WILEY

# The Inadvertent Accessible Content Architect

Tzviya Siegman

Books in Browsers V

October 2014

## All documentation starts somewhere

- Internal specifications, industry best practices, standards all begin with requirements and use cases
- Not everyone calls them by name
    - I want this book to look “great” on HappyReader, but we can ignore SadReader for this project
    - My customers like to read in night mode
    - Machines must be able to recognize that this chunk of content as a learning objective

## How do we get from requirements to specs?

- Prioritize requirements and make sure they fit into the big picture
- Define each requirement
- Test (as relevant)
- Write, write, write
- Test again

**/*This is rarely a fast process*/**

specifications

reusability

mobile

chunk

repurpose

EPUB

money journal

css

iOS

discoverability

Webkit

ebook

Kindle

budget

portability

XML touch Android

content

TIME

AJAX

HTML

marketing

accessibility

structure

## What is the first priority?

- Priorities vary: Time, budget, aesthetics, accessibility, reusability, portability…
- Many will say that aesthetics come first
  - “I don’t care what you do with the file. Just make sure it looks good.”
  - Not all content needs to look the same, so it is difficult to write broad documentation around aesthetics
  - Adjusting CSS is often easier if it is apart from the content structure

```
<h1 class="chapterTitle">
h1.chapterTitle {…}
```

  - This is separating presentation from content

**/*Accessibility points*/**

## What about time?

- Creating a spec or standard is often done under extreme time pressure
- Don’t start from scratch. See what already exists.
  - Build on existing standards, sample code, documentation
  - Don’t Repeat Yourself – has this been done already? Why are you doing it again?
  - Why include both <nav epub:type="toc"> and <li class = “toc”> for a displayed table of contents in an EPUB when they accomplish the same purpose? **/*Accessibility points*/**
- Templating, or creating standard structures, often saves time in the long run

## Defining requirements

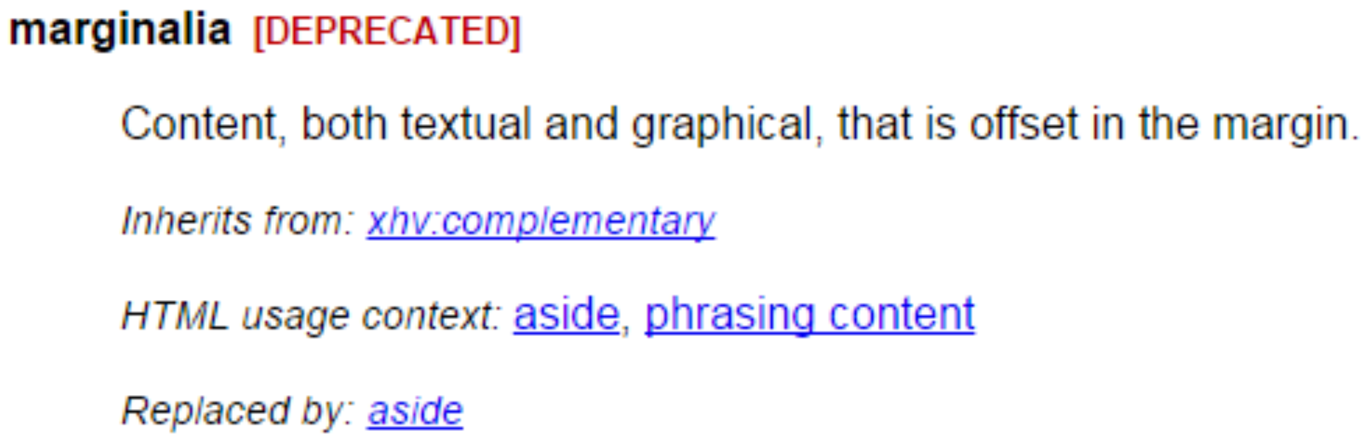
- Defining requirements is where the dirty work happens:
  - What is each element?
  - Why is it important to include?
  - Scope. How extensive is this thing?
- Defining requirements helps all stakeholders understand the needs and refines the vision of the project

## Use Case: <aside> and complementary content in EDUPUB and EPUB 3 SSV

- The EPUB 3 Structural Semantic Vocabulary included a number of terms to refine <aside> or other block-level elements: marginalia, pull-quote, note, notice, warning, footnote, sidebar…
- While assessing the needs of additional terms for EDUPUB, the working group first assessed the existing terms to determine overlap
  - New term <aside epub:type="help">. What if content that is classified as "help" is also classified as marginalia? Should it be <aside epub:type="help marginalia">?

## Resolving <aside>s

- We decided to revisit the list of terms and make use of the semantic implications of HTML **/*Accessibility points*/**
  - Several SSV terms have been deprecated now
  - Some seemingly redundant terms were maintained because they are already widely used and backward compatibility is also important



**marginalia** [DEPRECATED]

Content, both textual and graphical, that is offset in the margin.

*Inherits from:* xhv:complementary

*HTML usage context:* aside, phrasing content

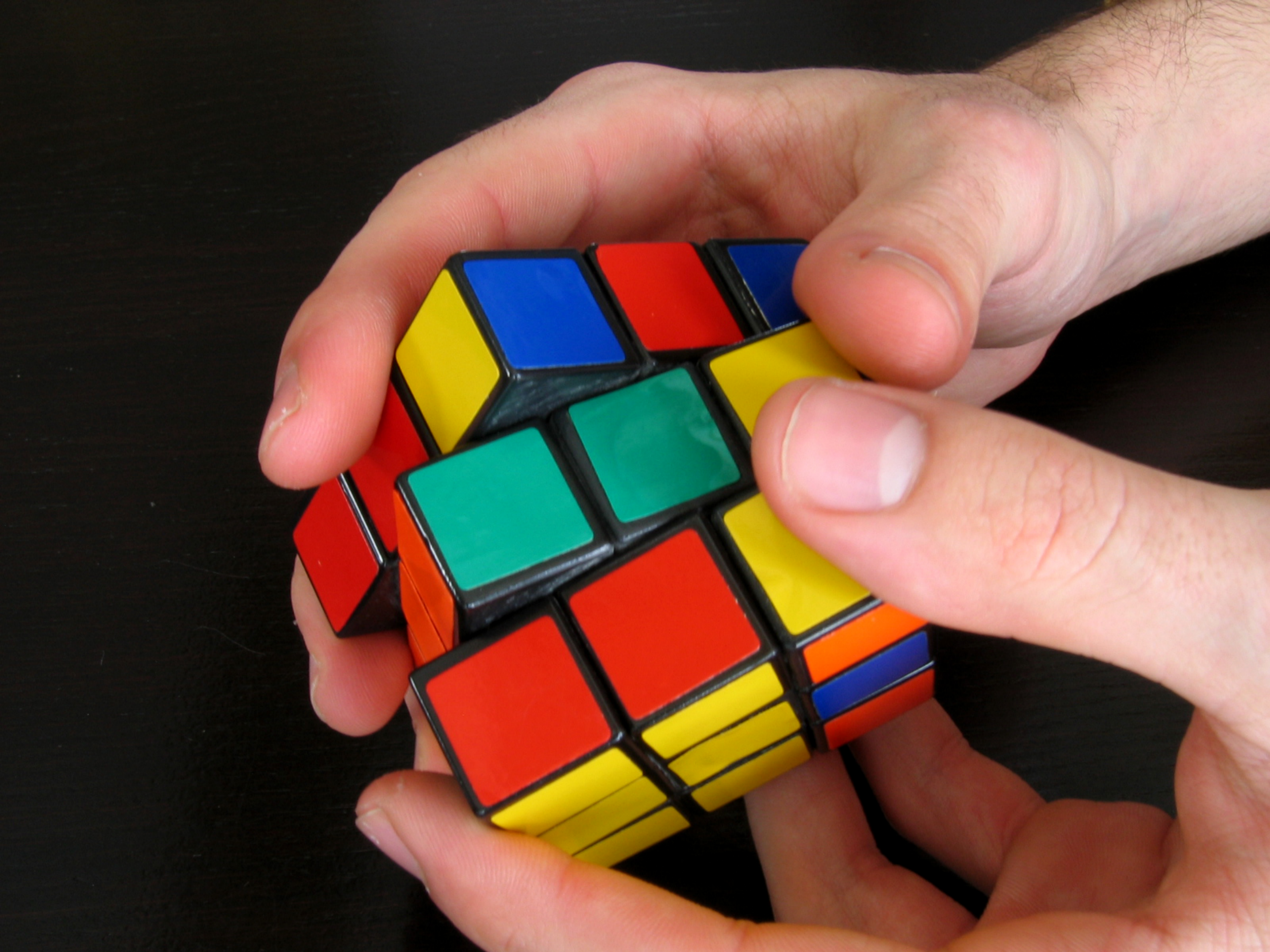
*Replaced by:* aside

## But, what if there’s a conflict?

- Conflicts arise all the time
  - Happy Reader recommends using ems or percentages for margin definition, but JoyousReader requires px
  - We want to include image descriptions, but it SO expensive
  - We would like the specification to indicate that definitions of key words display automatically alongside terms, but how does that affect reading order?

## Conflict Resolution

- Sometimes, it comes back to highest priority
  - We sell 90% of our content on HappyReader. JoyousReader will have to hop on board.
  - I don't care if HappyReader recommends doing it that way. That violates Industry Standards and makes the files non-accessible. HappyReader is going to come around eventually. Maybe our use of this standard will convince them. (I'm a big fan of this method)
  - But, a great rule of thumb is to find a happy compromise



## Break Nothing & KISS

- Don’t break content structure
  - Navigation
  - Hierarchy
  - List structures
  - Tables
  - Nesting structure
- There exist many standards, specifications, and best practices – use them as a resource
- Don’t make the specification difficult to follow
- Don’t try to accomplish too much at once

## Accessibility as Tie Breaker

- Often when assessing a conflict, or even how to prioritize needs when writing a specification or documenting a workflow, accessibility wins.
- Why?
  - Documentation already exists
  - Content structure is built in
  - More bang for the buck – budget, future-proof, no-one will sue us for being non-compliant, and BTW, accessible content can be beautiful
    - (see "Design Like You Give A Damn" http://www.youtube.com/watch?v=vK1tlLOavvM)

## Conflict: Tables

- Tables don’t have the greatest history on ebook reading systems
- Some devices can’t display more than 3 or 4 columns
- Some devices have difficulty displaying cells that contain too much text or spans and straddles
- Some reading systems recommend avoiding tables
- Non-fiction ebooks include a lot of tables
  - We have worked with our authors and editors to get away from using tables as a crutch. Sometimes lists are more appropriate
  - But, tables are an important tool that should not be lost to small screens. (This is a challenge for mobile web in general.)
  - Tables are also accessible when well-formatted

## Resolution: Tables

- If large tables are captured as images, they can be inaccessible to all readers
  - Images of text can be difficult to read for anyone
  - AT cannot interpret the text in an image
- If tables are captured in HTML, they are accessible to more readers and are future-proof for those systems that will soon improve table support

**/*Accessibility points*/**

## Example: Math

- Displaying math on the open web is tricky
- MathML is a robust markup language that provides the foundation for the inclusion of mathematical expressions in Web pages. MathML is accessible because it's machine readable.
- But, very few browsers, SDKs, or ebook readers support MathML
- Polyfills like MathJax are available for some platforms
- In other platforms, we are forced to use images ☹
- This was my first push toward a11y advocacy

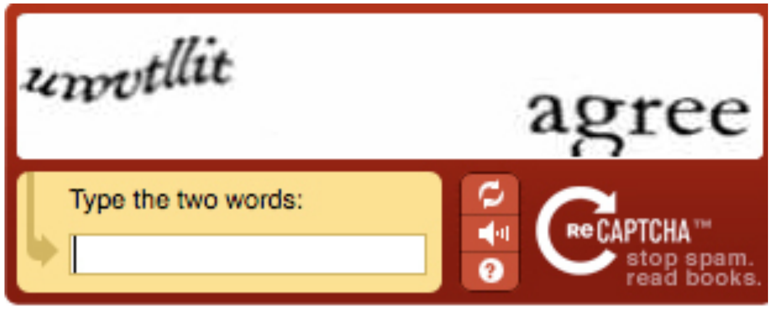
## Example: Content Reuse

- Publishers need to make sure that every piece of archived content is repurposable
- How can I tag or label each chunk in a meaningful way?
- Options: CSS class, namespacing, XML, data-*, RDFa
- All of these work to degrees
- IDPF has a vocabulary that has been extended by EDUPUB. These terms work well within the EPUB world.
  - http://www.idpf.org/epub/vocab/structure
- But what about HTML? Work in progress with accessibility in mind

**/*Accessibility points*/**

## OK, but what does a11y do for me?

- What do you do when you encounter this?



<img alt = “screen shot of Captcha”>

- I usually listen to the audio alternative after being mistaken for a robot after conflating u’s v’s and w’s
- I am a sighted customer taking advantage of a service provided for non-sighted users.

## How do I make sure my spec works?

- In the ebook space, we encounter countless reading systems, each with their own specifications or, worse, undocumented systems
- Structured content provides information for current and future machines: reading order, implied behaviors based on semantics
  - What does a machine do with <div class="list">?
  - All HTML readers (machines & humans) have a way to interpret <ul>
  - This is an example of creating meaningful structure

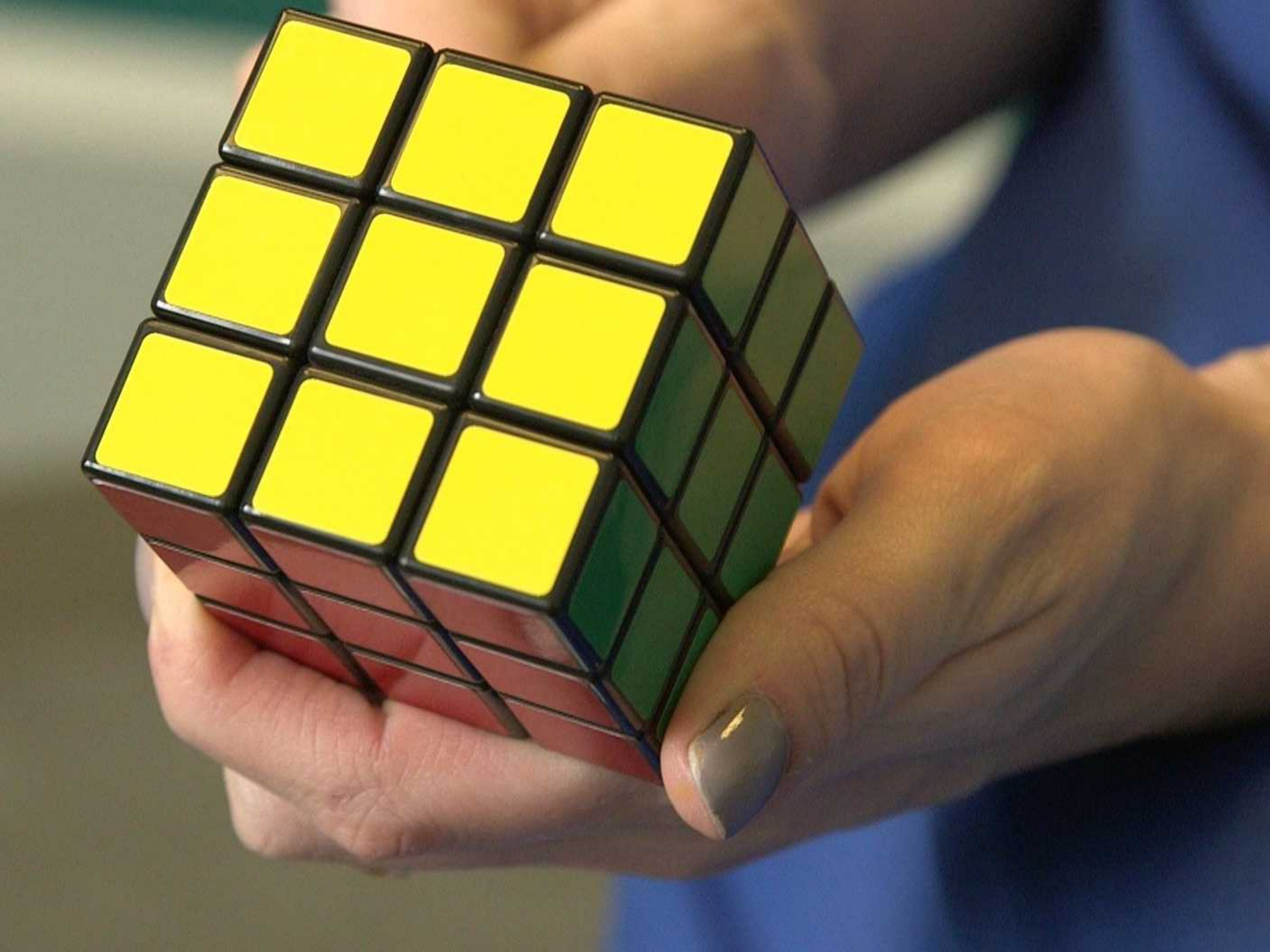
**/*Accessibility points*/**

- *Note: solid content structure does not assure that your spec will work. TEST! TEST! TEST! But, please, do not include bad code to achieve a result on one engine.*

## Conclusion

- Considering basic accessibility guidelines and best practices from the outset will help you write better documentation
- You will come back to it in the end

**/*Accessibility wins!*/**



## References & Resources

- HTML 5.1 Elements with information about ARIA usage
  http://www.w3.org/html/wg/drafts/html/master/dom.html#elements
- EPUB 3 Structural Semantic Vocabulary: http://www.idpf.org/epub/vocab/structure
- Web Content Accessibility Guidelines (WCAG) 2.0 http://www.w3.org/TR/WCAG/
- Accessible Rich Internet Applications (WAI-ARIA) 1.0 http://www.w3.org/TR/wai-aria/
- DIAGRAM Center Top Tips for Creating Accessible EPUB 3:
  http://diagramcenter.org/54-9-tips-for-creating-accessible-epub-3-files.html
- MathJax http://www.mathjax.org/
- Léonie Watson "Design Like You Give A Damn"
  http://www.youtube.com/watch?v=vK1tlLOavvM
- DIAGRAM Center http://diagramcenter.org/
- Photos from http://www.everystockphoto.com/

## Thank you

Tzviya Siegman

tsiegman@wiley.com

@TzviyaSiegman