Open Tools for Google Docs to Pressbooks Migration ... and Related Thoughts About What Makes for Good Platforms for OER

Jonathan A. Poritz

jonathan@poritz.net
poritz.net/jonathan

15 January 2025
Creative Commons Open Education Lightning Talks

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: 15 Jan 2025 10:52CET
These slides, also in editable form, are available at poritz.net/jonathan/share/OT4GD2PB/ .

Making and sharing OER: Which platforms?

A fairly common workflow in our community seems to be:

Create/adapt OER in Google Docs (hereafter **GD**)

and

share them with students in Pressbooks (hereafter **PB**)

For today, I'm going to call this ***The GD⇝PB Workflow***.

E.g., two organizations with which I've worked have done this.

- Open Oregon Educational Resources:

  (Amy Hofer, Veronica Vold, Phoebe Daurio, Jennifer Gardner, Petra Bitterli, and *many* others)

- Washington State Board of Community and Technical Colleges:

  ("**Open ProfTech**", Boyoung Chae, Monique Belair, Ashley Montenegro Ramirez; in collaboration (?) with Open Washington)

Why these platforms?

Pros of GD	**Pros of PB**
• familiarity: *e.g.*, MSWord-like • UI: WYSIWYG • supports collaboration well • supports a11y, after user effort • Google well-integrated in our lives	• familiarity: *looks* like an e-book • reading is clean, simple, responsive • allows embedded videos, H5P, *etc.* • supports a11y, after user effort • discoverability: the PB *Directory*

Cons of GD	**Cons of PB**
• easy to mess up a11y • extreme surveillance capitalism • WYSIWYG: no automation possible • terrible (non-existent?) math • interoperates poorly	• easy to mess up a11y • heavyweight solution to a lightweight problem • "text" editor and custom CSS, but limited flexibility • MathJax, not yet fully implemented • interoperates poorly

Most important pros and cons, in general

Pros of GD	Pros of PB
• familiarity: *e.g.*, MSWord-like • UI: **WYSIWYG** • **supports collaboration well** • supports a11y, after user effort • Google well-integrated in our lives	• familiarity: ***looks* like an e-book** • reading is clean, simple, **responsive** • allows embedded videos, H5P, *etc.* • supports a11y, after user effort • **discoverability: the PB *Directory***

Cons of GD	Cons of PB
• easy to mess up a11y • **extreme surveillance capitalism** • WYSIWYG: no automation possible • terrible (non-existant?) math • interoperates poorly	• easy to mess up a11y • **heavyweight solution to a lightweight problem** • "text" editor and custom CSS, but limited flexibility • MathJax, not yet fully implemented • interoperates poorly

Most important pros and cons, for this talk

Pros of GD	**Pros of PB**
• familiarity: *e.g.*, MSWord-like • UI: WYSIWYG • supports collaboration well • supports a11y, after user effort • Google well-integrated in our lives	• familiarity: *looks* like an e-book • reading is clean, simple, responsive • allows embedded videos, H5P, *etc.* • supports a11y, after user effort • discoverability: the PB *Directory*

Cons of GD	**Cons of PB**
• easy to mess up a11y • extreme surveillance capitalism • WYSIWYG: no automation possible • terrible (non-existant?) math • **interoperates poorly**	• easy to mess up a11y • heavyweight solution to a lightweight problem • "text" editor and custom CSS, but limited flexibility • MathJax, not yet fully implemented • **interoperates poorly**

There might be a problem

If teams across the open ed world are going to be following ***The GD⇝PB Workflow***, many folks will need to do this:

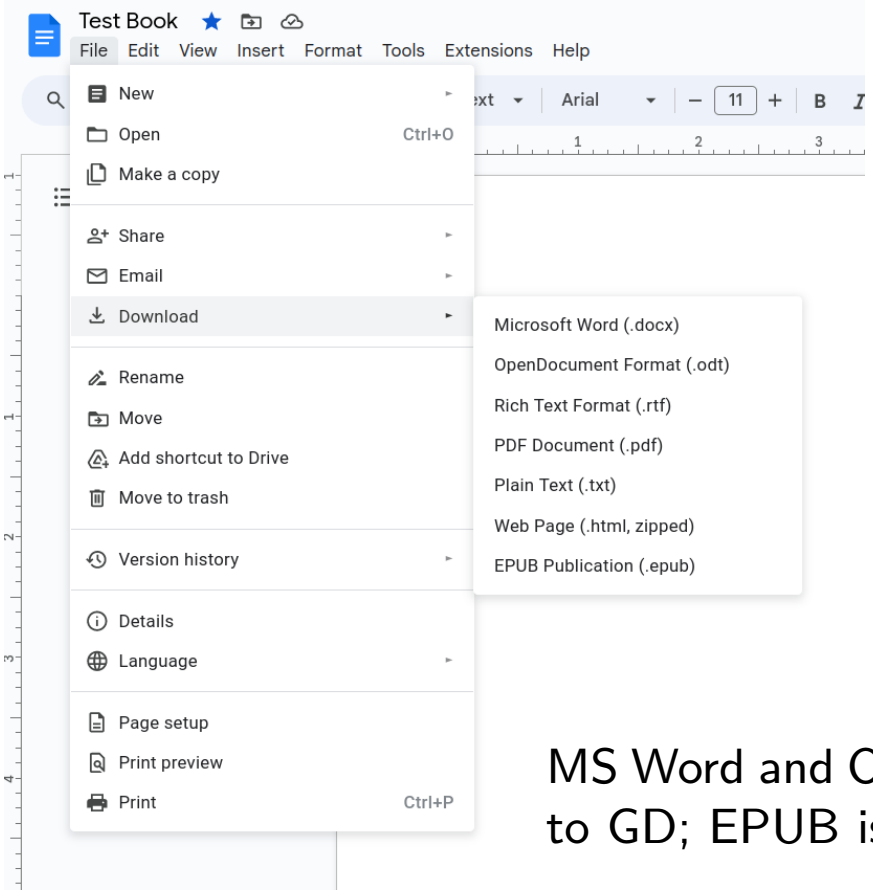
> Starting with a textbook prepared in GD, get it into PB as quickly, painlessly, and accurately as possible.

- "**Quickly**" and "**painlessly**" because many people will do this many times. In fact, arguably, because there will be all this back and forth, one wants also to be able to go back PB⇝GD....
- "**Accurately**" is doing a lot of work here, since we want not to lose the text, images, a11y features (alt texts!) ... and also we want something(s) in the GD to trigger special behavior in PB that make PB a platform we want to use for OER, such as the PB glossary feature, embedded videos and H5P, *etc.*

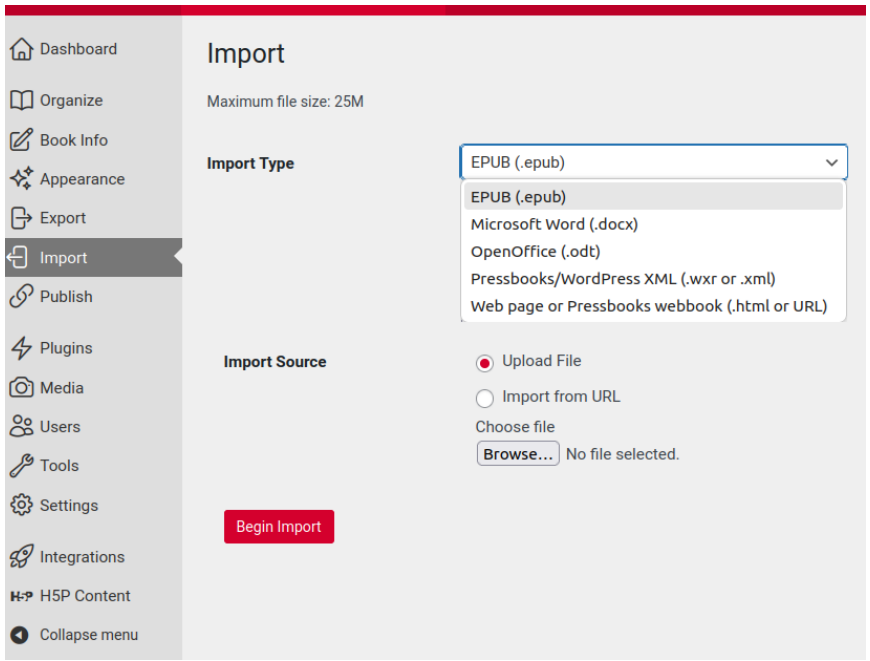
Unfortunately, platforms don't like to allow convenient entrance, exit, and interoperability: they have little incentive to allow it, and much to make interop hard. That's certainly the case with GD ... maybe not with PB?

Come on, surely this must be nearly automatic

GD export ("download"):



PB import:



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

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

Unfortunately GD⇝Word⇝PB fails in several ways

Experimentation with this process found that the alt texts of images were being lost along the way – that's a deal-breaker!!

Additionally, it was very unclear how to get elicit from the Word version of a GD book the features, like embedded videos, H5P, *etc.*, to appear in PB.

In some sense, the problem here is that the WYSIWYG UI for GD implements a point-and-click approach to the features it knows about, but it has no flexibility to deal with features other systems like PB might allow.

A way to allow that might be to add text decorations to the GD which could be translated to features in PB, such as wrapping a video URL in indications like "[embed] ... [/embed]" when it is desired that the video at that URL be embedded in the PB, something like "[GL] ... [/GL]" to mark text that will be associated with a glossary term in the PB, *etc.*

Adding enough of these decorations makes the GD start to look like it was written in a **markup language**, such as HTML or TEX, which is something many OER support teams are reluctant to impose on their authors.

A better approach: Get the OER out of GD as HTML

The way which seems to offer the most possibility for automation, not loosing important a11y information (alt texts), and leaving it in a form that enables straightforward manipulating to be able to take advantage of all of PB's affordances 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=OPeqXG-QxW3ZD8B
tmPikfA);.lst-kix_qpcrjg12wdx5-3>li:before{content:"" counter(lst-ctn-kix_qpcrjg12wdx5-3,d
ecimal) ". "}.lst-kix_5iabqo6dki7m-5>li:before{content:"\0025a0   "}ul.lst-kix_qv847ouxndz
f-2{list-style-type:none}ul.lst-kix_qv847ouxndzf-1{list-style-type:none}ul.lst-kix_qv847ou
xndzf-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_yzmht1etofw0-0>l
i{counter-increment:lst-ctn-kix_yzmht1etofw0-0}.lst-kix_h0qa59x4hax3-1>li:before{content:"
\0025cb   "}.lst-kix_e9xxj2arum8r-7>li{counter-increment:lst-ctn-kix_e9xxj2arum8r-7}.lst-k
ix_88guu1ohcsd4-1>li:before{content:"\0025cb   "}ol.lst-kix_2g5fjno92eyf-0.start{counter-r
eset: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_6omu1jmxbka6-1>li:before{content:"\0025cb   "}.lst-kix_78u85j5w8j7h-0>li:befor
e{content:"\0025cf   "}.lst-kix_78u85j5w8j7h-8>li:before{content:"\0025cf   "}.lst-kix_h0q
a59x4hax3-5>li:before{content:"\0025a0   "}.lst-kix_wtp5ial5q6ui-0>li{counter-increment:ls
t-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_5iabqo6dki7m-1>li:before{content:"\0025c
b   "}ul.lst-kix_qv847ouxndzf-3{list-style-type:none}.lst-kix_jfrs6xlc6t05-6>li:before{con
tent:"\0025cf   "}.lst-kix_bxpnqgj9tbtn-3>li:before{content:"\0025cf   "}.lst-kix_bxpnqgj9
tbtn-7>li:before{content:"\0025cb   "}ul.lst-kix_qv847ouxndzf-8{list-style-type:none}ul.ls
```

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 t
hese 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 disorde
rs.</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 dis
orders, 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>
```

At least we *can* deal with images

The HTML that is exported from GD at least has all of the GD alt texts in the generated HTML that looks like

```
<img src="image_filename.jpg" alt="Alt text goes here">
```

So we can upload the images to PB, and just copy HTML files into PB chapters via the "text" editor in PB (apparently, WordPress and PB think of HTML as "text"), and the alt texts will be there!

Some details:

1. I upload the HTML files using a lot of Python code I written that employs the "Selenium" library which enables programs to manipulate web sites as if they were people pointing, clicking, typing, *etc.* I did this because I was lazy, but arguably the work I've done for OO and the WA SBCTC would have been impossible to do by hand: many of their OER were 500-1000 pages long, and I would have been pointing and clicking until the heat death of the universe if I tried to do everything by hand....
2. For images, I import one Word file with all the images (or just a few files – max size of individual file imports is 25MB) by hand, no Selenium.

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

Once we have the HTML version of the GD book on our own machine (*i.e.*, neither Google nor Pressbook's servers!) – with an eye to uploading the HTML to PB with Selenium – we can easily fix almost anything with short snippets of Python.

For example, here is a bit of code to check that figures are numbered correctly in a GD:

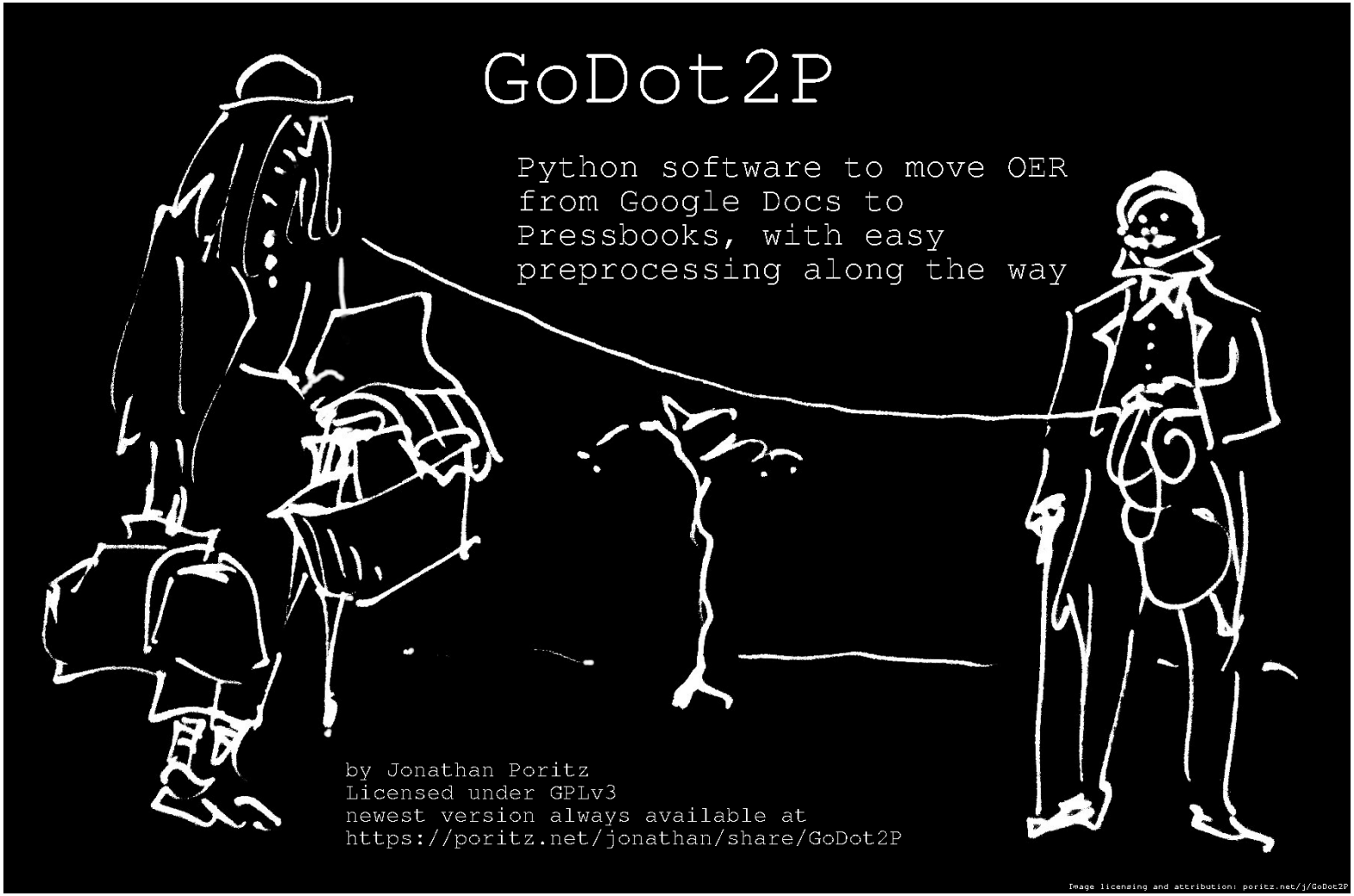
```
with open("book.html","r") as fh:
  old_lines = fh.readlines()
import re
headpat = re.compile(r'<h[1-6] id="oo([^.]+).*">')
figppat = re.compile(r'<p>Figure ([1-9][0-9]*\.[0-9]+)\..*</p>\n')
ln = -1
figures = 0
changed_labels = 0
chapstr = ""
while True:
  ln += 1
  if ln == len(old_lines):
    break
  line = old_lines[ln]
  hm = headpat.match(line)
  if hm:
    new_chapstr = hm.group(1)
    if chapstr != new_chapstr:
      chapstr = new_chapstr
      chap_fignum = 0
  m = figppat.match(line)
  if not m:
    continue
  oldlabel = m.group(1)
  if not "." in oldlabel:
    raise ValueError(f"old label {oldlabel} without a '.'!!")
  figures += 1
  chap_fignum += 1
  newlabel = f"{chapstr}.{str(chap_fignum)}"
  if newlabel == oldlabel:
    print(f"{newlabel} ok!")
  else:
    print(f"will need to change {oldlabel}->{newlabel} on line {ln+1}")
    changed_labels += 1
print(f"in {figures} got {changed_labels} incorrectly numbered!")
```

Hint: they never are! (But they can be renumbered by another code snippet.)

How to make ***The GD⇝PB Workflow*** work, in detail

1. In GD:
   - A Use all a11y features, with care
   - B Use a mild markup language to indicate where PB features should be employed after the transfer.
   - C File >Download >Web Page (.html, zipped)
2. Use the W3C's `tidy` (and a lot of editing by hand) to clean up all the HTML.
3. More cleaning, many other modifications to the HTML to allow for use of all PB features *e.g.*, by interpreting the custom ML, all in Python on local machine.
4. Upload things like all images, a database of glossary terms and definitions, by hand to PB.
5. Upload all HTML for PB "chapters" (which may be authors' chapters, or just authors' chapter sections, depending upon in house style guide) to PB with Selenium
6. Enjoy a celebratory beverage of your choice

Ta-da: GoDot2P



[1]

[1] 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.

Conclusion: in full

Platforms for adopting/adapting/creating OER must provide good support for:

- collaboration
- a11y features
- responsive display
- discoverability
- automation
- math and interactive features such as video, H5P
- **interoperability**

and must avoid:

- surveillance capitalism
- platform lock-in
- heavyweight solutions when the problem is lightweight

which probably involves using

- plain text files, wherever possible
- markup (or -down) to describe semantics and advanced features

In fact, we probably should be thinking about **protocols, not platforms**.

Conclusion: the tl;dr

Interop (and FLOSS) fundamental: in today's world, who in good conscience would sign up for a service, no matter how excellent, which is or could become a walled garden? Think of the enshittification waiting for such choices!

Don't be afraid of markup languages. **All** writing is a form of ML (arguably), all of mathematical notation certainly is – and despite the damage done to the world for literally millennia by people like me, math and mathematical symbols should not be scary or mysterious.

Why we are doing all of this work to make OER which look like commercial textbooks?

In my own classrooms, using OER I've written, I've come to appreciate ***The Beauty of Ugly OER***, or at least their pedagogical effectiveness. Why not stay in GD[2]?

Maybe, to misquote Audrey Lorde,

The master's textbooks will never dismantle the master's house.

Certainly, if we didn't pay for book designers to make fancy organization-specific custom PB CSS and for people like me to do the ***The GD⇝PB Workflow***, we would be thinking about *sustainability* in OER differently[3].

[2] At least if you don't mind surveillance capitalism.

[3] Even without replacing knowledgeable OER creators with plagiarism-as-a-service bots, as some have suggested we do.

Contact Info

Email: jonathan@poritz.net

Fediverse: @poritzj@mastodon.social

These slides:

poritz.net/j/share/OT4GD2PB.pdf

and all files for remix:

poritz.net/j/share/OT4GD2PB/

Or else use $\longrightarrow$

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

Shameless promotion: Look for new podcast

The Revolution Will Not Be Televised: The Theory and Practice of Resistance

whose first episode drops on Inauguration Day, 2025.