Ebooks: A new look and an update

I'm still surprised that things haven't changed for epub. Via Twitter I saw David Bergsland's article that, in turn, referenced Teresa Elsey's ePUB Secrets post

I haven't done serious work on epub in a while but the the playing field hasn't changed much since I did my last book about 4 years ago. Some of the issues remain the same and some issues are new and originate, in my opinion, from the merger of the IDPF and W3C (or haven't been addressed since that time).

Unless otherwise noted, all quotes are from Teresa Elsey's ePUB Secrets post.

The Article

> "Many of our familiar problems are prevalent, especially in less-mature international markets; for example, low-quality ebooks, many PDF-only, many EPUB2 files, publishers not understanding how to modify InDesign-produced EPUB files.

This is what caught my attention first and, I think, is the source of most other problems listed. How can we improve quality and provide stadardized knowledge for people working on epub and InDesign workflows?

I think part of the problem is that the epub specs (unlike the specs from W3C) are not driven by consensus. Vendors can choose to implement some specs but not others (which would explain, to me, why FXL hasn't seen wider adoption or seen a full working implementation be successful).

The other part is education. While there are many professionals in the #eprdctn community there is no standard curriculum that we can point to as a requirement for people entering ebook publishing. A few years ago there was a vendor push to create an #eprdctn certification. I remember I opposed the idea at the time because I doubted the need for such a credential back then and wondered if it wasn't an exercise in futility. Now I wonder if we're not in a position where the community generates requirements for a certification and we get serious about adopting it.

It’s also worth looking at Laura Brady’s Storify of an #eprdct discussion about professional development. May we get the roots of training and certification from it?

Why are we stuck in epub2?

> The majority of EPUBs submitted by US publishers are still EPUB2. Microsoft said 90% of their US catalog is EPUB2, with 63% of titles created in 2017 still EPUB2. Existing EPUB files still have lots of problems, like faulty or nonexistent TOCs, images used for text content, poor accessibility, and bad/missing metadata.

And why wouldn’t this be the case? the 2 largest book sellers, Amazon and Apple have thumbed their noses at the epub specs and have enchanced them (in Apple’s case) and completely thumbed their noses at it (like Amazon’s Kindle formats) knowing that there’s little the industry can do as they can’t really tell either distributor that they will not work with them and that they will take their business elsewhere.

If you buy a Kindle device you’re locked into that ecosystem with all the associated warts on the system. We’ve known for a while that Amazon can remove, and has removed books from users’ devices. Because their dominance of the market for MOBI and AZW3 books they can do whatever they want and can make changes without really caring about publishers and their end users.

Same thing with iBooks. They will read epub3 books but to get all the really revolutionary tools and techniques (many of which are actual epub3 ideas) to work on iBooks you have to use their tools or spend a lot of time and effort.

There is also the issue of compatibility. If you’ve see the work Jiminy Panoz has done for the Readium project, particularly Readium CSS you can see that even for readers that (suppossedly) support the same specification you have to do an insane amount of workarounds to have consistent reading experience across devices.

PDF as the predominant format

> There are still many places where PDF is a prevalent ebook format. EPUB is not a strong brand for users. For example, many readers may choose PDFs simply because they know and recognize the format. And of course there are many documents besides books – journals, magazines, news, documentation, textbooks – that are still most commonly available only in PDF.

When O'Reilly Media stopped selling books on their own website they stated that "After our announcement, the bulk of your requests have been for PDFs versus kindle or EPUB format". I would love to learn why is this.

At first I thought that it was search but most epub readers have search features. Then it dawned on me that it must be ubiquity. Everyonee has a PDF reaader. It is built into Firefox and Chrome for all platforms and Edge in Windows 10, for Mac it's part of the Preview App and there are many applications that you can download for desktop and mobile systems that will let you read a PDF file… The same can't be said of epub.

So we need to ask ourselves what's the value proposition for epub. Why should I pick epub3 over PDF or epub2 if I'm not certain the features, or the book itself, will work on my device or on my target devices?

> One interesting tidbit: Japanese ebooks are almost all EPUB3 because EPUB2 never had sufficient language support for Japanese, and 72% are FXL (manga!). So while we know the FXL user experience needs to be better, Japanese ebook producers are hyper aware of this fact and are driving innovation in FXL performance, speed, amount of device storage, etc.

This is what the value proposition was to the Japanese market: they can better work with their typographical requirements. Is there an equivalent for western markets?

Workflows and Tools

> Lisa McCloy-Kelley (Ebook VP at Penguin Random House)

> mentioned two future challenges she foresees as print, web, and digital publishing become integrated:
>
> - Fonts (different rights for different environments)
> - Image quality/optimization (some digital contexts now require better-than-print quality).

As a front-end web developer I’ve always found items like these frightening and amusing at the same time. I will address image quality first and the font issues in a separate section.

As long as we have an original with a high enough resolution we can create as many versions of the image, taking into consideration both resolution and Retina / High DPI versions.

Defining Retina and Retina Displays

When I say “Retina display,” I am describing device screens with a high enough pixel density that everything looks sharp to the human eye. When I say “Retina image,” I’m describing an image that has been made specifically to look sharp on a Retina display. These are two different things.

The output area of one pixel on a typical Retina display screen actually fits four bitmap pixels. This is why Retina screens are considered pixel dense. On a Retina display, each pixel in a bitmap image is stretched into an area of four pixels (2 x 2) — it’s the reason bitmap images that look sharp on lower pixel density screens and look “fuzzy” on higher pixel density screens.

Newer devices like the iPad Pro use a 3x retina display where the pixels of a bitmap image are stretched in to an area of 9 pixels. Making the job even harder.

If you want a bitmap image that is 100px by 100px to look sharp on a Retina display, you’ll need to provide extra pixels for the Retina display to work with. This is done by making the bitmap image twice as big to begin with. (In this case we’d make the initial image 200px by 200px.) The next step involves using some programming to take the big image and scale it down to the desired 100px by 100px size.

Since we don’t have images at exactly double the resolution we want we have to create them based on the sizes and densities that we need.

How to create retina images

Assuming we have a version of the image that is large enough to be a source, we can make images with enough pixels to make them look good on Retina displays.

For the purpose of this post and the technique I'm using, I'm making the following assumptions:

- The images are big enough to make into retina images
- Images are set at fixed sizes
    - small: 200px
    - large: 480px
    - extra large: 1280px
- To create high dpi (retina) images we make the pictures twice as big
    - We're only creating images for 2x retina displays
- Quality is set to 80 of the original image
- Use progressive (interlace) scan for JPEG and PNG images
- Skip enalrgement warnings when we try to make smaller images larger
- Strip all metadata from images

The example below uses Gulp and the gulp-responsive plugin to create a series of images at different pixel densities for each JPG and PNG images placed in a given directory, except the files under the touch directory.

```
gulp.task('processImages', () => {
  return gulp.src(['src/images/**/*.{jpg,png}', '!src/images/touch/*.png'
    .pipe($.responsive({
      '*': [{
        // image-small.png is 200 pixels wide
        width: 200,
        rename: {
          suffix: '-small',
          extname: '.png'
        }
      }, {
        // image-small@2x.png is 400 pixels wide
        width: 200 * 2,
        rename: {
          suffix: '-small@2x',
          extname: '.png'
        }
      }, {
        // image-large.png is 480 pixels wide
        width: 480,
        rename: {
          suffix: '-large',
          extname: '.png'
        }
      }, {
        // image-large@2x.png is 960 pixels wide
        width: 480 * 2,
        rename: {
          suffix: '-large@2x',
          extname: '.png'
        }
      }, {
        // image-extralarge.png is 1280 pixels wide
        width: 1280,
        rename: {
          suffix: '-extralarge',
          extname: '.png'
        }
      }, {
        // image-extralarge@2x.png is 2560 pixels wide
        width: 1280 * 2,
        rename: {
          suffix: '-extralarge@2x',
          extname: '.png'
        }
      }, {
        // image-small.png is 200 pixels wide
        width: 200,
        rename: {
          suffix: '-small',
          extname: '.png'
        }
      }, {
        // Global configuration for all images
        // The output quality for JPEG, WebP and TIFF output formats
        quality: 80,
        // Use progressive (interlace) scan for JPEG and PNG output
        progressive: true,
        // Skip enalrgement warnings
        skipOnEnlargement: false,
        // Strip all metadata
        withMetadata: true
      }]
    })
    .pipe(gulp.dest('dist/images'));
});
```

If command line or task runners are not your thing there are ways to script Photoshop to automate tasks like running the same commands on multiple files ussing Javascript or platfomr specific scripting languages (VBScript or Applescript).

Here is a tutorial for creating actions in Photoshop to automate tasks like modifying images.

Font Licensing, Open Source and Multiple Channels

Another thing that has always caught my attention is font selection and usage. Foundries have this insane idea that people creating ebooks can afford the fees that they charge for use in a single format, much less for multiple delivery methods.

Any use of web fonts requires at least 4 weights for the font used to prevent synthetic bold and italic font faces:

- Regular
- Italics
- Bold
- Bold-Italics

I normally add 1 additional font and weight, a monospace font at regular weight for source code and monospace. So that’s 2 fonts and a total of 5 weights.

For the purpose of this discussion we'll also assume that we want to purchase the primary font for a website with 250,000 pageviews and a single ebook title. I'm also asuming that I'm allowed to use the web font during development of the site and ebook.

One final requirement. The font publisher must make the font available in OpenType/TrueType and WOFF/WOFF2 at a minimum. Any additional formats are available upon user's request. There may be cases when we need to support older browsers and font formats that foundries don't normally provide.

With all the requirements in the open let's look at ITC Stone® Humanist, one of my favorite fonts, and how much it would cost to run in the configuration we specified.

The font's MyFonts Page lists some of the options for purchasing a license to the font. Here's the breakdown:

- **Individual weights are $87.50. The 4 fonts would total $350.00**
- **A 6 font package that includes the fonts we need is $472.50 for the standard version or $567.50 for the pro version**

If we want to add a desktop font that we can use with our DTP software the cost goes up, but that's a secondary concern as the TTF webfont should work well for development.

I did the pricing close to Black Friday in the US. Prices may be different at other times.

Open Source Licenses Used in Fonts

If the cost of commercial fonts becomes prohibitive are open source fonts a good alternative?

I'll look at two licensing options and will provide examples for you to decide on quality.

The SIL Open Font License is part of SIL (Summer Institute of Linguistic), itself a part of Wycliffe Bible Translators. It's the most used font license and is the primary licene we'll discuss in this section.

The Apache 2.0 license is an open source license from the Apache Software Foundation. It is a general license, not font specific like the SIL license.

According to Choose A License the Apache License is a permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

FAQs and answers to questions about usage. Some of the basic questions about fonts are lised below. I'm not a lawyer and this is not legal advice. If you're working for a corporation it always pays to have your legal team check the licenses before adopting them.

- SIL Open Font License
    - Web Fonts and Reserved Font Names
    - SIL OFL Restrictions
        - Can I make and use WOFF (Web Open Font Format) versions of OFL fonts?
        - What about other web font formats such as EOT/EOTLite/CWT/etc.?
        - Using SIL Fonts in Documents and Publications?
    - OFL FAQ
- Apache 2.0

Here re some open source fonts, the license they are released under, their websites (or specimen pages in Google Fonts), a Github repository and any additional notes for the font.

Open Source, License and Dowloads

Font Name	License	Website	Source Repository	Other
Roboto	Apache 2.0	Specimen	Github	Chrome for Android and Material Design Font
Noto	SIL OFL 1.1	Get Noto	Github	For languages not covered by Roboto when using Material Desgin
Raleway	SIL OFL 1.1	Specimen	Github	

Font Name	License	Website	Source Repository	Other
Source Code Pro	SIL OFL 1.1	Typekit	Github	I use it for monospaced code samples
DejaVu	Fonts are owned and licensed by Bitstream. DejaVu changes are in public domain	Website	Github	Special License
Open Sans	Apache 2.0	Specimen	N/A	Github Download
Lato 2.0	SIL OFL 1.1	Website		Can Download from Website

I used the first two chapters of Peter Pan that I created for my ebook experiments a few years ago to creaate ebooks using open source fonts to see which one works best for what kind of project.

- Original book without web fonts (zipped source – epub)
- Books using the following fonts
    - Google Roboto (zipped source – epub)
    - Adobe Sourcecode Sans (zipped source – epub)
    - Adobe Sourcecode Pro (zipped source – epub)
    - DejaVu (zipped source – epub)
    - Gentium Plus (zipped source – epub)
    - Raleway (zipped source – epub)
    - Jura (zipped source – epub)
    - Goudy Bookletter (zipped source – epub)

So if you use an open source font, you may be avoiding this kind of scenarios with font obfusctation and encryption in the future.

> My favorite fact to take back to my team is that Kobo says they've fixed font obfuscation on their readers, so we don't have to remove font encryption on InDesign-generated books

> for them anymore (… which we usually learned about when our FXL books reflowed because of missing fonts, to disastrous results; see above)

Another way to reduce the cost of fonts is to comission them specifically for your company and project. Comissioning a font would allow you to specify the looks, structure and desired language-support and OpenType features available to the font; as well as the desired level of support for print, ebooks and web content.

The interesting thing to consider is that, if we want to push the web as a publication medium, we shouldn't have many problems with the OpenType and TrueType fonts we already use (WOFF and WOFF2 are packaging systems for traditional font formats).

Web Publications

W3C Web Publications, like Web Publications for the Open Web Platform: Vision And Technical Challenges , seeks to combine the best of books with the best of the web. When I first read about PWPs they appeared to be an implementation of Progressive Web Applications. Reading the Web Publications editor’s draft has prompted more questions for me than it has answered, particularly about the perceived place of PWP in the open web and the web ecosystem.

I understand that this early specification is describing the structure of the publications and that there may be non HTML that are packaged inside the publication. Formats other than HTML would take away from using the platform as the driving vehicle for publication so what’s the point?

I will address some of the issues that PWPs raised for me in the following sections.

What’s the purpose of the PWP?

Why should we have PWPs? What is the value proposition of yet another container for our publications when we haven’t fully agreed and come up with interoperablle implementations for the existing epub3 specification?

Are we trying to create a better packaging system for our existing content?

Are we trying to leverage the features and APIs already available in the web platform? Are we trying to bend them to a vision of what a web publication should look like?

Whatever direction the format is trying to push for, I got no clear direction of where this is going. As a developer, the lack of clarity makes me wary about yet another format to create content for. I may end up just doing something similar to Jeremy Keith's Resilient Web Design or Baldur Bjarnason's This is not a book where they exercise the strengths of the web platform as static sites are converted into the publishing platforms and m content at the same time without having to change the packaging or add to the platform in order to publish the content.

How will this help with market fragmentation?

Another concern that I have is that this new packaging and content technology will be another version of epub3 and it will not fix the fundamental and systemic issues of epub as a platform.

As long as we don't give people a reason to get rid of their Kindles and other old eInk devices we can't really compete with the simple books those devices require. The devices will not read either epub or PWPs and there is no reason for them to change their minds.

There's no reason for Smashwords and other publishers to improve their service… as David Berglsand says in his article: ***I still use ePUB2, for example, because it works better with D2D [Draft 2 Digital] and Smashwords*** and why shouldn't it… they are working to the lowest common denominator that works best across devices and the lowest common denominator is epub2

Until we can show a competitive advantage for Apple and Amazon to drop propietary formats and extensions to standards we can't expect these propietary books to go away.

As long as we don't have a rationale for PWPs that tells users and publishers why they should be using this platform we will not see epub2 go away as a viable format for ebooks and digital publications.

Likewise, if we can't convince users that it's worth the investment to uppgrade to devices that support the newer technologies, the old epub2 and Kindle reaaaders will not go away… This is trickier because we don't want users to loose the investment they've madde in their books but, at the same time, we want them to access newer technologies.

Are these PWAs

If the PWPs are based on the open web platform I have yet to see actual integration points for the elements of the open web I think it fails to answer some, to me, fundamental questions about the purpose and organization of PWPs.

Some of the specific questions:

- Are we serving the content through HTTPS
- Are we installing a service worker?

I think it all boils down to this: are we treating PWPs as Progressive Web Applications that we enhance as needed for our books and publications? or are we coming up with yet another scheme to package content

Service Worker and Offline Access

The first thing that caught my attention is that it doesn’t require a service worker. Whatever content we put in the publication we need a service worker to act as the basis for additional features like offline acccess.

The servie worker provides a core set of functionality that are crucial for any successful publication. Any current book allows for the content of the book to be stored in the device to allow offline reading so why should PWPs be any different?

This aspect of the web platform should be included as part of the initial definition of PWPs.

Package, Storage And Network Requirements

Where do we store the content of the publications? Do the readers have to be always online to access the publications? How do we package the content for delivery?

Using a service worker we get one or more caches where we can store the content without depending on a network connection. If we’re online we can check if the content has been updated and put the new content in the caches if needed but unless we have large video files we don’t need to be online to interact with the content.

A related question is how wwill we package the content to distribute in a PWP and what will the requirements be. WPUB issue 94 illustrates some of the use cases for accessing WPUB.

I'm copying portions of the initial post for the issue to illustrate some othe isses that I see with the packaging and some additional issues with storage and content ownership.

> The working group has considered several possible answers to this question. They mostly fall into one of four scenarios:
>
> - A WPub URL resolves to a JSON manifest file
> - A WPub URL resolves to an HTML file that contains both a table of contents (TOC) and other metadata
> - A WPub URL resolves to an HTML file containing a TOC and which links to a JSON manifest file
> - A WPub URL resolves to a binary package (e.g. a ZIP file or SQLite database file)

Out of the four options the third one is the one that makes most sense.

The second option presents an HTML document with a table of content and metadata links. I'm guessing using `meta` tags and possible extensions to the `link` element.

Using a JSON manifest file sounds like a good idea but JSON is not a web viewable format so it would mmean that you jump directly from the JSON file to the book content. What happens if we're offline or we loose connectivity when we want to read the content?

Content should not be a binary representation of the data or tied into a single vendor implementation. This would disqualify option four as single vendor implementation is what killed the WebSQL specification.

The number and types of client are also intriguing and, I think, mutually exclusive and with so many mutually exclusive requirements.

If we stick to the web platform we shouldn't need WPub exclusive tools… If the WPub is part of the web platform then the platform tools should work with it.

> Each of these scenarios are considered in turn as they would

> be viewed by four types of clients:
>
> - An existing search engine bot
> - A WPub aware search engine bot
> - A reader using an existing user agent
> - A reader using a WPub aware user agent

The scenario that seems to be the most wanted proposal listed below. It includes the four types of agents discussed above. I think that if PWPs become popular search engines may include them in the regular search results like Bing does for PWAs and Google does for AMP content. Expecting inclusion just because it's a W3C spec is a disaster waiting to happen.

The need for supporting new and old user agents, to me, is more problematic. We either have to build both an epub version that will work with older implementations and a full web version that will conform to the WPub spec.

> **Scenario 3: HTML TOC & JSON Manifest**
>
> If a WPub URL resolves to an HTML file containing a TOC and which links to a JSON manifest file, the four clients may be expected to act like this:
>
> - An existing search engine bot (path 3A in the image below) will encounter and index the HTML page without modification.
> - A WPub aware search engine bot (path 3B in the image below) will be able to determine that the HTML page represents a WPub, and follow the provided link to the metadata if it wishes.
> - A reader using an existing user agent (path 3C in the image below) will see a TOC containing links to the components of the WPub.
> - A reader using a WPub aware user agent (path 3D in the image below) will be able to see and operate upon the WPub as intended.
>
> Scenario 3 seems to cleanly handle old and new clients in appropriate ways. Old clients could follow their noses to the components of a WPub, and new clients could easily load the JSON object to efficiently access metadata.

> Figure 1: Scenario 3: HTML TOC & JSON Manifest

The experience that we want to have will also dictate how we package the content and how do we distribute it. I don't agree that we need to have separate WPub aware search engines and user agents. If this is to work with the web platform then existing tools and infrastructure should be enough for WPub to work.

Encrypting Content And Content Ownership

Because publishers believe that DRM will protect their books and keep piracy down, we need to put this issue front and center when addressing digital publishing, packaging, distribution and ownertship. Most, if not all, books are encrypted with a few publishers offering their publications DRM free, some watermarked and some not. How would this translate to the open web?

One of my biggest fears since the Encrypted Media Extensions Specification was accepted as a recommendation by the W3C is that video will not be the only medium where encryption is going to be an issue.

Ebooks have multiple encryption schemas tied to vendors and distribution channels and any platform for velivery of this content should take this into account and either make it an issue for the DRM vendor to address or provide hooks to the different DRM systems. I would rather not have DRM at all but I've come to accept that it's impossible to have meaningful content without some sort of protection.

The Readium Foundation has created server and client reference implementations for the foundation's Light Content Protection DRM sytem. The gatekeeper for this system is the EDRLab; you have to get what I'm assuming is a X.509 End-entity certificate from the Readium Foundation to certify your LCP solution so compliant Reading Systems can accept your license and open the book for reading.

Readium LCP is not a standard nor it's exppected to become one. According to the Readium LCP FAQ:

> Readium LCP is not expected to be standardized by a formal

> standards group but rather remain an “industrial” specification for use by and among implementors who agree to participate in the interoperable Readium LCP ecosystem.

The system is not fully interoperable. You have to agree to participiate in the Readium ecosystem, license the encryption system and implement the client that works with the LCP server. If you choose not to do any or all of these things you cannot use LCP to “protect” your content.

This touches on encryption and also ownership. The later of the two is the more worrisome one to me. LCP is another way to make ebooks more expensive because of the additional cost of the DRM server certificate or certificates if you have to work on more than one system. It also prevents owners from using their books in any device they own, even those that are not licensed.

If we encrypt the content then who owns it?

This has always been one of the sticking points about ebooks for me. I spend money on purchasing books and someone else can revoke the license I purchased those books under, then do I really own them?

Unless the publisher or whoever handles the licensing is generous you can’t really share your books across devices. You would have to get multiple licenses for the versions we want and they are valid for a single user, I can’t loan out my ebooks. I also have to use the same reading application in all my devices if I want to read my ebooks in them and remain compliant with the license I was given.

Despite the anti circumvention provisions of the Digital Millenium Copyright Act and Article 11 of the WIPO Copyright Treaty there are ways to break ebook encryption schemes whether allowed to or not. People want to share and use their content without having to worry about the devices they read the content in, up to an including publishing executives.

Are publishers like O’Reilly (before they stopped selling books directly from their website), Packt, Tor and others who have released their books without DRM right in doing so?

I believe so, and here is why:

People are forced to choose the walled garden they play in because, with few exceptions in technical and fiction field, they have no alternatives because they’ve already been locked into one or more reading ecosystems.

In 2011, Charlie Stross wrote Cutting their own throats where he makes for a very compelling case to get rid of DRM altogether. He starts the article by stating that the biggest challenge to publishers is not piracy but Amazon (and more recently Apple) which provide for exclusive walled gardens that prevent readers from moving to a device or application of their choice, undercutting the value proposition of epub and WPub as a competitive platform.

Yes, Apple was sued and lost a case on ebook price fixing but Amazon has remained clear of any doubt or investigation even after the public fight it had with Hachette and the way it used its almost-absolute conttrol over the publishing market for paper books as well as digital through their channels, using the same tactic they used with McMillan a few years before.

DRM is a big part of what gives Amazon and Apple such power over both publishers and readers. If a reader has built a collection on Amazon for a Kindle she will be reluctant to get another ebook reader that will not be able to read the DRM of the MOBI or AZW3 book and, likewise, will not want to buy books from other sellers in formats that will not be readable in your shinny amazon device.

Amazon provides applications for other devices to read Kindle books but they are a way to keep the people reeled in and while it's possible to install epub readers in a Kindle Fire most of the applications have their own DRM restrictions. One of the comments on The Ebook Reader article indicates that most epub reader apps have been deleted from the Kindle Fire App Store with the Exception of OverDrive, an application used primarily by libraries as a way to lend ebooks like they do printed books and other physical media.

I believe that DRM causes more problems than it solves and the biggest loosers in that conversation are the readers.

Alternative Formats

Using web technologies like CSS Paged Media and CSS Generated Content for Paged Media allows us to generate content that is ready to use as input for Prince XML, Vivliostyle and Antenna House to generate PDF.

For a full example of how this technology, Vivlio style created a Japanese version of Lea Verou's CSS Secrets book, using the same techniques that O'Reilly used to create the English version.

A more basic example is the stylesheet I use to generate PDF versions of my blog posts.

So we have 3 possible formats: HTML for Web, PDF and a packaged HTML that can be served through WPub or epub. How much of this proocessing can we do using our current front end toolsets and processes.

Most authors consider Google Play Version of technical books (at least those from O'Reilly Media) not acceptable because it's a conversion from epub to PDF which is substandard.

Conclusion

SO where are we left?

I'm still hopeful that web publishing will finally leverage the tools and APIs that the web platform gives us to create fully interactive reading experiences that will work with regular user agents (web browsers) and ebook readers that have built in web browsers. I don't like the idea of WPub being everything to everyone, particularly when considering the hoops we already have to go through when creating ebooks.

I'm also hopefull that the draw of the web platform will be enough to draw people away from DRM. I'm also realist enough to know that we'll have a second round of debates over digital media encryption as they relate to ebooks.

Above all, I am hopeful that we'll see more creative publishing works using the web as a publishing medium.