EDUPUB Summit
Phoenix 26-27 Feb 2015

Meet the Editors

Markus Gylling, (IDPF & DAISY), Matt Garrish (Invited Expert), Bill Kasdorf (Apex), David Stroup (Pearson), Darryl Lehmann (Imagineeringart), Garth Conboy (Google), Brady Duga (Google)
EDUPUB Profile

**Package Level Metadata**
- Must specify as “edupub”
- Must identify one or more accessibility feature
- Must specify if teacher-edition or teacher-guide
- Must specify source when print derivative or teacher edition
- Other educational metadata is optional (i.e. "Should")
  - e.g. audience, educational role, alignment, activity type...
  - May be specified by RDFa/Schema.org or other
Renditions

- Focus is on reflowable accessible content
- Fixed layout best practice is to use semantics and not image
- Must also provide an accessible rendition if image based fixed rendition for accessibility
Content model

- Focused on reflowable content (for initial draft)
- Enables Accessibility:
  - on multiple platforms/form factors
  - defines reading order
  - define navigation/hierarchy
    - Must include TOC links to the full document heading hierarchy
    - heading ranking (i.e. use of h1-h6)
    - nested sections/asides
    - page-list nav required if page-markers exist
  - Teacher content must be identified with aria-label
- Guidance on image & media spec recommendations
EDUPUB Profile

**Semantics**

- EPUB 3 core semantics enhanced/extended to support educational content
  - e.g. learning objectives, outcomes and standards and assessment
- Does not currently specify semantics specific for a teachers edition
  - aria-label used to identify teacher content
- Focus on providing examples and best practices on implementation
- Sample CSS styling controlled by epub:type and/or class attribute values
Annotations

- EPUB Adaption of W3C Open Annotation
  (http://www.openannotation.org/spec/core/)
  - Requires XHTML5 for annotation body
  - Requires JSON-LD serialization
  - Uses EPUB CFI for anchoring to target
  - Define specificity levels (release, publication and work).
  - Syntactic restrictions to allow non-RDF-aware processing
  - Allows bundling of collections of annotations (+ zip for transport)
  - … and specifying target audience (teacher, age range, etc)

- Life: bundled with EPUB, or as separate entities
- Required features in EDUPUB-compliant RS: import, export
Distributable Objects

http://www.idpf.org/epub/do/

- https://drive.google.com/file/d/0B_r69cPgzjHjeXpibl9pdFBYU2s/view?usp=sharing
EPUB Scriptable Components (aka Widgets)

- Status
  - Scripted components team
    - Led by Will Manis, ex of B&N College, now living the life of leisure!
    - Participants from Nook, Google, Pearson, Inkling, Imagineering, Metrodigi, plus invited experts
  - Worked over the last year to develop two specifications
  - Draft specifications
    - Packaging of components
      - http://www.idpf.org/epub/sc/pkg/
    - Component communication protocol
      - http://www.idpf.org/epub/sc/api/
EPUB Scriptable Components

● ESC Packaging Details
  ● ECS’s are packaged as EPUBs
    ○ No new XML vocabularies are required.
    ○ The epubcheck validation tool can be used to ensure validity of the Scriptable Components, and that all necessary resources are present.
    ○ The Package Document can be used to store Scriptable Component metadata.
    ○ Fixed-layout metadata can be used to communicate the desired aspect ratio.
    ○ Standalone debugging of Scriptable Components is possible, as the Scriptable Components can be ingested into any Reading System with scripting support (a component is simply a “page” with spine-level scripting).
EPUB Scriptable Components

● ESC Packaging Details
  ● A Scriptable Component must:
    ○ Be a Packaged Object from DistributableObjects, except:
      ■ Its Base Document must be a valid XHTML Content Document (no fragments)
      ■ Its resources must be structured as spec-ed
      ■ It must have required metadata (epubsc:version)
      ■ Single Base Document in the spine
      ■ Have a dc:type of “scriptable-component”
  ● An embedded Scriptable Component must:
    ○ Incorporate Scriptable Component resources as spec-ed
    ○ It must include a scriptable-component <collection>
EPUB Scriptable Components

● ESC Packaging Details
  ● Embedding an ESC
    ○ Migrate Resources as is DistributableObjects, except:
      ■ The Base Document must be referenced from at least one <iframe> element
      ■ Any obfuscated fonts, the fonts must be de-/re-obfuscated
  ● When an ESC is embedded
    ○ The creation of an Embedded Component from DistributableObjects is required.
    ○ An Embedded Object is created per DistributableObjects Collection, except:
      ■ The Embedded Component <collection> must have the role attribute identifier "scriptable-component"
      ■ The collection must not include the following metadata from the Packaged Component:
        ● Primary dc:identifier elements
        ● The last modified date (dcterms:modified)
Embedded ESC Example

<collection role="scriptable-component">
  <metadata>
    <dc:type>scriptable-component</dc:type>
    <dc:title>Gallery</dc:title>
    <dc:creator>DynamicInc</dc:creator>
    <dc:language>en</dc:language>
    <meta property="epubsc:version">1.0</meta>
    ...
  </metadata>
  <collection role="manifest">
    <link href="../components/DynamicInc/Gallery/gallery.html"/>
    <link href="../components/DynamicInc/Gallery/css/common.sample.css"/>
    <link href="../components/DynamicInc/Gallery/css/gallery.css"/>
    ...
    <link href="../components/DynamicInc/shared/js/external/captionator-min.js"/>
    ...
  </collection>
  <link href="../components/DynamicInc/Gallery/gallery.html"/>
</collection>
EPUB Scriptable Components

- Components API
  - Purpose: provide interoperable mechanism for widget communication and interaction
  - Augments, does not replace EPUB 3.0.1 scripting
  - Designed to work in RS that support spine level scripting with no effort
  - Not addressed: Security and Privacy
EPUB Scriptable Components

- Architecture
  - All components must be in an <iframe>
  - Must assume it is in a separate domain (that is, no direct scripting access across the iframe boundary)
  - All communication done with postMessage
  - Messaging implementation included in the spine
    - Enables drop-in support if spine level scripting is supported
EPUB Scriptable Components

- **Init**
  - Initialize postMessage mechanism
  - Optional custom steps
  - post ready message

- **Messages**
  - Well defined structure
  - Identifies that it is an EPUB SC message
  - Other useful metadata (id, timestamp, etc)
  - Various reserved topics (ready, pause, resume, etc)
  - Components communicate with each other using the epubsc_publish method with a custom topic and topic data
  - Message go from child to parent, and parent to child when the child is subscribed
EPUB Scriptable Components

- Events
  - Components report events (mouse/touch, keyboard, etc) to their parent and whether they were handled
  - Uses the postMessage mechanism
  - Intended for Reading System UI (for instance, page turning)
  - Events only go from child to parent
- Code
  - The working group will provide code that can be used to implement the messaging API, both for widgets and the spine
  - The code is not normative, the spec is
  - Code is on github: https://github.com/IDPF/widgets
EPUB Scriptable Components

● Going Forward
  ● Scriptable component samples
    ○ Prototype code for packaging and protocol lives on github
      ■ Packaging - https://github.com/IDPF/componentUtility
  ● Code will be brought into alignment with the specs and productized/hardened
    ○ The team will be working on this over the next few months
EPUB Scriptable Components

- Community Asks
  - Expect the specifications to mature based on feedback and experiences with sample code
  - The team encourages active review of the specifications
  - File comments/bugs using the issue tracker
    - [https://code.google.com/p/epub-revision/issues/list](https://code.google.com/p/epub-revision/issues/list)
  - Please explore the sample code
  - Please integrate specifications into your work products
Q&A