Content Strategy:
DITA vs Markdown

Content strategy definition

- A content strategy defines the guiding philosophy for your organization’s technical content.
- Components of content strategy
  - Goals
  - Objectives
  - Source formats and architecture
  - Delivery formats and architecture
  - Design

Reminder: business requirements

- Most of us work for businesses
- Goal is to make money (profits)
- Two ways to make profits
  - Generate revenue
  - Cut costs

Technical content

- Content supports the product
  - User manuals, online help, IFUs
  - Financial services guidance
- Content IS the product
  - Medical reference content
  - Insurance content
  -

Characteristics of technical content

- User focused
- Maintained
- Usually includes extensive re-use
- Often translated
- Multiple delivery channels
- Managed

User requirements

- Levels of professional and product skill
  - Meet needs of novice/inexperienced, intermediate, and expert users
- Searchability/findability/navigability
- Accessibility

Content maintenance

- New versions/variants/models of products
- Medical reference content
  - New knowledge, and new conditions
  - Updated recommendations for best practice
  - Regulatory and insurance changes
- Financial and insurance content
  - Legal and regulatory changes
  - New financial instruments/policies

Regulated environments

- Tracking and approval of changes
- Need to track product requirement or defect driving update
- Regulatory authorities require review in context
  - Code-level diff not sufficient for these authorities
- Reviewers often lack technical savvy
  - Intelligent, accomplished in their own domains
  - Software tools not in their skillset

Re-use

- Benefits
  - Efficiency
  - Consistency
- Levels of re-use
  - map/topic (topic/section)
  - Granular
- Need re-use native to the design of the source format
  -
  -

Translation

- Track language/locale
  - Native metadata
- Support for translation tools

Delivery formats and channels

- Print/PDF
- Online/HTML
- epub
- Content delivery platforms
- Apps
- In the future: automatically-generated videos, virtual reality

Goal of lightweight formats

Simplify writing

Markdown

- Markdown “is intended for one purpose: to be used as a format for ***writing*** for the ***web***.”
- Markdown “syntax is very small, corresponding only to a very small subset of HTML tags.”
- “For any markup that is not covered by Markdown’s syntax, you simply use HTML itself.”

Asciidoc

Then why, oh why, do we make [writing] more difficult by burying the content in XML schemas like DocBook, allowing complex word processors to distract us or wasting time battling with finicky WYSIWYG editors?

Imagine if writing documentation was as *simple as writing an email*.

It can be.

That’s the idea behind lightweight markup languages such as AsciiDoc. They offer a plain-text syntax, embellished with subtle, yet intuitive markup, that’s designed for humans to read, write and edit in raw form.

ReStructured Text (ReST)

The primary goal of reStructuredText is to define and implement a markup syntax...that is readable and simple, yet powerful enough for non-trivial use.

(reStructured Text) is useful for in-line program documentation (such as Python docstrings), for ***quickly*** creating ***simple web pages***, and for standalone documents.

Creators of lightweight markup

- Markdown
  - John Gruber (software developer)
  - Aaron Swartz (software developer/blogger)
- Asciidoc
  - Stewart Rackham (software developer)
- ReST
  - Python development team (software developers)

Goal of lightweight markup

- Simplify a task that is ancillary to the creator’s main work.
- “One and done” content (blog), not maintainable content.

What gets lost

- Focusing on writing places emphasis on the ***writer***, not on the ***user***.
- Focusing on writing ignores the requirement to ***manage*** technical content.
- Focusing on writing ignores other requirements of technical content.
- Lightweight markup achieves it goals by ignoring major requirements of technical content.

Complexity

- The task of creating, managing, publishing, and delivering technical content is complex.
- Complex requirement drive formats that support those complex needs.
- XML/DITA are not complex simply for the sake of being complex.
  - The requirements they are designed to fulfill are complex, hence the complexity of the formats.

Whither lightweight markup

- Lightweight markup is here to stay in software.
  - Software developers in particular like it.
    - (Psst...they also hate XML!)
- Future in software is hybrid environments.
  - Lightweight markup for content contribution, especially when SW developers write content.
  - Integrate lightweight markup files into robust, XML-based content environments.

- Questions?
- Responses?