could do this for all the command line that I, the command line commands that I might need to do as part of working with this development environment.

00:40:42.000 --> 00:41:04.000

And it all will do it for me that I might will only have to change it in one document and I can, I can explain exactly what I'm doing and walk away from here. And, you know, any sort of regular text that I put in this document isn't going to show up in

00:41:04.000 --> 00:41:11.000

that JavaScript document just what I put in the little in the little code blocks there.

00:41:11.000 --> 00:41:35.000

So let me, let me open up things for questions, and that's all I have and I wanted to make sure I gave plenty of time for questions so that's all I have for people today, you can feel free, feel free to email me, Jay kolbert@unh.edu, my GitHub is Jo Colbert,

00:41:35.000 --> 00:41:44.000

and if you want to follow me on Twitter, where it's not a professional Twitter at all but I do talk about things sometimes it's underscore wild like Oscar Wilde at heart.

00:41:44.000 --> 00:41:48.000

Thank you.

00:41:48.000 --> 00:41:51.000

Thank you, Jay. Very interesting.

00:41:51.000 --> 00:42:02.000

Just reminder if anyone wants to ask a question, you can just raise your hand using the reactions link at the bottom, and then we can ask you to unmute, or you can send something to chat.

00:42:02.000 --> 00:42:23.000

And I see that Mary Roche Roche posted about the library workflow exchange yes that is exactly the thing that I was talking about, about people putting in information about what their libraries have been doing especially the technical services, and putting

00:42:23.000 --> 00:42:34.000

in the code that they did or how they did it or what their workflow or documentation so if you haven't looked through this library workflow exchange, please do it will save your life.

00:42:34.000 --> 00:42:52.000

It's incredible. Shout out if anyone works on on that.

00:42:52.000 --> 00:42:59.000

And I can stop sharing too so I can see to see if anyone has their hand up.

00:42:59.000 --> 00:43:12.000

Did you find that that there's any extra challenges with almond prima, because they do such a steady job of updating the systems,

00:43:12.000 --> 00:43:21.000

sometimes, especially right now my institution is running into

00:43:21.000 --> 00:43:34.000

about what version of unpaid well we do because Primo Alma Primo has two ways of integrating and paywalls stuff into your discovery layer.

00:43:34.000 --> 00:43:46.000

And they just added one recently so the second one is, but you have no control over how those work, or really kind of changed the labeling on them, I believe in whether or not you have them turned on.

00:43:46.000 --> 00:44:01.000

And so sometimes I run into difficulties because they might add a feature that I've been wanting, but then the way they implement it they give you no way of customizing it, and any documentation around it can be pretty frustrating.

00:44:01.000 --> 00:44:04.000

So,

00:44:04.000 --> 00:44:13.000

in primo ve, the way that you can customize things is actually pretty limited compared to the back office, if I'm understanding correctly.

00:44:13.000 --> 00:44:16.000

So for example the like the fetch item.

00:44:16.000 --> 00:44:18.000

the citation link or.

00:44:18.000 --> 00:44:28.000

The only configuration that I can do for it is basically whether or not it shows up in the top menu links, and then I can change the labeling on it.

00:44:28.000 --> 00:44:45.000

But it wasn't working with dry searches, and we didn't know how to like the fact that you had to go in and put in across rough account information. In all, that's not in the Primo part of the documentation, the guy who sort of does a lot of our own this

00:44:45.000 --> 00:44:50.000

stuff, because we don't have one unified systems person.

00:44:50.000 --> 00:45:07.000

It was him doing a lot of googling and trying to figure things out to go oh we actually have to put in a cross roof account in order to get DDOY searches, especially in the, the fetch item to work because none of that information is in the the Primo side

00:45:07.000 --> 00:45:22.000

of things. So sometimes with all of the updates, it, they, they don't always give you information on how you might customize it or what you can customize or what you're in control of beyond just here's how you implement it.

00:45:22.000 --> 00:45:41.000

And that can cause some challenges, in my experience, I do like the roadmap, though that's always been really helpful.

00:45:41.000 --> 00:45:45.000

Any other questions Hands up.

00:45:45.000 --> 00:45:57.000

Any chance not seeing anything Oh Hi Lisa, I see one of my colleagues.

00:45:57.000 --> 00:46:04.000

Well, I guess if there are no further questions we can wrap up, and

00:46:04.000 --> 00:46:10.000

lot of Thank you, no questions at this point for us now I am, I do.

00:46:10.000 --> 00:46:26.000

Just because I, I read it recognize that it can be kind of hard to follow along with that sometimes I actually did make a little,

00:46:26.000 --> 00:46:39.000

a little like this is how I get my little literate programming stuff set up on my computer if people are interested in getting into that. Do I host a Jupiter server locally.

00:46:39.000 --> 00:46:48.000

No I haven't because I haven't played with Jupiter, that much because I use. org mode, and there's no need for me to use it.

00:46:48.000 --> 00:47:00.000

And most times if I followed along the tutorial, the Jupiter notebook has been in, I think it's binder, or something so I can follow along on online but no I don't host.

00:47:00.000 --> 00:47:13.000

I don't have any Jupyter things set up, but this is how I get my, my Emacs org mode. org Babel, is the literate part of.

00:47:13.000 --> 00:47:43.000

org mode in Emacs if people are interested in tinkering and getting Emacs to set up on their computer I recognize that can be hard, but I made sure I made that repository with some examples and instructions on that.

00:47:47.000 --> 00:47:55.000

Did I start a today I learned file I technically have a repo on GitHub.

00:47:55.000 --> 00:48:03.000

But what I've switched to instead is I, I have started making my own digital garden, actually.

00:48:03.000 --> 00:48:09.000

And the digital garden is going okay. The only thing about it is you have to remember to update and maintain it.

00:48:09.000 --> 00:48:19.000

So let me actually I'll just share my, my digital garden with with with folks.

00:48:19.000 --> 00:48:26.000

And, yeah, so that's my little garden.

00:48:26.000 --> 00:48:44.000

I recently took a bunch of notes while reading a book. So sort of like I was reading and these were all the sort of comments and commentary I had while I was reading it and they made that a, like a post in my, my digital garden.

00:48:44.000 --> 00:48:57.000

So that's sort of instead of doing a, today I learned repository, I am just doing that but as I get a digital garden so not just today I learned, sort of any sort of ideas or thoughts that I have.

00:48:57.000 --> 00:49:01.000

But I'm meaning to put more of my premium stuff in there.

00:49:01.000 --> 00:49:12.000

But I'm just getting started on it.

00:49:12.000 --> 00:49:19.000

Alright, I guess we can at the break and can you can go take a very cold thank you for

00:49:19.000 --> 00:49:21.000

not feeling good.

00:49:21.000 --> 00:49:32.000

Yeah, it's it's it's not co bid or an ammonia, it's allergies, so. But thank you.

00:49:32.000 --> 00:50:02.000

All right, then we'll take our break and next session will be starting in eight minutes at two o'clock.