

Open Education Network Tea Time

GoDot2P: A FLOSSy package to help move OER developed in Google Docs to Pressbooks ...and some Thoughts on OER platforms

Jonathan A. Poritz

jonathan@poritz.net
poritz.net/jonathan



6 November 2023
Open Education Network Tea Time



This slide deck, except where otherwise indicated, is by [Jonathan Poritz](#) and is released under a [Creative Commons Attribution-ShareAlike 4.0 International License](#). This version: 07 Nov 2023 00:19CET
These slides, also in editable form, are available at poritz.net/jonathan/share/OENTTGoDot2P/.

I live now in a place where there is not a tradition of giving land acknowledgements, but I have that habit from many years before I moved here, so I want to say:

I am grateful the chance I had to live and learn, and build some part of the foundation on which was built the work I will describe today, on the unceded territory of the Ute Peoples. The earliest documented people in that area also include the Apache, Arapaho, Comanche, and Cheyenne. An extended list of tribes with a legacy of occupation there can be found on the [Colorado Tribal Acknowledgement List at Colorado College](#) – you can find the URL by searching for that phrase.

BTW, “FLOSS” in the subtitle stands for “free/libre/open-source software.”

I am very much looking forward to this session because it seems like my life in recent weeks or even months has been a no-holds-barred attack on technology, and I feel like I'm in danger of becoming a crotchety old man shouting at the youngsters to get off my lawn with your damn gadgets – a Luddite. I even was on a panel yesterday at Lucca Comics and Games, the third largest comic-con on the planet, arguing with a member of the European Parliament about how AI doesn't actually do all the amazing things it is said to do, and instead is begin used today to “grease the slide into dystopia,” [to use a phrase I started shouting at the Creative Commons Summit last month].

But: actually, a new book – *Blood in the Machine*, by Brian Merchant – argues that the Luddites were not against technology and in fact were enthusiastic about some technology, they just didn't like technologies which took autonomy and agency away from people and were used to concentrate power and wealth among the few ... so maybe I am a Luddite!

Luddite or not, today's talk is about a bunch of software I wrote to help humans do a particular task in a particular situation ... because I love technology that empowers humans. As a paper bag at Lucca C&G yesterday said: “Proud to be a Nerd.”

Overview: The Project and the Plan

The Project:

Creating/adapting OER in Google Docs (hereafter **GD**) but sharing them with students in Pressbooks (hereafter **PB**) seems to be a fairly common workflow in our community.

Today we'll discuss a particular experience of trying to ease – and partially automate – that workflow. Not because it is the best, but simply because it is a case study of the details of how to make it work ... and it has resulted in some (open, of course!) tools that might be useful to others who use this – or even other, related – workflow.

The Plan:

- Who
- What
- How
- Why

The most useful part of this to folks doing, or planning to do, the same or a substantially similar workflow, is probably the **How**.

But the **Why** is probably the most fun, and I'd love to get folks thoughts – maybe we can have a discussion!

A friend of mine at my former institution wrote an OER structured to begin every chapter with a case study: she thought a concrete example was the best way to introduce a new concept, every single time .., in every textbook on every subject.

While I don't know that I agree her approach is *always* the best, I'm going to follow it today, talking about a case study of the GD~PB workflow.

This outline is definitely not the most original, but it just seemed to be the way it was coming together, so I decided not to fight it too hard...



is an amazing team. Every person I know in that organization is skilled, thoughtful, humane, talented, effective, ethical, and knowledgeable.

Some names: Amy Hofer, Veronica Vold, Phoebe Daurio, Erin Wells, Coleen Sanders, Taylor Wynia, Liz Pearce, Michaela Willi Hooper [not a complete list, my apologies!].

In June of 2022, they were looking for a person to be “Open Education Instructional Technologist.”

Really amazing! I've had the privilege of working with many OER projects and teams – including

- some through the OEN,
- many through the Colorado OER Council,
- a few through the Rebus Foundation,
- a handful through OE4BW,
- a great many (309 students total!) through the CC Cert course,
- and several through other random situations

– and OO is one of the most amazing organizations, with the best people and the most amazing track records, with whom I've had the chance to work!

Slide 3/ 25

Who ... or maybe *Where*

In June 2022, we were packing up this

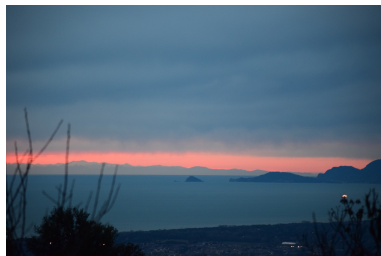


and quitting our jobs¹,

to head to this



and this



and looking for remote gigs in the open movement ... so that was perfect.

¹ Tenured jobs ... it was a little crazy.

That was our by then empty nest in Colorado Springs, CO, USA. Not a bad place ... a lot of work to get a house built in 1896 ready to sell, though!

[The daughter of our landlord in Italy asked if houses in the US are “so new” because of all the tornadoes ... I think she had recently seen a documentary about Tornado Alley in the US, or else had watched *The Wizard of Oz*.]

Now we live in a house that could have been designed by M.C. Escher: it has stairways that lead up to places lower than where you started, I am convinced. And in the winter, when the leaves fall off of the chestnut trees and the air is clear, we can see Les Alpes Maritime of the coast of France – we know our selves to be **very lucky**.

And in case any of you are thinking of burning your lives to the ground and moving to Italy, I can tell you that our cost of living is much, much lower here than it was in Colorado ... even including the cost of the medication I take regularly, which is 43¢ a month.

What

OO's plan was that teams of authors would build "openly-licensed, targeted pathway materials with an equity lens" for about a dozen courses.

In Google Docs.

These books would be transferred to Pressbooks by their Open Education Instructional Technologist.

Some of the GD and/or PB versions would be shared with students and used in the classroom.

Based on peer, accessibility, and pedagogical review as well as student feedback, new versions of these books would be developed in GD.

New PB conversions could be made of some or all of the revised books, if it would help in the revision process.

A final PB version would be made from the final GD version, using some custom stylistic choices made in collaboration with a book designer.

I apologize that I am not perfectly clear on all of the details of the back-and-forth of this process, and who would be responsible for what, when.

For example, I know that Chemeketa Press “the academic publishing arm of Chemeketa Community College in Salem, Oregon” is playing an important role, and I think there was an separate book designer who made a wonderful plan for how the final version of the book would look.

But from my point of view, as Open Education Instructional Technologist, I have tried to narrow my focus to the simplified version on the next slide.

What: Simplified

Given a textbook prepared in GD, get it into PB as quickly, accurately, and painlessly as possible.

"Quickly" and "painlessly" because this was, as we can see from the previous slide, going to happen many times, for the dozen or so OO OER.

"Accurately," in that we needed to preserve essentially all of the information put into GD versions, and get it into appropriate places and formats for the PB versions as desired by OO.

What: OK, that was too simple – GD style

OO has some particular practices for how to organize the GD versions of books: a style guide, which said things like, to cite just a very few:

- embedded images should have alt texts, in the usual GD way;
- but some images require longer “image descriptions” rather than a simple alt text, which should be in separate, stand-alone GDs
- chapters should be **Heading 1s**, sections **Heading 2s**, *etc.*
- chapters have numbers, as do figures, but not sections, subsections, *etc.*

This is all aspirational – humans are not good at being accurate, methodical, and consistent.

For example, it's hard to get authors to write alt texts, because (it seems) they don't like to slow down their writing to do that – but alt texts must be written by the authors, because they describe the pedagogical purpose of each image in the text, it is essentially impossible for outsiders to create them

It's also easy to use an incorrect heading level with a [sub][sub]section title, particularly because these are chosen by a pull-down menu in GD on which an author might make a mistake.

Likewise, it's easy to forget to renumber figures, or to do so incorrectly, when moving bits of text around in a book, as authors often do.

What: OK, that was too simple – PB style

On the PB side, there are many choices to be made: instructional designers, book designers, and overall editorial oversight will result in a “house style” ... which may be viewed as confining by some authors, in some situations.

OO house style for intermediate PB versions includes

- a choice of base PB theme: **Malala**;
- some customizations of this theme mostly around spacing (easy to do with PB’s “custom CSS.”)
- a decision to make GD *Chapters* become PB *Parts*, GD *sections* $\rightsquigarrow$ PB *chapters*, GD *subsections* $\rightsquigarrow$ PB *sections*, etc. ... this was to reduce the amount of scrolling students would do when looking at a typical textbook reading assignment;
- choices of some textbook artifacts – such as sub[sub]sections of *Learning Objectives*, *Key Terms*, *Activities*, *Licenses and Attributions for <section name>* – which would PB textboxes of various flavors;
- how to handle long image descriptions;
- what to do with PB **glossaries**.
- etc.

I have a lot of sympathy for the authors who feel uncomfortably constrained by house style: one of the reasons I wrote the OER I've written – three that anyone would call “textbooks” and two to four that are quite different beasts – is a kind of obstinacy about wanting my damn class to go my damn way, which, once you let it overwhelm you (and you need *some* strong emotion to drive you to do all the work of making your own course materials ... it's certainly not the pay!), is hard to suppress just to go along with the house style someone else wrote!

Side note from a non-US perspective: the disease of textbook cost increases **is not global!** This is hard for Americans to get their heads around, but it is true: textbooks (and education itself, and healthcare, and good food, *etc.*) are not insanely expensive in some (maybe most) countries. Therefore, the “come for the cost, stay for the quality” slogan which I was so used to quoting when spreading the word about open education in the US **is a non-starter** in much of the world.

Instead, in a number of places on the planet, one has to lead with academic freedom and encouragement of the kind of educator individualism that makes a rigid house style particularly unappealing.

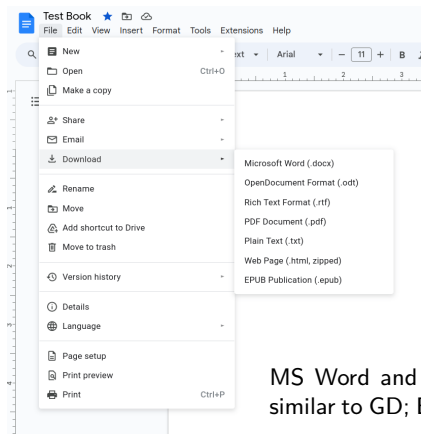
I don't know that I feel the need to eschew long scrolling pages as strongly as the OO folks – my OER often have long (even book-length!) scrolls, and, arguably, traditional books have pretty long scrolls... but of course it was OO's decision!

Long image descriptions were turned into linked, page-level footnotes with a return link, following a strategy described by BCcampus.

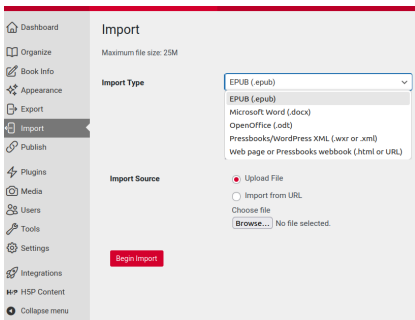
Glossaries are a surprisingly complex issue, which I'd be happy to talk about if there is interest.

How: Surely this must be nearly automatic

GD export (“download”):



PB import:



MS Word and OpenOffice are quite similar, and both are similar to GD; EPUB is a bit different, we rejected it quickly.

So: why not export to Word from GD and just import that Word document to PB?

Perhaps I was a coward not to look deeper into EPUBs – they just seem to me to be yet another detailed technology for which there are not great FLOSS tools to manipulate directly.

How: Problems with GD↔Word↔PB

Experimentation with this possible workflow found that the alt texts of images were being lost along the way!

Additionally, it was very unclear how to do the other OO house style customizations, such as GD *Chapters*↔PB *Parts*, textbook artifacts becoming particular PB textboxes, long image descriptions, *etc.*

Of course, it could all be done by hand, pointing-and-clicking until the cows come home. But *computers are supposed to do annoying, repetitive tasks for humans, **not the other way around!***

I think the loss of alt texts was happening in the GD export. This goes against what we tend to think would happen: just as it is hard not to assume that a more expensive textbook is “better” than a free one, it is hard to believe that the products of one of the largest and richest companies on the planet (market cap as of the time of this writing of around **\$1.6trillion!!!**) would lose important metadata that is vital for a11y during an export process.

It is a mistake to think of Google (or any of those plutocratic top four or five companies) as obviously not making mistakes. How could they, when they can afford the top talent on the planet? (Actually, they can afford the top talent *that is for sale* on the planet... maybe not everyone is for sale.) *E.g.*, if your OER work involves using YouTube videos, it might interest you to know that YT's version of Creative Commons licenses is **absolutely terrible**: information pages about CC licenses contains straight up false information; the licensing statement it uses is not very well constructed; and the link from that licensing statement makes it clear that YT's CC license is only [CC BY 3.0](#). But one of the main improvements when CC went from v3.0 to 4.0 is a feature which pretty much eliminates copyleft trolling ... so it is vital that we use v4.0 licenses!

Note that the desire expressed in this slide to have computers “do the boring part” is something that is very hard to do through pointing-and-clicking interfaces ... or at least it seems to be difficult. In any case, this gives me an excuse to do one of my favorite rants!

Digression 1: GUI vs CLI

GD – also Word, OpenOffice (LibreOffice), PB, Windows, Mac, iOS, Android, and most other programs used hundreds of times a day by billions of humans – all employ a **Graphical User Interface [GUI]**: all that “pointing-and-clicking until the cows come home.”

The other way humans control their computers is with a **Command-Line Interface [CLI]**. Using a CLI, a human types words into a “terminal”, with options and inputs and outputs specified in that text.

CLIs are much better than GUIs

At least, for certain tasks. Think about it: when you go into a classroom to explain integration by parts or Marx’s labor theory of value, you don’t do charades, you use words.

If you want your computer to do something complicated and powerful – as it can, although maybe it isn’t convenient for the company who sold it to you, or who sold you your software that you realize this – use words when you give it instructions, don’t play charades with it!

I did say “for certain tasks.”

A police officer directing traffic usually just points to where cars should go, and that is probably more effective than having each one stop, roll down the window, and describe where to go. (Although stopping would probably be done with a gesture, and we could discuss the semiotics of the officer’s uniform....)

Similarly, for years when I shut down my computer, I would go to a “terminal window” – the main place people who want to use their computers for real work communicate with those computers – and type

```
shutdown -h now
```

That works fine, but I do it often enough, and I have enough screen real estate, that I put a button  up near the top of the screen (near other single-purpose buttons) which does that same command automatically, when it is clicked with the mouse.

There are plenty of other tasks that are easier to do with a GUI than with a CLI ... but the project of GUIfying everything is about taking power away from the users of computers. It’s as if reading and writing were invented, and only governments, large corporations, and a few crazy anarchists learned how to do it, while everyone else was told it is too complicated for the masses, and you need to go to a specially trained person to read or write for you.

Digression 1 (cont.): CLling the GUI

GUIs as an instrument of oppression are so prevalent, that the people who do the actual work of making them, who certainly won't weaken their control by using them, must have a CLI way of driving a GUI.

In Python, there is a package called **Selenium** can do all the things humans do when operating a GUI – click on links, enter text into boxes, choose something from a pull-down menu, *etc.* – under the control of a program.

Actually, Selenium does this for web sites, not general GUIs: it couldn't directly operate Word or LibreOffice, for example.

Selenium is mostly used to test web sites, *e.g.*, stress-testing them by automatically operating at super-human speed, as if many requests came into a website nearly simultaneously from all over the world.

In principle, we could use a tool like Selenium to go directly from GD's GUI to PB's GUI, driving the interaction with code (the ultimate CLI).

I mostly program in Python these days, it's a really nice language. All FLOSS, of course, as are packages like Selenium and many others I use.

Having heard my rant on CLIs vs GUIs, you will not be surprised that merely automating the GUI interaction did not appeal to me. I do use Selenium later in the workflow, but in a very limited way.

How: Get the OER out of GD as HTML

The way which seems to offer the most possibility for automation, not losing important ally information (alt texts), and leaving it in a form that enables straightforward manipulating to realize OO's house style is to export from GD using the option "Web Page (.html, zipped)".

This results in a zip (compressed) file, which when unzipped reveals a folder of image files and a single, large HTML file, which we might as well call `book.html`. It's ugly:

```
<html><head><meta content="text/html; charset=UTF-8" http-equiv="content-type"><style type="text/css">@import url(https://themes.googleusercontent.com/fonts/css?kit=0PeqXG-QxW3ZD8BtmPikfA);.lst-kix_qpcrjg12wdx5-3>li:before{content:"" counter(lst-ctn-kix_qpcrjg12wdx5-3,decimal) ". "}.lst-kix_5iabqo6dk17m-5>li:before{content:"\0025a0" }ul.lst-kix_qv847ouxndzf-2{list-style-type:none}ul.lst-kix_qv847ouxndzf-1{list-style-type:none}ul.lst-kix_qv847ouxndzf-0{list-style-type:none}.lst-kix_6orila2j8o67-2>li:before{content:"" counter(lst-ctn-kix_6orila2j8o67-2,lower-roman) ". "}.lst-kix_7qiebeeinotp-8>li:before{content:"\0025a0" }.lst-kix_zh0djvhcuw11-5>li:before{content:"\0025a0" }.lst-kix_qpcrjg12wdx5-7>li:before{content:"" counter(lst-ctn-kix_qpcrjg12wdx5-7,lower-latin) ". "}.lst-kix_ymzhtletofw0-0>li{counter-increment:lst-ctn-kix_ymzhtletofw0-0}.lst-kix_h0qa59x4hax3-1>li:before{content:"\0025cb" }.lst-kix_e9xxj2arum8r-7>li{counter-increment:lst-ctn-kix_e9xxj2arum8r-7}.lst-kix_88guu1ohcsd4-1>li:before{content:"\0025cb" }ol.lst-kix_2g5fjno92eyf-0.start{counter-reset:lst-ctn-kix_2g5fjno92eyf-0 0}.lst-kix_jfrs6xlc6t05-2>li:before{content:"\0025a0" }.lst-kix_6orila2j8o67-6>li:before{content:"" counter(lst-ctn-kix_6orila2j8o67-6,decimal) ". "}.lst-kix_6omuljmxba6-1>li:before{content:"\0025cb" }.lst-kix_78u85j5w8j7h-0>li:before{content:"\0025cf" }.lst-kix_78u85j5w8j7h-8>li:before{content:"\0025cf" }.lst-kix_h0qa59x4hax3-5>li:before{content:"\0025a0" }.lst-kix_wtp5ial5q6ui-0>li{counter-increment:lst-ctn-kix_wtp5ial5q6ui-0}.lst-kix_zh0djvhcuw11-1>li:before{content:"\0025cb" }ul.lst-kix_qv847ouxndzf-6{list-style-type:none}ul.lst-kix_qv847ouxndzf-5{list-style-type:none}ul.lst-kix_qv847ouxndzf-4{list-style-type:none}.lst-kix_5iabqo6dk17m-1>li:before{content:"\0025cb" }ul.lst-kix_qv847ouxndzf-3{list-style-type:none}.lst-kix_jfrs6xlc6t05-6>li:before{content:"\0025cf" }.lst-kix_bxpnqgj9tbtn-3>li:before{content:"\0025cf" }.lst-kix_bxpnqgj9tbtn-7>li:before{content:"\0025cb" }ul.lst-kix_qv847ouxndzf-8{list-style-type:none}ul.lst-kix_qv847ouxndzf-7{list-style-type:none}ul.lst-kix_78u85j5w8j7h-1>li:before{content:"\0025cb" }ul.lst-kix_78u85j5w8j7h-7>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-8>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-9>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-10>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-11>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-12>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-13>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-14>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-15>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-16>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-17>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-18>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-19>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-20>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-21>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-22>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-23>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-24>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-25>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-26>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-27>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-28>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-29>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-30>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-31>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-32>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-33>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-34>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-35>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-36>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-37>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-38>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-39>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-40>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-41>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-42>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-43>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-44>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-45>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-46>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-47>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-48>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-49>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-50>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-51>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-52>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-53>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-54>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-55>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-56>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-57>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-58>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-59>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-60>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-61>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-62>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-63>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-64>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-65>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-66>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-67>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-68>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-69>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-70>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-71>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-72>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-73>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-74>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-75>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-76>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-77>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-78>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-79>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-80>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-81>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-82>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-83>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-84>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-85>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-86>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-87>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-88>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-89>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-90>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-91>li:before{content:"\0025cf" }ul.lst-kix_78u85j5w8j7h-92>li:before{content:"\00
```

That particular book goes on for 1.6MB of HTML ... all on one line! And filled with HTML features like classes and anchors whose names are meaningless alphanumeric strings... argh!

How: Let's tidy things up a bit

Folks at the World Wide Web Consortium long ago wrote a program (released as FLOSS, fortunately) which makes ugly HTML look better. It even has a mode that can fix some of the particularly bad features Google puts into its HTML files. Here's what that file looks like after we apply the tidy program:

```
<!DOCTYPE html>
<html>
<head>
<meta name="generator" content="HTML Tidy for HTML5 for Linux version 5.6.0">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title></title>
</head>
<body id="t.3">
<h1 id="h.gjdgxs">Chapter 1: Introduction: The History of Mental Disorders</h1>
<p>[box]</p>
<h2 id="h.1fob9te">Learning Objectives</h2>
<p>The following learning objectives tell you what's most important in this chapter. Use these statements as a study guide to make sure you get the most out of this chapter.</p>
<ol start="1">
<li>Describe early treatment of mental disorders that illustrates the history of abuse and stigma around mental illness and disability.</li>
<li>Discuss modern developments and reforms in the treatment of people with mental disorders.</li>
<li>Evaluate the consequences of institutionalization and deinstitutionalization of people with mental disorders.</li>
<li>Recognize discrimination against people with disabilities, and specifically mental disorders, as a barrier to treatment and progress.</li>
<li>Recommend criminal justice reforms that may impact people with mental disorders.</li>
</ol>
<p>[/box]</p>
<p>[box]</p>
<h2 id="h.aoycyemv93xp">Key terms</h2>
```

Now we're down to 969KB, and it's much more readable, e.g., being split over 3,269 lines.

But still there are those

```
id="t.3"
```

and

```
ol start="1"
```

and

```
id="h.aoycyemv93xp"
```

which are a bit unpleasant to look at. The rest is fine HTML.

Digression 2: Why are we afraid of markup languages?

When one learns a new subject, one learns many terms of art – often normal words which have been given particular meaning in the context of that discipline.

A **markup language [ML]** does a similar thing, using words and symbols that have other uses to control appearance – whether a bit of text is italics or bold – and semantics – whether a bit of text is a level 1 heading or level 2 or even perhaps a link to some particular URL.

As part of the GUIfication of nearly all human-computer interactions, there has come to be a kind of fear of markup languages.

I don't understand this fear, particularly in higher ed: when I teach number theory, I define prime numbers, what are twin primes, and we marvel together that the Twin Prime Conjecture is still open even though Euclid, 2,300 years ago, could have understood it.

Why then can I not tell my students, who live large parts of their lives on World Wide Web, that in a web page, `<ol>` starts a numbered list, `</ol>` ends it, and within the list, `<li>` starts a list item and `</li>` ends the item?

I do, in fact, often teach my students just a little HTML like that.

Phoebe Daurio, the OO person with whom I worked most directly in the last year (but who, unfortunately for OO!, left for another position elsewhere a couple of months ago), and I used to joke about how “non-programming humans” don’t seem to like markup languages, and we shouldn’t force the OO authors to do anything that looked at all ML-like.

Note that there is markup language called **Markdown**, which I think was originally envisioned as a limited subset of HTML, that would be easier and friendlier for non-programming humans to use. But actually, I believe, it has gradually grown to the point that it is pretty much as powerful as HTML, while still remaining easy and friendly. There are publication systems which use markdown, and I think that is a very good way that OER teams might want to use in the future – it’s friendly, but at the same time gives a huge amount of control.

Actually, in the end we did, and you might have noticed the [box] and [/box] in the tidied html book I showed before: that is essentially a little bit of ML that we’re asking OO authors to use when they want the text those marks enclose to be in a textbook when moved to PB.

Digression 2 (cont): documents in ML are just text

The irrational fear of markup languages means that most of the time we work with them in WYSIWYG word processors for documents, such as Word and LibreOffice, and for web pages, like the create/edit page GUI in PB (or its parent program WordPress).

This is a little like doing surgery wearing mittens. Yes, those operating rooms can be over-air conditioned, but having comfortable fingers is probably not worth the loss of sensation and dexterity.

As a specific example of what WYSIWYG mittens sacrifice: suppose you want to search for all level 3 headings in GD, or for a *License and Attributions* subsection of an OER which might be at the wrong heading level ... you're out of luck: GD has no way to search for those things! When someone uses WYSIWYG to hide the real ML which is there behind the scenes, you are relying upon them to give you the tools to every manipulation you might want to apply to your document, and we have seen that relying upon Google's technical excellence is a mistake.

When we work with OER as HTML or markdown or some other ML, we can always get that fine control ... and even easily automate manipulations with code.

“WYSIWYG” here stands for “What You See Is What You Get.”

Among programming humans, there is a distinction between a “word processor,” which WYSIWYGs over the ML, and a “text editor” (or just “editor”), which simply works with the text.

How: Let's deal with images

With the images from an OO OER in GD downloaded and unzipped into a folder on my laptop, I could use Selenium to upload them to the PB media library.

What about the alt texts? They are metadata not in the images files, but rather in the HTML, attached to each place the image is used. The way this happens is with an `<img>` HTML tag. Fully spelled out with an alt text, it would look like

```

```

While we could extract these alt texts from the (plain text! easy to process with code!) file `book.html` and use Selenium to upload it as part of the media library entry, that's actually the wrong thing to do.

Alt text is a relevant description of how an image fits into the text at that particular point of the exposition for vision-impaired folks. As such, it should be attached to the **use** of the image, **not to the image itself!**

I don't understand why PB has alt texts associated to images themselves....

Our best approach is simply to upload the images (all at once in a big `.docx` file is the fastest way, rather than monkeying with Selenium) and leave the alt texts in the HTML.

My code makes a .docx file which simply contains all of the book's images, and one has to upload it to PB by hand: PB automatically creates media library entries for all images in .docx files it uploads.

In fact, if that .docx file is larger than 25MB, first it must be broken into (maybe two or three) smaller pieces, and individually uploaded.

I then download the HTML version of the PB page it made from that .docx upload, which allows my code on my laptop to make a logical connection between the images as they appear in book.html and the references PB has made to its media library.

It's relatively little work – a dozen or so clicks per GD↔PB conversion – so I haven't bother to automate it with Selenium. But that could be done if it were desired.

How: Let's deal with **everything else!**

As we have said, PB under the hood is HTML. We have the book on our machine ... as HTML, so text, so it can be searched and manipulated directly by hand or in code. Want to investigate something about the way the book is structured? Write code to extract and show that thing for humans to think about it. Find something you don't like in the HTML, write a bit of code to run through the whole book and fix it!

Some examples:

- all links in Google docs are redirected through Google – let's just strip those all off for our students' privacy and because they serve no pedagogical purpose;
- internal links have to be rewritten to point to the correct PB URL, not to some strange alphanumeric anchor in the GD version;
- where desired, HTML fragments can be added which will cause particular PB textboxes to appear in precisely controlled situations (this is done with a tiny bit of ML in the GD OER);
- sections, subsections, subsubsections, *etc.*, can be correctly numbered without authors needing to worry about it;
- if authors or oversight teams are comfortable with a little more ML, we can automate figure numbering, so that they will never be misnumbered and authors don't have to worry about counting;
- links to external sites can be checked to see if they are still alive or have suffered link rot;

Google redirects are quite clever: you can't see them when you hover over a link in GD, since they've hacked the way browsers show their links. Nevertheless, the redirects are there – because, of course, Google got to be a \$1.6trillion company by following the approach of surveillance capitalism.

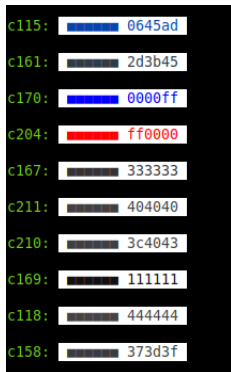
Figure numbering does require a little of ML because there has to be some way to refer to the figure in the text, even if the number is not known by the author at the time of writing.

External link checking can be complicated by the fact that sites don't like to be visited many times in quick succession from the same IP address – that looks like a DOS attack.

How: More of everything else

More examples:

Text can have different colors, when certain HTML tags are present. I wrote some code which finds and displays all of those bits of color-changing HTML



Sometimes it helps to see an outline of the book, with numbering of all sections, subsections, subsubsections, etc.:

```
1 Chapter 1: Introduction: The History of Mental Disorders
1. 1 Chapter Overview and Learning Objectives
1. 1. 1 Learning Objectives
1. 1. 2 Key Terms
1. 1. 3 Chapter Overview Licenses and Attributions
1. 2 Historical Perspective on Mental Disorders
1. 2. 1 Early Treatment of Mental Disorders
1. 2. 2 Mental Disorders in the Middle Ages & Early Modern Era
1. 2. 3 Reforming Treatment of Mental Disorders
1. 2. 4 Historical Perspective on Mental Disorders: Licenses and Attributions
1. 2. 4. 1 Open Content, Shared Previously
1. 3 Mental Disorders in Modern Times: the Rise of Institutional Treatment
1. 3. 1 The Creation of State Hospitals
1. 3. 2 SPOTLIGHT: Carrie Buck and the Practice of Eugenics
1. 3. 3 Racial Segregation in Institutions
1. 3. 4 Treatment of Mental Disorders in the 1900s
1. 3. 5 Mental Disorders in Modern Times: the Rise of Institutional Treatment: Licenses and Attributions
1. 4 Modern Developments: The Decline of Institutional Treatment
1. 4. 1 SPOTLIGHT: The Oregon State Hospital, Then and Now
1. 4. 2 Modern Developments: The Decline of Institutional Treatment: Licenses and Attributions
1. 5 Deinstitutionalization and Disability Rights
1. 5. 1 SPOTLIGHT: Women Who Shaped the Disability Rights Movement
1. 5. 2 Deinstitutionalization and Disability Rights: Licenses and Attributions
1. 6 Mental Disorders Post-Deinstitutionalization
1. 6. 1 Inadequate Community Alternatives
1. 6. 2 Links to Homelessness and Incarceration
1. 6. 3 Mental Disorders Post-Deinstitutionalization: Licenses and Attributions
```

Showing the color-changing HTML is useful so that we can see if the author is using color to convey meaning, a bad a11y practice!

I have used the outliner so many times to figure out when the book had some structural problem, it turns out to be fabulously useful!

How: Last steps

After the middle stage of this work – fixing all issues, links, structural issues, preparing HTML for PB textboxes, etc. – a program 00split chops up the tidybook.html into one file per PB chapter (which, remember, comes from a GD section) and makes a file I call the manifest, giving the instructions for which bits become PB Parts, which PB Chapters, what the names of those will be, etc.

Then a final program 00upload uses the information in the manifest to upload those files one by one, as a human would do it with pointing-and-clicking, but using Selenium.

Example manifest:

```
# At 0:12CEST on 19/10/2023, this was
# /home/kaleb/00/00split -v -p stable/manifest preamble tidy_book.html in directory /home/
kaleb/00/books/mentalcrimjustsysprelaunch which resulted in this file
#CSS: stable/custom.css
PM: About this book
stable/About this book.html
PM: How to Navigate this Book
stable/How to Navigate this Book.html
PM: About The Publisher
stable/About the Publisher.html
Part: Chapter 1: Introduction: The History of Mental Disorders
00html/1.0.html
Part: Chapter 2: Mental Disorders in the Criminal Justice System
00html/2.0.html
Part: Chapter 3: Disability Activism and the Rights of People with Mental Disorders
00html/3.0.html
Part: Chapter 4: Reducing the Criminalization of Mental Disorders
00html/4.0.html
Part: Chapter 5: Law Enforcement Approaches to Mental Disorders
00html/5.0.html
Part: Chapter 6: Mental Disorders in the Criminal Courts: Competence and Insanity
00html/6.0.html
Part: Chapter 7: Incarcerated with a Mental Disorder
00html/7.0.html
Part: Chapter 8: Mental Disorders in Community Corrections: Reentry and Supervision
00html/8.0.html
Part: Chapter 9: Civil and Criminal Commitments
00html/9.0.html
Part: Chapter 10: Opportunities and Challenges in the Behavioral Health and Criminal Justice Fields
00html/10.0.html
Chapter[1]: 1.1 Chapter Overview and Learning Objectives
00html/1.1.html
Chapter[1]: 1.2 Historical Perspective on Mental Disorders
00html/1.2.html
Chapter[1]: 1.3 Mental Disorders in Modern Times: The Rise of Institutional Treatment
00html/1.3.html
Chapter[1]: 1.4 Modern Developments: The Decline of Institutional Treatment
00html/1.4.html
Chapter[1]: 1.5 Deinstitutionalization and Disability Rights
00html/1.5.html
Chapter[1]: 1.6 Mental Disorders Post-Deinstitutionalization
00html/1.6.html
Chapter[1]: 1.7 Chapter Summary
00html/1.7.html
Chapter[1]: 1.8 References
00html/1.8.html
Chapter[2]: 2.1 Chapter Overview
00html/2.1.html
Chapter[2]: 2.2 What Are Mental Disorders?
00html/2.2.html
```

Note the manifest allows one to specify a custom CSS file, Front and Back Matter sections, and the GD Parts and sections.

Image files are already on PB by this point, as explained earlier, and the local HTML files to be uploaded have correct URLs to their locations in the PB site

How: Complete workflow

- 1 In GD: File >Download >Web Page (.html, zipped)
- 2 `% unzip *.zip`
- 3 `% mv *.html book.html`
- 4 `% 00prep -v -b "cX cY" book.html`
- 5 `% 00outliner -n -t -l tidy_book.html` and fix errors
- 6 `% 00img_list -v -o imgs tidy_book.html`
- 7 Import imgs.docx into PB (after splitting into <25MB pieces if necessary)
- 8 Download the resulting PB HTML file(s) and reassemble pieces if necessary into a local file pb_imgs.html
- 9 `% 00split -v tidy_book.html`
- 10 Consider doing `% 00fix_links` and other adjustments to the HTML
- 11 `% 00upload -v 00html/manifest`
- 12 In PB: set any desired Theme Options, insert Book Info
- 13 Enjoy a celebratory beverage of your choice

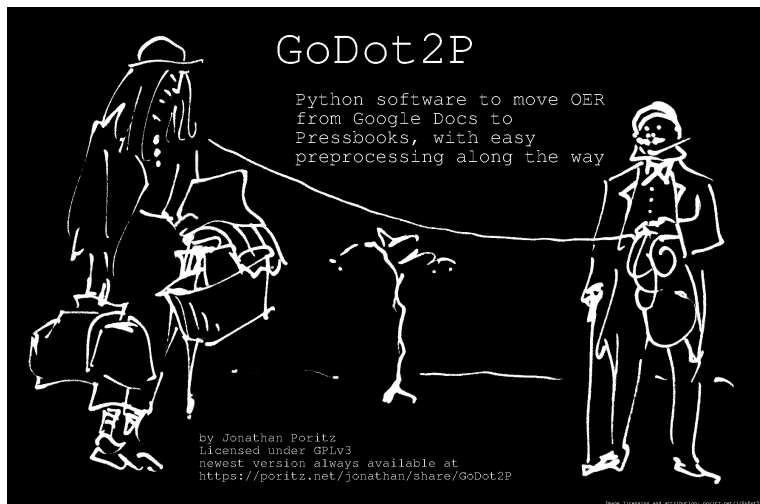
I'm showing you what I would do on my laptop running Linux (of course). Here %
command means run the command in a terminal window (yay, CLI!)

My celebratory beverage is espresso.

Note there is also a program 00download which enables one to download a snapshot of
all the PB HTML, for copying, modifying, etc., working directly with the HTML.

Slide 21/

Ta-da: GoDot2P



² Adapted from Poster for drama performance of "Waiting for Godot" by Fewskulchor released under Creative Commons BY-SA 3.0 Unported; this version by Jonathan Poritz replaces the original text with the name of this software project and other relevant information, and is released under a Creative Commons BY-SA 4.0 Int. license.

Traditionally, one picks a fun name for a FLOSS project. Since this software is all about going from **Google Docs** to (2) **Pressbooks**, I thought of **GoDot2P**. There was then this nice openly licensed image I could modify and use as cover art!

And also when my home network is slow and it is taking a while to upload a large book to PB, I can darkly mutter “is that Godot fellow ever going to get here?”

Why (and why not): GD

Why do people like to compose in GD? Maybe:

- it's a lot like Word ... so it's familiar to many people
- it has good version control ("track changes")
- it has good threaded comments
- ... it comes from a rich and powerful company that we trust?

What are problems with GD as a place to create OER?

- it can't do math
- surveillance capitalism
- not FLOSS ... so who knows what's under the hood and you only get the tools you are given
- does not interoperate nicely with other tools, e.g., incorrectly handling alt texts on export
- it could change any day without warning
- do you really trust the company that had to drop "don't be evil" from it's corporate ethical policy because it was too restrictive?

But what do **you** think? Tell me, I'll take notes! Or put your thoughts in the chat and we'll save it.

I really hate Google, as should be apparent – although I use their products, quite a bit, I confess it!

There are lots of other tools which have all of the advantages cited here, but maybe fewer of the disadvantages.

Why (and why not): PB

Why do people like to share OER in PB? Maybe:

- pretty good ally (I've been told...)
- easy to share, clone, publicize (the PB "Directory")
- it can support embedding video, Knight timelines, *etc.* ... and H5P
- it's FLOSS
- the PB people are really nice humans

What are problems with PB as a place to share OER?

- it can't do math ... not really
- it is FLOSS, but has no active developer community other the company Pressbooks... they could change things and we wouldn't have real option
- interoperate somewhat roughly with other tools, although there seems a big drive to have LMS integration ... because why not connect something good to something stupid, that will make the world better incorrectly handling alt texts on export

But what do **you** think? Tell me, I'll take notes! Or put your thoughts in the chat and we'll save it.

I kind of hate PB, for a specific technical reason I can explain if anyone is interested ... but many people like it, so I can put that negative feeling aside.

You have to love it's learning curve!

Slide 24/ 25

Contact info

Contact info:

Email: jonathan@poritz.net

Fediverse: [@poritzj@mastodon.social](https://mastodon.social/@poritzj)

Get these slides at

poritz.net/j/share/OENTTGoDot2P.pdf

and all files for remixing at

poritz.net/j/share/OENTTGoDot2P/

Or else just scan



to go to my homepage, where you can follow links to all of my shared materials.