MIDDLESEX Community College

Tools and Technologies for Tech Writers 2023

Week 11

DITA

Notices

This document was prepared as a handout for the Middlesex Community College Tools and Technologies for Technical Writers class, Winter semester 2023.

Prepared by Zoë Lawson, course instructor.

Contents

DITA Resources

A few sites to look at about DITA.

DITA 1.3 Specification: http://docs.oasis-open.org/dita/dita/v1.3/dita-v1.3-part3-all-inclusive.html

DITA Open Toolkit: https://www.dita-ot.org/

Oxygen XML DITA Authoring: https://www.oxygenxml.com/doc/versions/25.1/ug-author/topics/author-dita.html

DITA Best Practices: https://www.amazon.com/DITA-Best-Practices-Roadmap-Architecting-ebook/dp/B005FEOU48/ref=sr_1_1?crid=2HXA23N0U4QW8&dchild=1&keywords=dita+best+practices&qid=1586399477&s=books&sprefix=DITA+Best%2Cstripbooks%2C208&sr=1-1

Boston DITA User Group (DITA BUG): http://bostondita.org

DITA XML (May not work in Firefox): http://dita.xml.org/home

DITA Users mailing list: https://dita-users.groups.io/g/main

Structured authoring resources

Structured authoring is an important concept in technical writing you should understand.

I recommend you review these articles to get a better grasp on structured authoring.

- Structured Authoring and XML

Working with maps

A ditamap is how you organize your content.

There are two major types of ditamaps.

Map style	A map style ditamap is just a table of contents. You can add a title, but not much else. You can create a hierarchy of topics. Used with non-PDF content, such as HTML output, or to reuse. Maps are also used for all the non-TOC things you can do with ditamaps, including key definitions, relationship-tables, constraints, etc., but those are advanced topics.
Bookmap style	Intended for use with books. Bookmaps contain a lot of metadata to create all the front and back matter of a book, such as titles, sub-titles, publisher information, copyright information. You can also add specific references for chapters, appendices, table of contents, index, and so on.

About the Oxygen DITA Maps Manager

This is a glorious panel in Oxygen that makes working with the topics in your map much easier.

The major function of a map file is to be your table of contents. The **DITA Maps Manager** displays your map file in a hierarchical way. You can easily add new topics, reorganize your topics, and open topics to edit them from the **DITA Maps Manager**.

Please remember that maps can do a whole lot more (relationship tables, key definitions, etc.), but for 90% of your DITA career, it's the file you use to control the structure of your topics.

Add a new topic from the Maps Manager

This is one of many ways you can add a topic to a map.

1. Select a node in your map.
2. Select where you want to add the new topic.
   - **Append child** > **New** — This is the only option available when you select the root node of the map. This adds a new topic as a child of the selected node.
   - **Insert before** > **New** — Adds a new topic before the selected node.
   - **Insert after** > **New** — Adds a new topic after the selected node.

   The **New** dialog opens.
3. In the **Filter** field, enter the topic type you want to create.
   - `concept`
   - `task`
   - `reference`

4. Select the topic type from **Framework templates** > **DITA** > **Maps**.
   To add a concept type topic, select **Framework templates** > **DITA** > **Maps** > **Concept [DITA/Topics]**.
5. Enter a **Title**.
6. Confirm the path and file name.

   Double-check your path. There is a folder for you in the `Week11-DITA/Homework` folder. Make sure you are saving your files in your folder.

   The file name is auto-generated based on the title for the topic.

   For example, the title `My first concept` becomes `my_first_concept.dita`.
7. Click **Create**.

Oxygen creates the new topic, which opens in the Editor. The new topic is also added to the map where you requested.

Maps - behind the scenes

Oxygen is lovely that it provides a graphical representation of your map, but remember, it's just an XML file.

Maps are very powerful, and there are a lot of different things you can do, but generally, table of contents maps are just a hierarchical set of topic references.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Map//EN" "map.dtd">
<map>
    <title>test</title>
    <topicref href="h1_test.dita">
        <topicref href="h2_topic.dita"/>
        <topicref href="h2_topic2.dita">
            <topicref href="h3_topic.dita"/>
            <topicref href="h3_another_topic.dita"/>
        </topicref>
    </topicref>
    <topicref href="h1_section2.dita">
        <topicref href="h2_moretopic.dita"/>
    </topicref>
</map>
```

Create a map in Oxygen

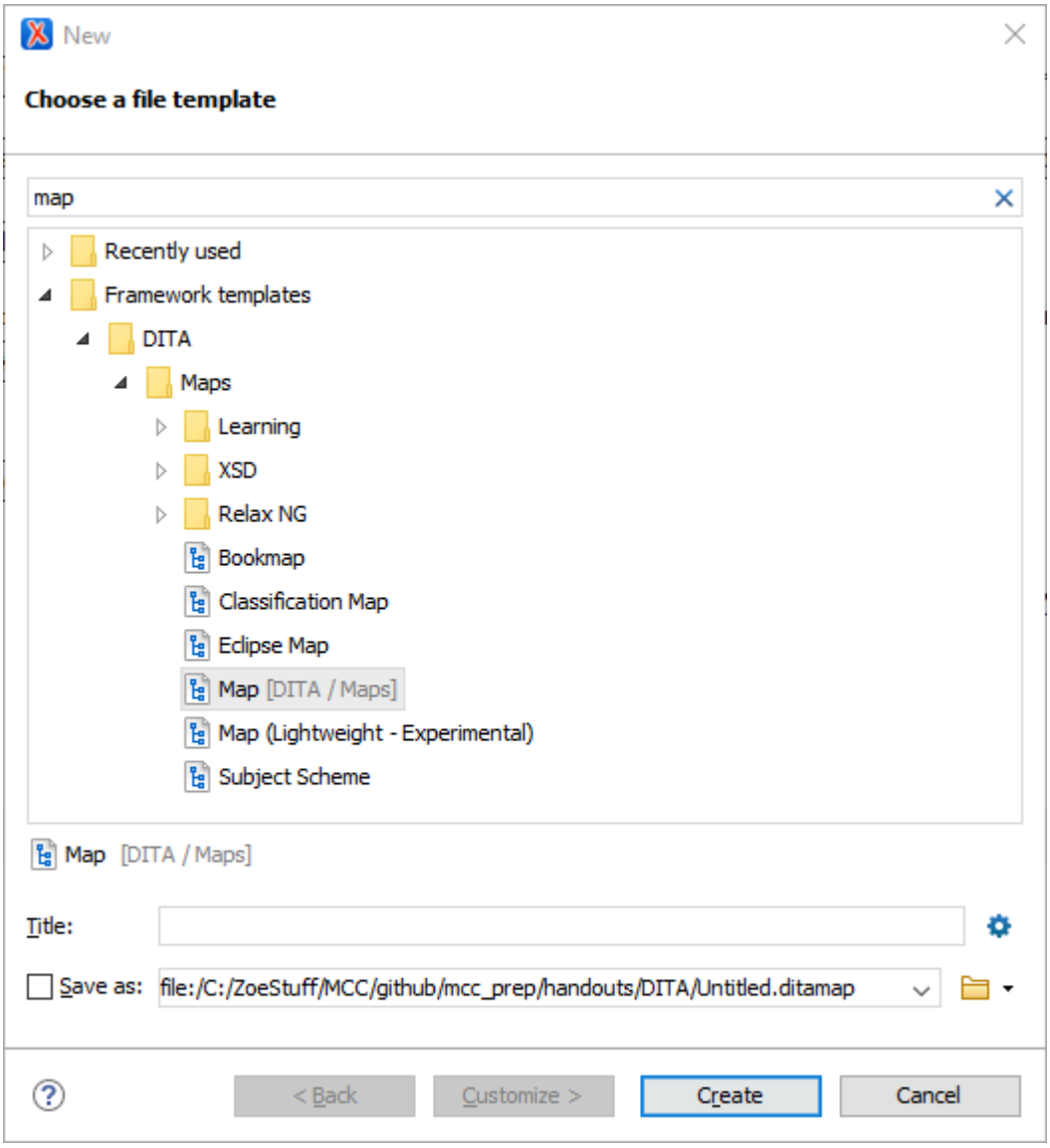
You rarely make maps, but they are how you get started.

Akin to learning how to cast on in knitting, you spend much more time working in maps than making new maps.

I've set up bookmaps for you to use, but if you want to experiment on your own, have fun.

1. Use your favorite method to make a new file in Oxygen.
   - Select **File** > **New**
   - Press `CTRL+N`

   The **New** dialog opens.



2. In the **Filter** field, enter `map`.
   This filters the list of available templates to just the map style templates.
3. Select **Framework templates** > **DITA** > **Maps** > **Map [DITA/Maps]**.
4. Enter a **Title**.
5. Confirm the path and file name.
   The file name is auto-generated based on the title for the map.
   For example, the title `My first map` becomes `my_first_map.ditamap`.

6. Click **Create**.

Oxygen creates your new map file.

Oxygen prompts whether you want to open the map in the editor or the DITA Maps Manager. I recommend you open the map in the DITA Maps Manager.

Working with topics

Topics are the heart and soul of your content.

A topic is your basic building block. A topic should be one unit of content that contains a single idea.

The theory is that you write a whole bunch of topics, and then you organize your topics into output, such as a book or help. By making lots of small chunks, you can mix and match the building blocks as needed.

Basic DITA has three main types of topics: concepts, tasks, and reference.

Concept	About a thing. Information you want people to know before they try to do a thing. This topic type is the least restrictive. It's generally a blob of text, with assorted lists, tables, and images as necessary.
Task	How to do a thing. The core of this topic type is `<steps>`. This topic is the most restrictive. There are many specialized sections to help you make consistent presentation of procedures.
Reference	Stuff about a thing you want to look up. A concept tells you the general "about", such as how a process works. A reference topic contains specific details you want to look up, such as valid values for a property. A general rule of thumb is if the information might fit into a table, it should be a reference topic. Reference topics are a bit more restrictive than concepts, but not as detailed as a task.

Concept Basics

A topic that describes an idea.

Concepts are the most generic of the specialized topics.

A concept topic must contain the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE concept PUBLIC "-//OASIS//DTD DITA Concept//EN"
 "concept.dtd">
<concept id="All_topics_must_have_an_id">
    <title>Topic title</title>
    <shortdesc>The short description of the topic</shortdesc>
    <conbody>
        <p>Your topic content goes here. Most all basic authoring
 elements are
           allowed here.</p>
    </conbody>
</concept>
```

<concept>	The root element for the concept topic.
<title>	All topics must have a title. Unless you provide <titlealts> (alternative titles for search results or navigation), the content of this element shows up wherever you link to this topic. The <title> appears in the topic as the heading, in the table of contents, and as the generated text in links to this topic.
<shortdesc>	While not required, the short description really, really should be. See About short descriptions on page 14.
<conbody>	Container for the content, or body, of the topic.

The <conbody> contains the meat of your topic. You can use most all of the basic building blocks in a concept. See Basic authoring on page 15.

Reference basics

Reference topics are designed for information you look up.

The reference topics are mildly restricted. Many people get mildly stymied by reference topics at first due to these restrictions.

The reference topic is similar to the concept topic, except there's an odd list of things you can put into the <refbody>.

A reference topic must contain the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE reference PUBLIC "-//OASIS//DTD DITA Reference//EN"
 "reference.dtd">
<reference id="All_topics_must_have_an_id">
    <title>Topic title</title>
    <shortdesc>The short description of the topic</shortdesc>
    <refbody>
        <section>
            <p>The <refbody> can contain one of the following:</p>
            <ul>
                <li>section</li>
                <li>table</li>
                <li>example</li>
                <li>refsyn</li>
                <li>properties</li>
            </ul>
        </section>
    </refbody>
</reference>
```

<reference>	The root element for the reference topic.
<title>	All topics must have a title. Unless you provide <titlealts> (alternative titles for search results or navigation), the content of this element shows up wherever you link to this topic. The <title> appears

	in the topic as the heading, in the table of contents, and as the generated text in links to this topic.
<shortdesc>	While not required, the short description really, really should be. See About short descriptions on page 14.
<refbody>	Container for the content, or body, of the topic. The <refbody> is restricted and you can only add one of the following: • <example> — A specialized section that can contain any of the basic authoring elements that's intended to be an example of something. • <properties> — A specialized table for a list of properties and description. (This option is going away and is rarely used. I would avoid using it.) • <refsyn> — A specialized section container for syntax items. Syntax generally refers to command line or programming objects. This block can contain any of the basic authoring elements. • <section> — A container for any of the basic authoring elements. • <table> — A table. If you feel you need some introductory text, you'll have to add a <section> element before the <table>.

Task basics

A task topic contains all the information on how to do a thing.

Task topics are the most specialized of the basic DITA topic types. They take a bit to get used to, but are pretty powerful.

A task topic must contain the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE task PUBLIC "-//OASIS//DTD DITA Task//EN" "task.dtd">
<task id="All_topics_must_have_an_id">
    <title>Topic title</title>
    <shortdesc>The short description of the topic</shortdesc>
    <taskbody>
        <prereq>Optional section</prereq>
        <context>Optional section</context>
        <steps>Optional section</steps>
        <result>Optional section</result>
        <tasktroubleshooting>Optional section</tasktroubleshooting>
        <example>Optional section</example>
        <postreq>Optional section</postreq>
    </taskbody>
</task>
```

<task>	The root element for the task topic.
<title>	All topics must have a title. Unless you provide <titlealts> (alternative titles for search results or navigation), the content of this element shows up wherever you link to this topic. The <title> appears

	in the topic as the heading, in the table of contents, and as the generated text in links to this topic.
<shortdesc>	While not required, the short description really, really should be. See About short descriptions on page 14.

<taskbody>	Container for the content, or body, of the topic. The <taskbody> is restricted and you can only add one of the following:
	• <prereq> — Prerequisites. This block can contain any basic authoring elements. The intention of this section is to list items you need or things you have to do before doing this procedure.
	• <context> — A brief context about the procedure. What are you doing and why. Basically, a mini-concept in your task. This block can contain nay basic authoring elements.
	• <steps> — The heart of the procedure. These are the steps to do the thing. There are actually three types of steps:
	  • <steps> — an ordered list of actions. Can only contain <step> elements. Super fancy . See The step on page 12.
	  • <steps-unordered> — an unordered list of actions. <step> elements. Super fancy See The step on page 12. I use this a lot when I have a single-step procedure, or when describing 3 ways to do a thing, such as how you can select **Edit** > **Copy**, right-click and select **Copy** from the context menu, or press **CTRL +C**.
	  • <steps-informal> — a blob of text where you can use any of the basic authoring elements.
	• <results> — A container for the results of the procedure as a whole. This block can contain any basic authoring elements.
	• <tasktroubleshooting> — A container for common issues and other troubleshooting information for the procedure as a whole. This block can contain any basic authoring elements.
	• <example> — A container for an example. This block can contain any basic authoring elements.
	• <postreq> — A container for what to do next. This block can contain any basic authoring elements.

Many processors provide default headings for each of the task sections that you can use if you wish.

The <step>

This is probably the most restrictive element out there. It can be a pain to learn, because there are a lot of restrictions. However, once you get a hang of it, it's useful.

```
<step>
    <cmd>The action, or command, to perform.</cmd>
    <info>Information about the step</info>
    <choices>
        <choice>List of options to choose from</choice>
        <choice>Option 2</choice>
```

```
        <choice>There's also a <choicetable> if you prefer a table</
choice>
    </choices>
    <stepxmp>A step example</stepxmp>
    <stepresult>The step result</stepresult>
    <steptroubleshooting>Troubleshooting info for this step</
steptroubleshooting>
</step>
<step>
    <cmd>Do step two.</cmd>
    <info>Which has additional steps.</info>
    <substeps>
        <substep><cmd>You can have only one level of substeps</
substep>
        <substep><cmd>substeps work like steps with the same options</
substep>
    </substeps>
</step>
```

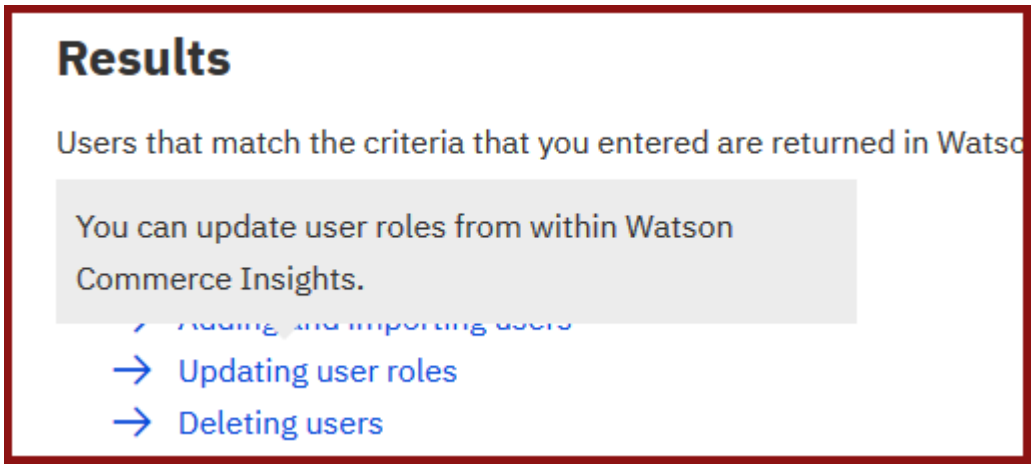
<step>	A container for an entire step. It's related to a . There are a limited number of elements you can use here, and they have to be in a specific order.
<cmd>	A required element for the <step>. This command contains the action of the step, such as “Click **OK**.” You can use most of the inline elements in this block.
<info>	A container for information about the step. You can use most of the basic authoring elements, but not new sections.
<choices>	A list of options or things you can choose from. This is a specialization of . <choices> contain <choice> elements. Generally rendered as an unordered list.
<choicetable>	A specialized table for a list of options.
<stepxmp>	A step example. If you start writing “For example, enter your username,” you probably want to be using a <stepxmp>.
<stepresult>	The result of the specific step. Your “The widget dialog box opens.” goes here.
<step troubleshooting>	Short troubleshooting you might want to add for the particular step. “If the row is in edit mode, this operation fails. Click outside of the cell to exit edit mode and select the row again.” could go into a <steptroubleshooting>.
<substeps>	Currently, DITA only allows for one set of substeps. The theory was that generally, nested substeps gets gnarly and you want to re-write anyway. A <step> can contain one set of <substeps>. <substeps> contains <substep> elements that work just like <step> elements, except you can't add additional <substeps>.

About short descriptions

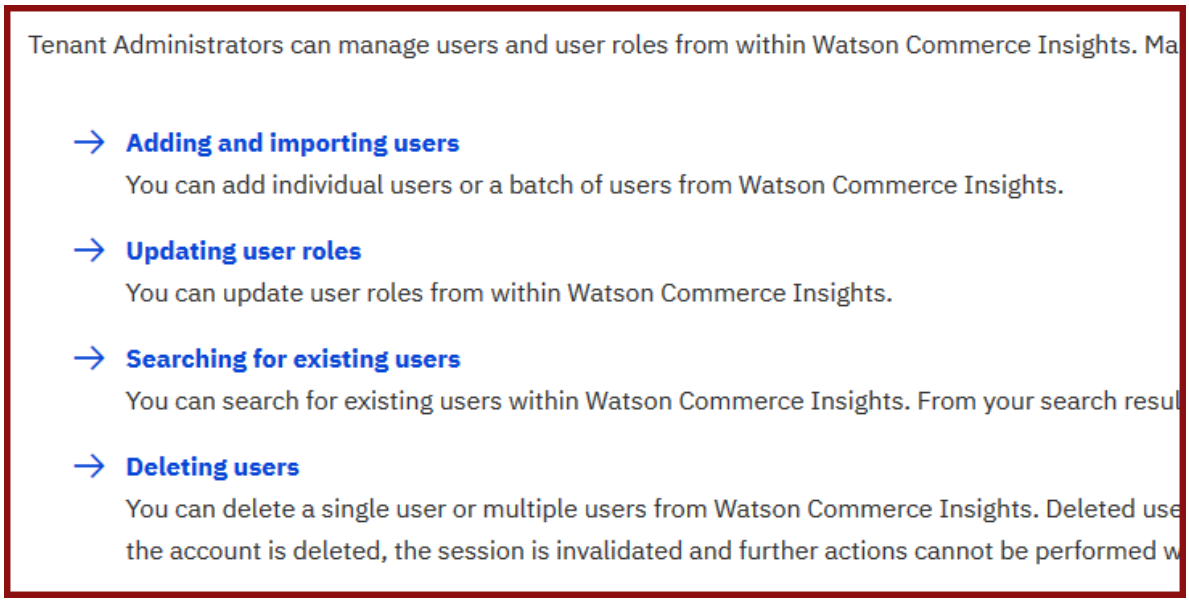
Every topic can have a short description, and should.

Use the `<shortdesc>` element to provide a short description of your topic. In general, it should be just a sentence or two, just text. In fact, some automatic checkers complain when your `<shortdesc>` is over 50 words.

It's more than just the introduction to your topic. Many DITA processors take the short description and use it with links. For example, if you transform your content into HTML, the short description content could be a popup when you hover over a link.



You can also configure DITA topics to generate various types of automatic links. These links can also pull in the short description.



Short descriptions may also be weighted heavier for search results, which can improve your SEO (search engine optimization).

Learning to write short descriptions is an artform, similar to a first paragraph in a news story. It is well worth practicing.

When writing short descriptions, you want to avoid the following:

- "This topic..."

  Don't write a circular introduction. Think of this as the lead in to an article. You want text that concisely describes the contents, and makes people want to read it (if it's what they're looking for).
- Lead-ins to lists:

  The short description is often used by tools as a brief description of this topic. If you have a lead-in, the contents are usually dropped from the link. Imagine a search result that just says "Read this to learn:"
- Images

  Not necessarily a no-no, but in the alternative ways short descriptions are used, the image may not appear.
- Links

  Again, not necessarily verboten, but if the text is showing in a pop-up, the link may not work.

Writing good short descriptions is a common topic for presentations at conferences. It might be worth putting researching (and practicing) short description authoring on your to-do list.

- https://www.oxygenxml.com/dita/styleguide/Syntax_and_Markup/c_Guidelines_for_Crafting_Short_Descriptions.html
- https://www.oasis-open.org/committees/download.php/57803/DITA-Adoption_2016_Writing-Effective-Short-Descriptions.pdf

Basic authoring

DITA uses many authoring structures similar to HTML.

For the majority of your writing, you use elements that should be familiar. The following elements work exactly the same as HTML.

- `<p>`
- `<ul>`
- `<ol>`
- `<li>`

There are several elements that are not exactly the same as HTML, but are very similar.

- `<image>` — for inserting references to images.
- `<note>` — for notes, tips, warnings, etc.
- `<xref>` — for inserting links.

Tables are also similar to HTML, but not quite the same:

```
<table>
  <title>Optional title for the table</title>
  <tgroup cols="2">
    <colspec colwidth="1*"/>
```

```
    <colspec colwidth="2*"/>
    <thead>
      <row>
        <entry>Animal</entry>
        <entry>Gestation</entry>
      </row>
    </thead>
    <tbody>
      <row>
        <entry>Elephant (African and Asian)</entry>
        <entry>19-22 months</entry>
      </row>
      <row>
        <entry>Giraffe</entry>
        <entry>15 months</entry>
      </row>
      <row>
        <entry>Rhinoceros</entry>
        <entry>14-16 months</entry>
      </row>
      <row>
        <entry>Hippopotamus</entry>
        <entry>7 1/2 months</entry>
      </row>
    </tbody>
  </tgroup>
</table>
```

And there additional elements you can use.

- `<fig>` — A figure. Use this as a wrapper around an `<image>` and a `<title>` to make a figure with a caption.
- `<section>` — A section. Can contain a title. This enables you to add a chunk of content inside of your topic. Use with caution. If you're making a section, why aren't you making a stand alone topic? There are some good reasons, such as making a set of descriptions that are small that you don't need to have individual topics for.

  Be aware that how section titles are handled depend on your processing. They may appear like the next level heading, but they might not appear in the table of contents.
- `<dl>` — A definition list. Often when documenting things, you need to have a term followed by a description. (In fact, this bulleted list really should be a definition list instead.) A definition list takes the following format:

```
<dl>
    <dlentry>
        <dt>Definition term</dt>
        <dd>Definition description. This should contain all sorts of
 descriptive text.
            You can use most other blocks such as <p>, <ul>, <image>
 etc.</dd>
        <dd>You can also have additional <dd> elements since one term
 might have
            multiple definitions.</dd>
    </dlentry>
    <dlentry>
        <dt>Another term</dt>
```

```
        <dd>More descriptive text</dd>
    </dlentry>
</dl>
```

How definition lists display depend on your processing. They can appear as tables or lists that look like glossaries, or whatever your heart desires. You can also nest definition lists, which can get entertaining.

This is just a subset of what you can include. These are basic building blocks that generally don't have semantic meaning.

Semantic tagging

Instead of generic bold or italic, identify content by its intent.

There are dozens of semantic elements that you can learn about using the DITA 1.3 Specification or the OxygenXML DITA Reference, but here is a smattering of elements I use documenting software.

This is a combination of both inline and block elements.

Element	Description	Style
`<apiname>`	The name of an API	inline
`<cmdname>`	The name of a command line tool	inline
`<codeblock>`	For code samples	block
`<codeph>`	For inline code snippets	inline
`<equation-block>`	To contain an equation. If your processor supports it, you can enter MathML, the XML standard for mathematical notation.	block
`<equation-inline>`	To contain an equation. If your processor supports it, you can enter MathML.	inline
`<example>`	A specialization of section to contain content that is an example of something	block
`<filepath>`	For file names and paths	inline
`<keyword>`	Used for keyrefs	inline
`<msgblock>`	Long "messages" from the software	block
`<msgph>`	Short inline "messages" from the software.	inline
`<parmname>`	For parameters	inline
`<ph>`	For phrases - inline content for reuse or conditions	inline
`<screen>`	Examples of command line or terminal windows	block
`<tm>`	Marking words that require a trademark (® or ™)	inline
`<uicontrol>`	A label of an item on a screen or dialog you do something with.	inline

Element	Description	Style
`<userinput>`	Text a user needs to enter	inline
`<varname>`	A variable. Often used with paths or things you need to enter like *myServer* or *port number*	inline
`<wintitle>`	The name of a window, screen, dialog, panel, etc.	inline
`<xmlatt>`	An XML attribute	inline
`<xmlelement>`	An XML element	inline

Insert an image

Inserting images in DITA is similar to adding an image in HTML.

1. Add an `<image>` element.
2. Add the `@href` attribute, providing the relative path to the image file you want to include.
3. If this is a large image, add the `@placement` attribute and set it to `break`.

Insert a link to another topic

Links to other topics are also known as "local" links.

1. Add an `<xref>` element.
2. Add the `@href` attribute with the relative path to the topic you want to link to.

For example, here's a link to Working with topics on page 9. The source is `<xref href="working_with_topics.dita"/>`.

Insert a link to external content

If you want to link to anything on the internet, use an external cross-reference.

1. Add an `<xref>` element.
2. Add the `@href` attribute with the relative path to the topic you want to link to.
3. Add the `@scope` attribute and set it to `external`.
4. Add the `@format` attribute and set it to the type of content you are pointing to. In general, if you are pointing to a web page, use `html`.
5. Provide text for the link as the contents of the `<xref>` element.

For example, here's a link to Google. The source is `<xref href="http://www.google.com" scope="external" format="html">Google</xref>`.

Set up content for a conref

You can reference almost any content using a content reference. Before you can, you have to provide an `@id` attribute.

There are many different types of content references. The basic `@conref` lets you reuse a single element. You can also use `@conrefend` to reference several elements in a row.

One difficulty with `@conref` is that you have to provide a path to the file that contains the element you are referencing. If you decide to move that file, you have to update every reference. This can be onerous in large content sets. Therefore you can use `@conkeyref` instead. This is a combination of `@conref` and `@keyref` so you can reference a DITA key as the path to the file.

No matter the type of content you are referencing, the element you are referencing needs an `@id` attribute.

- All `@id` must be unique.
- Human readable values are much easier to work with

1. Confirm the root `@id` attribute is unique and human readable.
2. Select the element you want to reuse and add a unique and human readable `@id` attribute.

The following paragraph is ready to be referenced.

My first referenced content.

The source looks like this:

```
<p id="example_conref">My first referenced content.</p>
```

Inserting a conref

You must use the same element that you are referencing.

1. Insert the same element you are referencing.

   For example, if you are referencing a <p>, insert a <p> element.
2. Add the `@conref` attribute.
3. Set the value of the `@conref` attribute as follows:

   `Path/to/topic#root_id_value/element_id`

The following paragraph is an example conref.

My first referenced content.

The source looks like this:

```
<p
 conref="set_up_content_for_a_conref.dita#set_up_content_for_a_conref/
example_conref"/>
```

Create a key definition

Keys must be defined in a map file.

You can use DITA keys for short phrases or references, that is, paths to other files.

Short phrases are great for product names and other short variables.

Paths to files let you set up links to files that contain content to be referenced or images. Since you can reference maps from maps, you can set up one map that has paths to all of your high-resolution images and another map that has low-resolution images. Then you can use the high-resolution images in a bookmap that builds a PDF, and the low-resolution for HTML.

- Use the following format for short string.

```
<keydef keys="sample_string">
    <topicmeta>
        <keywords>
            <keyword>Words from a key</keyword>
        </keywords>
    </topicmeta>
</keydef>
```

- Us the following format for a path to a file. This could be an image, another DITA topic, or an external website.

```
<keydef keys="sample_path" href="path/to/file.png"/>
```

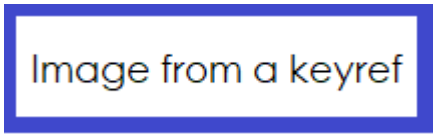
Insert a keyref

Key references are easier to use than content references.

- Add the `@keyref` attribute to an element set to the value of the `@keys` attribute of the value you want to use.

Here is the sample string: Words from a key. The source is `<ph keyref="sample_string"/>`.

The following image uses the `@keyref`.



The source is `<image placement="break" keyref="sample_path"/>`.

Week 11 homework

Document a thing using DITA

Download a trial of Oxygen Author: https://www.oxygenxml.com/xml_author/download_oxygenxml_author.html

I created a ditamap for each of you in the `Week11-DITA/Homework` folder. I also created a folder for each of you in the `Homework` folder for your topics.

Minimally:

Using your ditamap, create at least three new topics.

- A concept
- A task
- A reference

Use some semantic tagging as you author your content.

If you can, document a thing using DITA in your ditamap.

Assuming all automation works correctly, when you commit and push your changes to GitHub, a GitHub action starts to generate a PDF.

The resulting PDF should appear in `mcchandout`.