**Unit 2**

**XML Fundamentals Part 1 & 2**

**Elements**

* Elements are fundamental building blocks of every XML document

<height>107</height>

* *Closing Tags* –every element must have a closing tag, except for *self-closing* elements
* *No Punctuation*—don’t use colons, hyphen, periods in your tag names, though a parser will allow them allow, they are not a good convention.
	+ Punctuation like **: colons** are used in *namespaces*—we will talk about that later in course
* *Case matters*—don’t mix cases. In general lower case is typically used for tag names.
	+ Note—in general, it’s good to pick a convention and stick with it, and try to use one that is used by the XML community.
* *Well Formed XML or “well-formedness”*
	+ All of the things we discussed, closing tags, a root element, all are part of making your document well-formed. If you are XML document is not well formed, a parser will not see it as XML and throw and error.
	+ This is different from ***valid*** XML or ***validating*** your XML, which we will talk about when we do DTDs and Schema
	+ A well-formed document means you following the rules for writing XML

**Explicit Structure**

* Ensure that you are explicit with how you label and structure your document.
	+ Ex. <name> Neil Percival Young</name>
	+ Do this instead:

 <first\_name>Neil</first\_name>

 <middle\_name>Percival</middle\_name>

 <last\_name>Young</last\_name>

**Nesting**

* in general, use nesting to *structure* your content, and to make help make elements explicit
	+ Nesting allows to make **Parent/Child Relationships**
	+ Nesting allows to show *hierarchy* in our XML
	+ Nesting lets you show the difference between the *general* and the *particular*
	+ Nesting is good to a point. Don’t use it unless if serves a distinct purpose.

<name>

 <first>Neil</first>

 <middle>Percival</middle>

 <last>Young</last>

</name>

XML must be properly nested:

<name>

 <first>Neil</first>

 <middle>Percival</middle>

 <last>Young*</name>*

</last>

**Create Verbose Markup**

* Ensure that you XML is easy to read. It may be well formed, and perfectly legal XML, but impossible to read.
	+ Our XML editor will help us with that.
	+ Use clear, but concise tag names.

**Whitespace**

* Whitespace is striped out in HTML, but in XML it is left in. The whitespace is considered “**Parsed Character Data”** or **PCDATA** and will be passed onto the browser for viewing. But, *carriage returns* and *line feeds* in your document will be stripped out.
* Avoid excessive use of carriage returns. If you need a line break, then you need to mark it up that way. See example in Rusty Harold’s *Effective XML*:

**DON’T DO THIS:**

<poem type="sonnet" poet="Eleanor Alexander">

 For me, my friend, no grave-side vigil keep

 With tears that memory and remorse might fill;

 Give me your tenderest laughter earth-bound still,

 And when I die you shall not want to weep.

 No epitaph for me with virtues deep

 Punctured in marble pitiless and chill:

 But when play time is over, if you will,

 The songs that soothe beloved babes to sleep.

 </poem>

**DO THIS**

<poem type="sonnet" poet="Eleanor Alexander">

 <stanza>

 <line>For me, my friend, no grave-side vigil keep</line>

 <line>With tears that memory and remorse might fill;</line>

 <line>Give me your tenderest laughter earth-bound still,</line>

 <line>And when I die you shall not want to weep.</line>

 <line>No epitaph for me with virtues deep</line>

 <line>Punctured in marble pitiless and chill:</line>

 <line>But when play time is over, if you will,</line>

 <line>The songs that soothe beloved babes to sleep.</line>

</stanza>

* In general, you will not need to worry about this kind of formatting. Your main goal will be not to use the return key in your documents.

**CDATA**

* is basically unparsed text or character data. The parser does not treat it as XML. It is often used in order to mark up XML syntax. <![CDATA [ …xml syntax goes here…]]> not used much in the professional world, and we will not use it in this course.

**Attributes**

* Attributes are a way to designate information about an element. They are essentially used for element **metadata** or data about the data contained in the element. Use them sparingly. They are typically used for information that is only needed by the content creators, and the content users will not need to see. Many DTDs that have been created over the past 20 years do use them, so it’s important that you understand how to use them and their role. Increasingly not used as much in new content models.
	+ Example: <engine size=”liters”>3.2</engine>
* We might think of Attributes as “name-value” pairs
* In other words: “Attributes are used to add additional information the element without adding text to the element itself. It refines the meaning of the element’s tag name.

**Naming Conventions**

* Pick a naming convention, and stick to it. You will see that Hunter uses <Name> a capital letter at the beginning of every element. Goldberg uses all lower case. When a compound word is used, Hunter uses what is known as “CamelCase” like <LastName> in the programming world. Goldberg uses an underscore <last\_name>

**Comments**

* Comments are important because they help you document your own work so that you know why you coded something a certain way. It also is a reference for others so that they know why you did what you did.
* They are also helpful for when you are debugging your projects if you want to hide XML from the parser.

Example: <!-- comments go here -->

**XML Declaration**

* Every XML document should have a *declaration*
* You should use UTF 8. Your projects should still work without this designation; down the road it will be important. Unicode is what allows your content to be *internationalized* to any language.
* Use the UTF-8 encoding for all of your multilingual, international, or non-English data and documents.

*<?xml version="1.0" encoding="UTF-8"?>*

* *Root Element* – every XML doc. must contain 1 and **ONLY** 1 root element.
* You must have a minimum of <?xml version=”1.0”?>

**Parsers**

* Your XML must be read by a parser. There are many parsers that are out there. Parsers read XML differently, and will look at that in our brief exercise. There’s a parser inside your XML NotePAD++ and each Browser flavor has its own browser, each will parse the document slightly differently. Netscape/Chrome/Firefox or IE or Safari the next version of IE has been given the code name “Spartan”

**Predefined Entities**

* Avoid using these symbols in your content:
	+ &, <, >, “, ‘,
	+ These are reserved by the parser for other things, so you must use a special sequence of letters to tell the parser that you want an ampersand &amp and so on. Look at page 14 of you book for all the

**There are 5 predefined entity references in XML:**

|  |  |  |
| --- | --- | --- |
| **&lt;** | **<** | **less than** |
| **&gt;** | **>** | **greater than** |
| **&amp;** | **&** | **ampersand** |
| **&apos;** | **'** | **apostrophe** |
| **&quot;** | **"** | **quotation mark** |

* This is also known as “escaping” characters.

**Example XML doc**

<?xml version="1.0"?>

<!--put comments here-->

<musician id=”001”>

<name>

 <first>**Neil**</first>

 <middle>**Percival**</middle>

 <last>**Young**</last>

</name>

<birth\_location>

 <city>**Toronto**</city>

 <state\_province>**Ontario**</state\_province>

<country>**Canada**</country>

</birth\_location>

</musician>