An Intro to XML

Kyle Miller
EH 401

Contents

What is XML and How is it used?

XML stands for eXtensible Markup Language and is just that, a markup language. XML is used to transport and alter data. The user places tags around text to identify what each line means, its placement on a document, and where data needs to be stored. By inputting information in XML format, data can easily be modified, shared, or moved with the ease of altering one document.

As stated above, XML is used to make data adaptive and accessible. XML makes modifying multiple documents simple through the change of one document. Users create a structure that all other data is modeled after. XML ensures data stays up-to-date, consistent across all versions, and easily editable.

After understanding some of the vocabulary, XML is very easy to use. XML identifies how the data in the document is related. This means when software goes to select a piece of data, it knows what's relevant. XML is used to make data management the easiest job of the day.

Why should I care (as a Technical Communicator)?

You might be asking yourself why you should care about some programming language as a Technical Communicator. But I ask why wouldn't you care? Technical Communicators manipulate all sorts of information, whether it's creating a user guide, compiling a list of Q&As, or updating information on the company website. XML makes the process significantly more straight forward. The best part about learning XML is you (probably) won't even have to write anything in XML format: just read it and understand what you need!

Now, there are some drawbacks to using XML. The first is the oddity of it! It looks like a garbled mess of characters with little connection. Another is taking the time to learn a completely new system of formatting information.

The second drawback is one of the biggest hurdles of using XML so this guide will focus on providing the base amount of knowledge needed to read an XML formatted document.

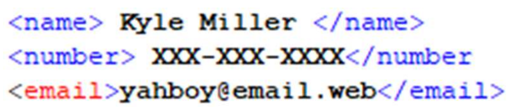
XML Terminology

Each XML document must be declared at the beginning. The declaration is the same each time and should be copy pasted before starting. The declaration can be ignored when viewing the document and looks like such:

<?xml version=”1.0” encoding=”UTF-8”?>

Tags

Tags have already been mentioned, but they are placed at the beginning and ending of information to denote what is what. Below, an example of tags is being used to identify different parts of an employee’s contact information.

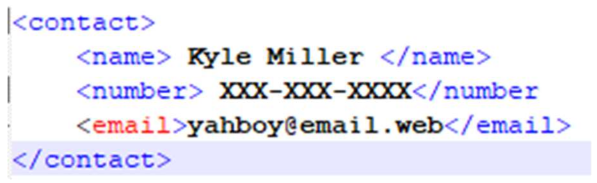


```
<name> Kyle Miller </name>
<number> XXX-XXX-XXXX</number
<email>yahboy@email.web</email>
```

The tags begin with a “<” and end with a “>”. The closing tags always include a “</” to close out the tag. Tags are user-defined, so each can be shaped to fit the elements of the document.

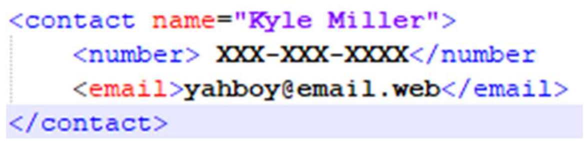
Elements

Elements are essentially everything in an XML document. The tags, text inside the tags, or even tags containing tags are all elements. Take the example from above. If we wanted to make a list of different employee’s contact information, we could create a tag labeled contact enveloping the employee’s information.



```
<contact>
    <name> Kyle Miller </name>
    <number> XXX-XXX-XXXX</number
    <email>yahboy@email.web</email>
</contact>
```

Each part of that example is an element. Elements can also include an attribute to connect data to specific elements. For example, we could add an attribute to the above example to include the employee’s name with the contact tag.



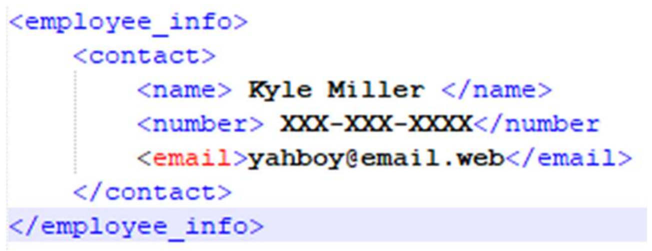
```
<contact name="Kyle Miller">
    <number> XXX-XXX-XXXX</number
    <email>yahboy@email.web</email>
</contact>
```

There is no difference between the two implementations and XML doesn't care which method is used.

Element Hierarchy

An XML document contains several pieces of information, so you will see many different tags declared. How do we know which tag comes first, second, or last? XML formats the different tags like a tree where each different line is a separate branch.

A root element, a parent element, and a child element all make up the different parts of the tree. The root element must be unique. For the previous example, contact would be the root element of the document and parent element of name, number, and email. We can add another tag to change our root element and add another layer to the example.



```
<employee_info>
    <contact>
        <name> Kyle Miller </name>
        <number> XXX-XXX-XXXX</number
        <email>yahboy@email.web</email>
    </contact>
</employee_info>
```

Now, we have the root element employee_info. It's the root element because it starts the document. It's also the parent element to contact. Contact has three child elements in name, number, and email.

In-Text Documentation and Formatting

Another aspect of XML to *comment* about are comments! These are basically notes included by the documents author to add insight or clarity to something. These are opened by "<!--" and ended with "-->". These aren't mandatory, but they can help those who did not write the document understand certain parts of it.

One very important thing to note is XML requires the document to be well-formed. Thankfully this can easily be done by verifying a few things:

- Make sure each tag you create is closed using "</"and each tag is case sensitive.
- Do not forget the root element of the document.
- Elements should be properly nested.

Each document must follow correct syntax and there are online tools you can use to tell what errors exist in a XML document.

If we wanted to test the XML I've used in this example, we would get a few errors due to tags not being properly closed! XML also checks if the document is valid. This requires the document to be both well-formed and conforming to a standard structure.

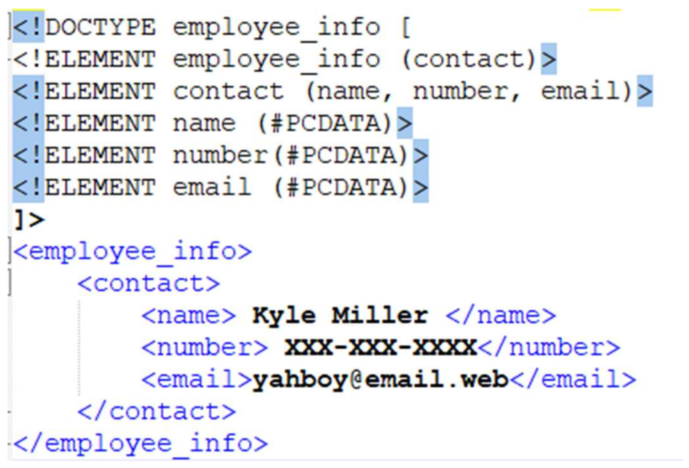
Structure in XML

Congrats! You know the basics of XML. Now you can dissect any XML document to pull the proper information out. Every document you see will contain parts of the examples above; However, there will be more to the XML documents when working professionally.

Let's ask a hypothetical. If two separate teams are working on the same project, and they tag each element differently, then how will the data match? XML documents need structure to define what each element of the document accomplishes and to ensure the data will match. There are two methods of doing this: DTDs and Schemas.

DTD's

Document Type Definition (DTD) defines what elements the XML document will include. The root, parent and child elements, the type of data the information is, and the placement of said data is all defined in the DTD. DTDs can be declared externally as a different file or internally on the same XML document.

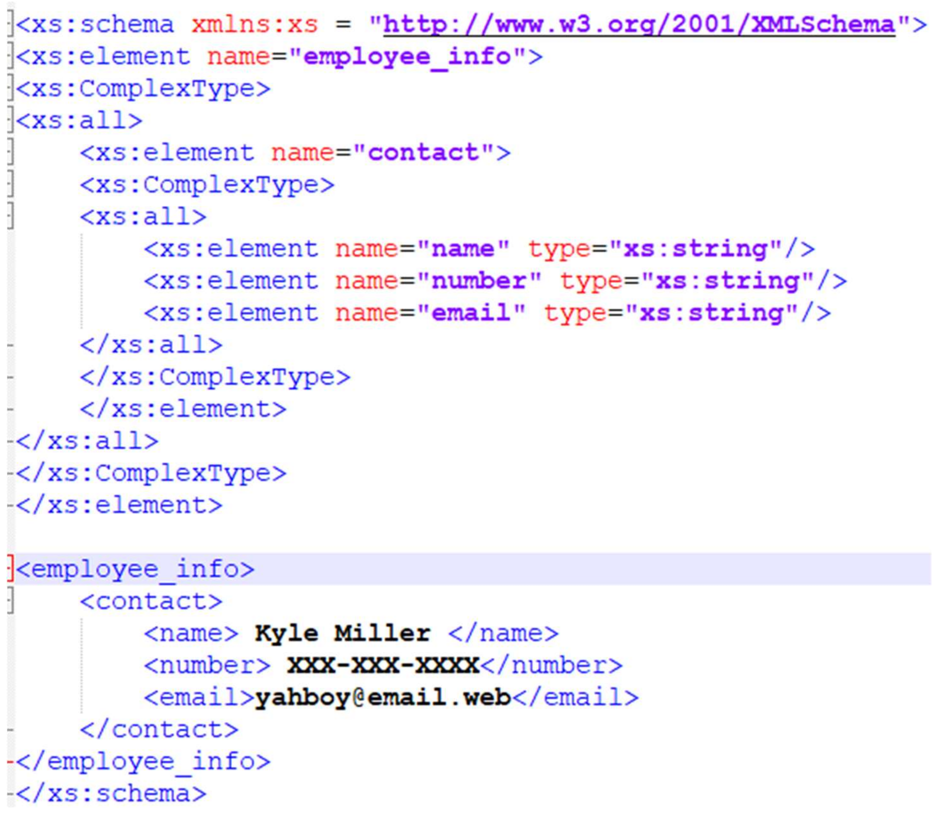


```
<!DOCTYPE employee_info [
<!ELEMENT employee_info (contact)>
<!ELEMENT contact (name, number, email)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT number(#PCDATA)>
<!ELEMENT email (#PCDATA)>
]>
<employee_info>
    <contact>
        <name> Kyle Miller </name>
        <number> XXX-XXX-XXXX</number>
        <email>yahboy@email.web</email>
    </contact>
</employee_info>
```

Above is the DTD example. !DOCTYPE employee_info tells us what the root element should be. !ELEMENT employee_info tells what the element must include. The same thing applies to !ELEMENT contact. Contact has to include the name, number, and email of the employee. The name, number and email elements are child elements, so they are given #PCDATA to denote they cannot include additional elements.

Schemas

Schemas do the same thing as DTDs, but schemas are written in an XML based format. DTDs are modeled after a different markup language that predates XML. Another key difference between the two is that schemas are just easier to understand. They don't require learning parts of a new language. Schemas also allow the user to set restrictions and bounds for data and it can convert datatypes.



```
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
<xs:element name="employee_info">
<xs:ComplexType>
<xs:all>
    <xs:element name="contact">
    <xs:ComplexType>
    <xs:all>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="number" type="xs:string"/>
        <xs:element name="email" type="xs:string"/>
    </xs:all>
    </xs:ComplexType>
    </xs:element>
</xs:all>
</xs:ComplexType>
</xs:element>

<employee_info>
    <contact>
        <name> Kyle Miller </name>
        <number> XXX-XXX-XXXX</number>
        <email>yahboy@email.web</email>
    </contact>
</employee_info>
</xs:schema>
```

In this Schema example, things look more like XML format. The first line included is another declaration that is copy and pasted into the document. It simply states we are using S7chema structuring. <xs:element> defines what information is an element. If the element is a root or parent element, it must include <xs:ComplexType>. <xs:all> shows that each child element can appear only once in any order. There are other indicators we can include with schemas that alter what and how elements appear. The type= indicator functions similar to the #PCDATA by denoting what type of information the text can be. Schema also has different indicators that limit what data the element can be. The last thing to note about schemas is the last line. You must include the closing </xs:schema> tag after all your other elements. Otherwise, the document will not validate.

DTDs and Schemas are like Nike and Under Armor. The two brands will cloth you, but in slightly different looking ways. Structure is the most important part of any good XML document. Without a well-formed structure, there wouldn't be any point to implementing XML on a large scale!

Additional Resources

This guide covers the basics of XML and briefly mentions some higher-level aspects of XML. There are several resources that make using and reading XML an easy task. The first thing I recommend is using a file reader to view XML files. As seen in my examples, the software will highlight matching parts of the document, and color codes items to help differentiate between information. It really makes a difference! Notepad ++ is my personal recommendation but there are multiple available.

In terms of better understanding XML, the w3schools website offers an in-depth look into all the parts of XML, DTD's, and Schemas. XML.com is another good resource. It's a forum-type of website covering all things XML related. And of course, YouTube is invaluable with many channels offering explanations of XML. Between this guide and those two resources, XML can become a powerful tool in any Technical Communicators arsenal.