To: Chris Lindgren

From: Haiyen Le

Subject: DITA topic model design rationale

Date: 12/23/2015

This topic model contains five documents, not including the DITA Map. There is the Introduction concept topic “c\_intro”, which briefly describes the blockquote HTML element. The concept topic also explains how there are two purposes to blockquote based on the witing elements it’s meant to emulate, as block quotation or pull quotes which should be considered when using the element (the implication is that you should use blockquote for explicitly one writing element or the other on an HTML document, not both). This is followed by three task topics. Two of the task topics “t\_quotemarks” and “t\_quoteline” instruct the user on how to implement CSS code that’s already been written and tested to alter the output of blockquote from the default. One of the task topics creates oversized gray quotation marks that frame the quote text, with enough spacing to have it stand out from the regular text. The other task topic has code that renders the quote indented from the left with a bordering line, which is a more minimalistic way to show that the text is a quotation. The third task topic describes how to use a live text editor, in this task CodePen, to preview and edit the code for blockquote instantaneously. My reference topic describes the properties commonly used to edit the output of blockquote, particularly for the two examples of code I provided.

My structure for this topic model is pretty basic unfortunately, due to time and environmental constraints (I’m in an internet-less vacation condominium as I type this up). In the beginning, I also had trouble coming up with task topics in relation to blockquote, and ones based off of my historicizing project. Blockquote is a pretty straightforward element, so I decided to have my tasks revolve around the CSS coding that affects how it appears. I also mentioned in my project that the way that blockquote is a text element that can be edited very well with CSS, and that flexibility led to the various ways it’s used online for quotes, which can disregard the official specifications for it. CSS styling would also be something that’s generally useful for the user if they wish to use it. After all, the default appearance is simple indentation of the text. In contrast to HTML coding where you simply frame the text with the proper tags, CSS coding can become very extensive for any element that’s written as the selector, and there are many ways to style blockquote into something that tells the readers that it’s quoted text. This leads to the problem of having too many things to cover with just task topics. During the research for my historicizing project, I came across pre-made CSS styles for blockquote on CSS Tricks, and then that became the basis for my tasks. The goal for my topic model became “Showing the user how to edit the default blockquote with CSS”. I did not want to involve the controversy over proper semantics usage because I worried picking a side would limit what I could do, so I went the route of choosing two different styles of blockquote that each would appeal to a side. The style with the large quotation marks is the one I see used in online news articles for pull quotes, which is the more appealing and attention-grabbing style, but less professional. Meanwhile the style with the line border is often used in more professional articles, as it indicates the usage of a blockquote clearly, but does not detract from the text or take up too much space. The “quotes” style is the one that appeals to the side that supports using blockquote outside of its specification, while the “line” As for the idea for using CodePen as part of a new task topic, it came from when I was experimenting with code samples that I found online. I found it immensely helpful to check on small snippets of code efficiently like blockquote CSS. Codepen also helped me when I was explaining my DITA topic model to my groupmates during our last class. When I got a feel for how the CSS properties affected the output, I figured that maybe a reference topic to explain some of them might help the user in general, and noted in particular the ones used the code I gave as a sample for the user to implement or edit. My topics ended up related to each other in some way, thankfully. I guess that the CodePen task topic is what ties everything together, as the two “example” tasks can use it, and the same goes for the reference topic of users want to make further edits. However, I wrote it as something done independently from those two task topics, because there are limitations to using CodePen, and its purpose in this scenario is convenience to quickly inspect the output of pieces of code. As a result, the architecture has them all as sibling topics, because they all can work independently from each other as tasks. The two things that I wish I could do are create a supertask for the two “coding example” tasks to be under as child topics, and used the conref attribute for reusability. The supertask is for the awareness that the two child topics are very similar to each other, although what stopped me from writing it is not really knowing what to put in such a supertask that doesn’t take away from those two topics, aside from an introduction to CSS and even that seems superfluous. As for the lack of a conref, I regret not coming to more office hours to understand it better, because the book and the oasis do not help me understand how to implement the conref attribute. I know that it’s used to make parts of content reusable and that does sound immensely helpful. If I knew how to implement it, I’d be reusing that table from the reference topic into the task topics. In the end though, I’m satisfied with what I could accomplish, however.