To: Chris Lindgren

From: Anna Irish

Subject: DITA topic model design rationale

Date: December 22, 2015

The purpose of this memo is to describe and rationalize my DITA topic model design. My topic model describes three uses for the grid layout module Flexbox. The three uses are: creating a navigation bar, reordering flex-items, and using the flex-grow property. Surrounding these tasks are a concept topic and reference topic. The concept topic is an introduction to the topic model explaining Flexbox and its many layout functions, two-dimensional and three-dimensional. The reference topic describes Flexbox properties mentioned in the task topics. The main goal of my topic model is to highlight Flexbox’s many uses and show the diversity of Flexbox’s properties.

The topic model’s information architecture is interconnected. Each task topic includes step-by-step coding instructions. Even though each task has a different end goal, there are overlapping Flexbox properties in each. These properties include: display, justify-content, and flex-flow. All Flexbox properties mentioned in the task topics are associated with an attribute and defined in the reference topic. This helps readers understand what each property does. The display property is the most important property in each task. This defines that you are working with Flexbox. Without this property, the other Flexbox properties would not work. The reference topic is important because it helps readers understand why the specific property is used in the task topic. It can also be helpful to readers who want to use Flexbox properties outside of the task topic suggestions. The concept topic is referenced in the prerequisite section of the task topics. It is an introduction to the topic model and an overview of Flexbox. Since Flexbox is a layout module it has many different uses and properties, it would take a much larger topic model to cover everything Flexbox includes. I imagine my tasks as subtasks in this sort of greater topic model.

All of the task topics have a prerequisite section detailing what the user needs before starting the task. They all require the user to have an operational understanding of HTML and CSS and a basic understanding of Flexbox. Each tasks requires the user to gather certain information. For the task topic: creating a responsive navigation bar, the user must know the page titles and URLs. It does not require the user to have a fully designed or operational website. For the task topic: reordering flex-items, the user must know the type of items (navigation bar elements, header, aside, footer), number of items, and desired styling of items. This user needs to gather this same information for the flex-grow task topic.

My DITA topic model went through various design decisions over the course of my writing. All the design decisions were made with my intended goal in mind: highlighting some of Flexbox’s uses and showing the wide application of those uses. The first main design decision I had to make was figuring out the scope of my topic model. Flexbox can be used for anything from creating an entire page layout to centering an item on the page. I decided to keep the scope of my project more focused on smaller uses. This prevents Flexbox’s properties from getting lost in the coding and gives the overall topic model more focus. A topic model covering an entire layout using Flexbox would result in a different goal than this topic model.

The second main design decision I made was choosing the three task topics. I wound up with: creating a responsive navigation bar, reordering flex-items, and using Flexbox’s flex-grow property. I decided to choose each of these topics for a different reason. Creating a responsive navigation bar with Flexbox was the first task I thought of. Instead of trying to explain how to code an entire page layout, I choose one specific element almost all webpages have. This task makes it easy to create a navigation bar with equally spaced items (page links). I decided to create my second task topic, reordering flex-items, because it is one of Flexbox’s unique properties. Order allows the user to rearrange how items appear in the flex container without changing the source order. With this property you can rearrange the items on your page without having to move the coding around.

The third main design decision I made was deciding the user’s levels of expertise. All of my task topics require a certain level of knowledge from the reader. I describe the reader’s necessary experience to complete the task in the task topics’ prerequisite section. To complete each task the user must have an operational understanding of HTML and CSS. Flexbox is not meant for beginner coders; it is complex and builds off of prior HTML/CSS knowledge. Although I suggest the reader is experienced with HTML and CSS, I only suggest a basic understanding of Flexbox. I chose these levels of knowledge for two reasons. The first is because at the end of each task is a step for the user to style their end result how they want. For example, the navigation bar task topic helps the user create a responsive navigation bar. The only style suggestions I include are removing the text decoration from the links and the bullet points from the list. Aside from this, it is just a plain navigation bar. I leave the reader room to design the navigation bar how they want. The second reason I suggest having a broader understanding of HTML/CSS but only a basic knowledge of Flexbox is because the scope of my project is focused on only a small portion of the Flexbox layout module.

The fourth main design decision I made was in the prerequisite section of the task topics: order and flex-grow. In this section I instructed the user to gather the following information before beginning the task: type of items, number of items, and desired styling of items. The navigation bar task topic has a specific use case in mind, where the task topics order and flex-grow do not. The order property can be used for a number of different reasons. One suggested use case is repositioning a sidebar above main content but keeping the source order intact. Giving the user the decision to choose what type of items they want to use for the task gives the tasks reusability. Some suggested items for order are navigation bar elements, header, aside, footer, etc. Basically, the user can choose to reorder anything within a defined flex-container. I choose to put in the same options for the flex-grow property with different suggested item types. For flex-grow the user may want use the task for items such as columns, buttons, table elements, or navigation bar elements. I included the navigation bar elements in both to tie back in with the first task. If you complete the first task, you may find the two other tasks helpful in styling the navigation bar. For example, the user may make the homepage element larger than the rest of the pages using flex-grow or the user may want to rearrange the order of the pages. These tasks help with the main goal of my topic model: to show the diversity of Flexbox’s properties.

I made several smaller design decisions in my task topics. One of these decisions is choosing for all of my task topics to be completed in Codepen.io. Although they could be completed in any text editor, I choose Codepen.io because it allows you to edit the HTML and CSS code in one spot and see the output. It makes the tasks simpler than including information such as CSS style sheets and coding in different files. It also helps focus the task topics on the coding of Flexbox’s properties because it takes away distractions of going back and forth between HTML and CSS documents. Assuming the user has an operational understanding of HTML and CSS, they can export their completed task to separate HTML and CSS files. Writing in Codepen.io also helps the user visualize their creations, which is important when the user arrives at the styling step of each task. Another design decision I made was figuring out what to include in the reference topic. I decided to only have one reference topic due to the overlapping properties in my task topics. Since I suggest only a basic understanding of Flexbox in the prerequisites, I decided having a table to define each property in the tasks would be beneficial. I use several of Flexbox’s properties in the task topics, but only a few of the properties’ attributes. The property and attribute combined create a unique function of Flexbox. This reference page lets the user easily figure out what each property and its associated attribute do. I think this reference page is important for reusability because these properties can be used outside of the task topics.