WP/PWP/EPUB4 Implementation Plans and Testing

22 June 2017

Ric Wright
Readium Foundation

- Implementation
- Testing
- Exit Criteria
- Testing Champions

Implementation

Implementation - Reading Systems

- Readium has implemented almost all of 3.0.1 and pieces of 3.1
- Fully intend to support EPUB 4, PWP and WP – whatever they end up being ☺
- Readium has already taken the initiative with Readium-2 to explore what might be beyond an EPUB reading system
- *Note:* R2 is NOT a replacement for R1
- R2 is instead an evolution from R1, with a more platform-specific and modular design
- It's not complete, but great progress is being made

Implementation - Reading Systems

- Readium-2 is being developed for server deployment , native mobile apps and the desktop (where it is wrapped in Electron)
- It is to be expected that other major reading systems will also provide implementations, e.g.
  - Vital Source
  - Google
  - Library for All
  - and many others...
- It will be interesting to see how other Reading Systems approach the architecture and implementation

Implementation - Authoring

- This has proven to be EPUB's Achilles heel
- Authoring EPUB 2 is fairly easy – there are many tools out there – some of them good
- EPUB 3 is *much* harder
- There are few tools out there that export real EPUB 3 (not just EPUB 2 with EPUB 3 tags)
- *None* of them support all of it
- Little or no support for JavaScript or Media Overlays
- WP and EPUB 4 might be worse
- How can that be addressed?

Testing

Test Parameters

- Major dichotomy
  - Are we testing *content* itself?
  - Or are we testing the *rendering* of content?
  - Or both?
  - What is the exit criteria for each?
- Who executes the tests?
  - Independent organization?
  - Self-interested parties? (i.e. implementers)
- Dissemination of the results
  - Public or private?
  - Website, database, web-service?

Testing Reading Systems - *Goals*

- What are the goals of our testing?
  - Adherence to the specification
  - Completeness of the implementation
  - Interoperability between implementations
  - Quality (detecting bugs)
  - Performance
    - Memory
    - Speed
    - Reliability
  - Accessibility
  - Failure conditions
  - Stress conditions
  - Regression testing

Testing Reading Systems - *Methods*

- EPUB TestSuite
  - Useful but doesn't cover complex real-world problems
  - More for testing completeness and providing a yardstick for exit criteria
- Automated Tests
  - Difficult to create, hard to maintain
  - Requires dedicated manpower
  - Can't test some aspects (e.g. accessibility)
- Unit Tests and Continuous Builds
  - Very important to Reading Systems
  - Very RS-specific

Testing - Reading Systems – *Test Suite*

- EPUB TestSuite Shortcomings
  - Doesn't cover all of 3.0.1 and little (none?) of 3.1
  - Many tests are of the underlying browser engine
  - Tests of JS, SVG, WebGL or interactivity are very limited
  - No performance testing
  - No stress testing (size, complexity)
  - No failure cases (e.g. malformed or illegal markup)
  - Limited and hard to assess tests of FXL
  - Limited and hard to assess tests of CJK, Arabic, Hebrew
  - No tests of other scripts and languages
  - No localization (all English)
  - And, of course, new specs from EPUB 4/WP

Testing Authoring - *Goals*

- Adherence to the specification
  - Passes EPUBCheck (or similar)
- Fidelity
  - Of the content (author/editor intent)
  - Across different devices
- Quality (detecting errors)
- Performance
  - Memory
  - Speed
- Accessibility
- Best practices check

Testing Authoring - *Methods*

- EPUBCheck or similar
  - Simple sanity check, useful but only a start
- Automated Tests
  - Difficult in practice?
- Manual Testing
  - Labor-intensive but necessary
  - Large number of devices and usages complicates it
- Workflows
  - Publishing workflows vary widely across publishers
  - Testing tends to be very specific to the workflow
- Best Practices Tool

Testing Authoring – *Best Practices*

- This not specifically a spec item, but it is clear need
- Need is for a tool that flags
  - Use of illegal or deprecated markup
  - Inefficient markup
  - Inefficient content (e.g. oversize images)
  - Incomplete or incorrect metadata
  - Including font files with insufficient rights
  - Poorly structured EPUBs (too many or too few spine items, etc.)
  - Etc.
- Ideally, such a tool could be used either interactively or in batch mode

Exit Criteria

Exit Criteria – Reading Systems

- Typically, the W3C requires:
  - 2 independent, interoperable implementations of each feature
  - Detailed test-plan
  - Accessibility, Privacy, i18n and Security assessments and plans
- Given the complex nature of the use-cases and implementations which using microservice architectures, it could be challenging to decide what a “complete feature” is and is it complete
- Simple cases, yes, but all those edge cases...
- This emphasizes the need for a clear set of test cases
- What constitutes “done”*:*
  - Performance? If a feature is dog-slow, is it complete?
  - Bugs? How bad does a bug have to be to reject a feature?

Exit Criteria - Authoring

- Typically, the W3C requires 2 implementations that ***consume*** the content
- Should that be broadened to ***authoring*** the content?
- IDPF didn't have such a requirement and it has proved (IMO) a problem, *cf.* lack of authoring tools
- OTOH, EPUB4/PWP/WP may be such a complex beast and the use-cases equally complex that defining such criteria may be a challenge
- We can cover the simple cases, but what about all those edge cases?
- But probably worth thinking about - at least as a thought-experiment
- Again, a best-practices guide would be excellent

Testing ~~Champion~~ Ambassador

Testing Ambassador – Reading Systems

- Given the potential complexity of the spec and the fact that at least one of the implementers (Readium) is open-source (read: herding cats in the dark), it will be very challenging
- OTOH, even in these early days, entities such as EDRLab and Feedbooks are dedicating significant resources to the effort
- Validating the *interoperability,* given the complexity of the spec and the use/workflows, will be challenging
- The role of the ambassador will probably be a combination of coach, cheerleader and critic

Testing Ambassador - Authoring

- Similarly, validating the *interoperability* given the potential complexity of the spec and the use/workflows will be challenging
- There will also be a perceptual issue in that as authoring of EPUB 3 is still difficult and poorly supported, convincing publishers to support a new, more complex spec will be *hard*
- It may be almost impossible unless we *mandate* authoring support
- But that is a chicken and egg situation
- Without RS support, how can publishers commit?
- Without authoring support, how can RS commit?

Readium.org

22 June 2017

Appendix

Readium-2

- Highly modular
- Microservice architecture
- Based on the Web Pub Manifest specification
- RESTful
- Leverages platform strengths
- Designed to be accessible
- Both client and server implementations
- Supports both applications and headless usage