Meta Information

BSc Information Technology – Yr 2

Semester 2

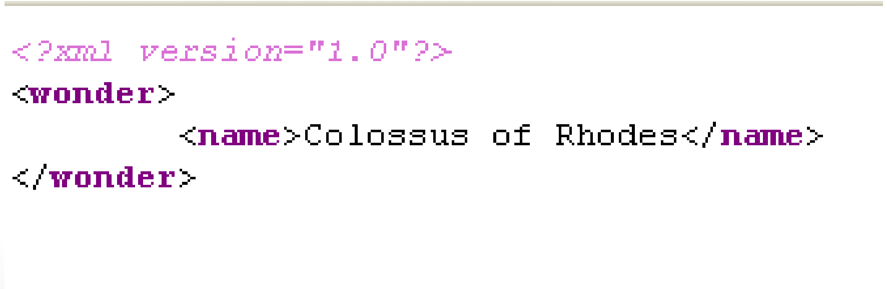
Dr. Brenda Mullally

XML Rules

Rules for Writing XML:

XML has a structure that is extremely regular and predictable. It is defined by a set of rules, the most important of which are described below. If your document satisfies these rules, it is considered well-formed.

1. A root element is required. – Every XML document must contain one, and only one root element. This root element contains all other elements in the document. The only pieces of XML allowed outside the root element are the comments and processing instructions.



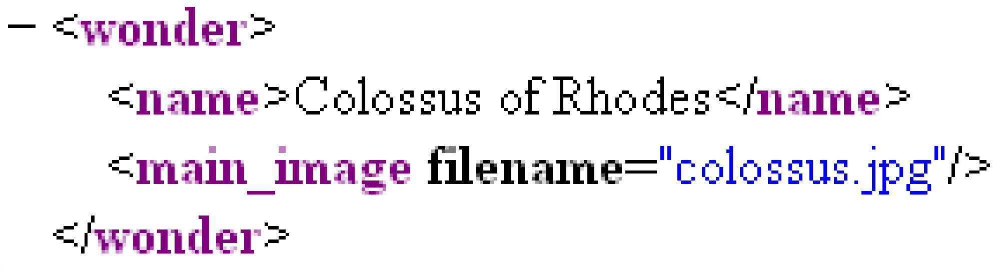
```
<?xml version="1.0"?>
<wonder>
        <name>Colossus of Rhodes</name>
</wonder>
```

One root element (wonder), first line is outside of root as is a processing instruction not part of the XML data.

XML Rules

Rules for writing XML:

2. Closing tags are required. – Every element must have a closing tag. Empty elements can use a separate closing tag, or an all-in-one opening and closing tag with a slash before the final >.

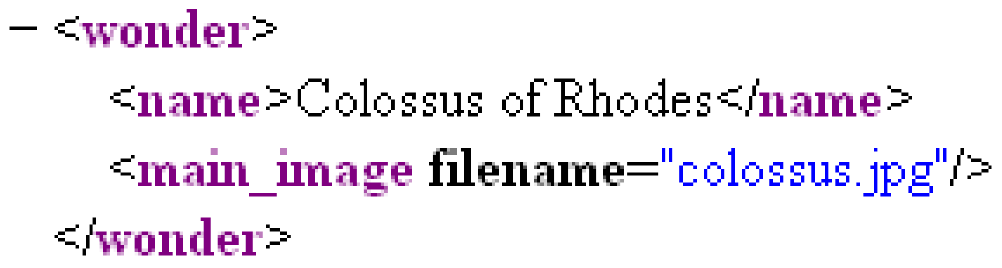


```
- <wonder>
    <name>Colossus of Rhodes</name>
    <main_image filename="colossus.jpg"/>
  </wonder>
```

XML Rules

Rules for writing XML:

3. Elements must be properly nested. – If you start element A, then start element B, you must first close element B and then close element A.

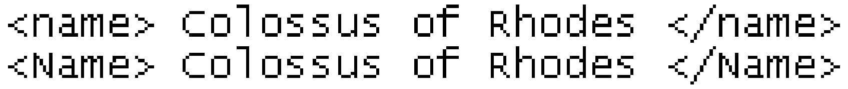


```
– <wonder>
     <name>Colossus of Rhodes</name>
     <main_image filename="colossus.jpg"/>
  </wonder>
```

XML Rules

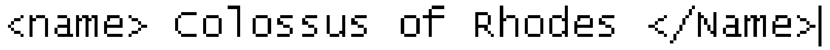
Rules for writing XML:

4. Case matters. – XML is case sensitive. Elements named wonder, WONDER, and Wonder are considered entirely separate and unrelated to each other.



```
<name> Colossus of Rhodes </name>
<Name> Colossus of Rhodes </Name>
```

Valid XML



```
<name> Colossus of Rhodes </Name>
```

Invalid XML

XML Rules

Rules for writing XML:

5. Values must be enclosed in quotation marks. – An attribute’s value must always be enclosed in either matching single or double quotation marks.

The quotation marks are required. They can be single or double, as long as they match each other.

```
<main_image file="colossus.jpg"/>
```

XML Rules

A **well-formed** XML document is one that complies with the following rules:

- XML documents must have a root element
- XML elements must have a closing tag
- XML tags are case sensitive
- XML elements must be properly nested
- XML attribute values must be quoted

- A "**valid**" XML document is one which is well-formed, but also conforms to a DTD.

Document Type Definitions

Ensure Data Integrity:

Trust is important for data – that it hasn’t been corrupted, truncated, mistyped or left incomplete. Broken documents can confuse software, format as gibberish, and result in erroneous calculations.

Transmitting and converting documents always entails risk that some information may be lost.

XML gives you the ability to guarantee a level of trust in data.

First there is well-formedness, every XML parser is required to report syntax errors in markup.

Document Type Definitions

Missing tags, malformed tags, illegal characters, and other problems should be immediately reported to you by the XML parser.

```
<announcement<
   <TEXT>Hello, world! I’m using XML  & XSLT</Text>
</anouncement>
```

Two mismatched tags and an illegal character will be found by the parser, as well as showing you where the errors occurred.

Document Type Definitions

Checking if a document is well-formed can pick up a lot of problems:

Mismatched tags, a common occurrence if you are typing in the XML by hand. The start and end tags have to match exactly in case and spelling.

Truncated documents, which would be missing at least part of the outermost document (both start and end tags must be present).

Illegal characters, including reserved markup delimiters like <,>, and &.

Document Type Definitions

The well-formedness check has its limits. The parser doesn’t know if you are using the right elements in the right places.

For example, you can have HTML document with a p element inside the head, which is illegal. To catch this kind of problem, you need to test if the document is a valid instance of HTML. The tool for this is a validating parser.

A validating parser works by comparing a document against a set of rules called a document model. One kind of document model is a document type definition (DTD). It declares all elements that are allowed in a document and describes in detail what kind of elements they can contain.

Document Type Definitions

The most important benefit to using a DTD is that it allows you to enforce and formalise a markup language. You can make your DTD public by posting it on the web, which is what organisations like W3C do. For example you can see the DTD for “strict” XHTML version 1.0.

Document Type Definitions

- The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements:

```
  <!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

```
<!DOCTYPE note SYSTEM “note.dtd”>
```

Document Type Definitions

- A limitation of DTD’s is that they don’t do much checking of text content. You can declare an element to contain text, or not, that is as far as it goes. You can not check if an element should be filled out is empty, or if it follows the wrong pattern. There is no way to testing in a DTD that an element is filled in or not.
- An alternative document modelling scheme provides the solution. XML Schemas provide much more detailed control over a document, including the ability to compare text with a pattern you define.
- So there are several levels of quality assurance available in XML, all of which will be dealt with during this module.

DTDs – Element Type Declarations

- Identify the rules for elements that can occur in the XML document. Options for repetition are:
  - * indicates zero or more occurrences for an element;
  - + indicates one or more occurrences for an element;
  - ? indicates either zero occurrences or exactly one occurrence for an element.
- Name with no qualifying punctuation must occur exactly once.
- Commas between element names indicate they must occur in succession; if commas omitted, elements can occur in any order.

XML naming rules

XML elements must follow these naming rules:

- Names can contain letters, numbers, and other characters
- Names cannot start with a number or punctuation character
- Names cannot start with the letters xml (or XML, or Xml, etc)
- Names cannot contain spaces

Viewing XML

The XML document will be displayed with colour-coded root and child elements. A plus (+) or minus sign (-) to the left of the elements can be clicked to expand or collapse the element structure. To view the raw XML source (without the + and - signs), select "View Page Source" or "View Source" from the browser menu.

If an erroneous XML file is opened, the browser will report the error.

note_error.xml

XML Basics

XML Errors Will Stop You

Errors in XML documents will stop your XML applications.

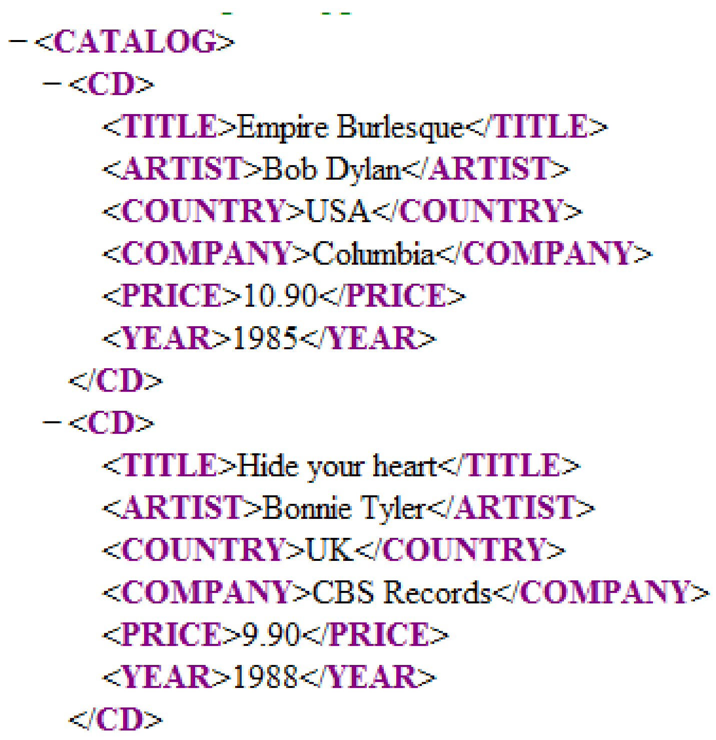
The W3C XML specification states that a program should stop processing an XML document if it finds an error. The reason is that XML software should be small, fast, and compatible.

HTML browsers will display documents with errors (like missing end tags). HTML browsers are big and incompatible because they have a lot of unnecessary code to deal with (and display) HTML errors.

With XML, errors are not allowed.

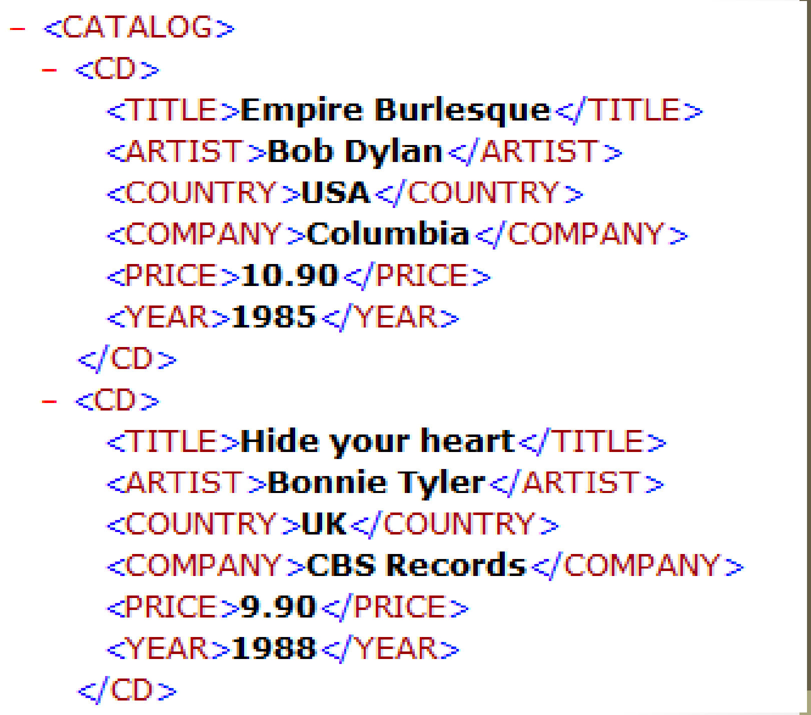
Viewing XML

Firefox



```
- <CATALOG>
  - <CD>
      <TITLE>Empire Burlesque</TITLE>
      <ARTIST>Bob Dylan</ARTIST>
      <COUNTRY>USA</COUNTRY>
      <COMPANY>Columbia</COMPANY>
      <PRICE>10.90</PRICE>
      <YEAR>1985</YEAR>
    </CD>
  - <CD>
      <TITLE>Hide your heart</TITLE>
      <ARTIST>Bonnie Tyler</ARTIST>
      <COUNTRY>UK</COUNTRY>
      <COMPANY>CBS Records</COMPANY>
      <PRICE>9.90</PRICE>
      <YEAR>1988</YEAR>
    </CD>
```

Internet Explorer



```
- <CATALOG>
  - <CD>
      <TITLE>Empire Burlesque</TITLE>
      <ARTIST>Bob Dylan</ARTIST>
      <COUNTRY>USA</COUNTRY>
      <COMPANY>Columbia</COMPANY>
      <PRICE>10.90</PRICE>
      <YEAR>1985</YEAR>
    </CD>
  - <CD>
      <TITLE>Hide your heart</TITLE>
      <ARTIST>Bonnie Tyler</ARTIST>
      <COUNTRY>UK</COUNTRY>
      <COMPANY>CBS Records</COMPANY>
      <PRICE>9.90</PRICE>
      <YEAR>1988</YEAR>
    </CD>
```

Viewing XML

- **Why Does XML Display Like This?**
- XML documents do not carry information about how to display the data.
- Since XML tags are "invented" by the author of the XML document, browsers do not know if a tag like <table> describes an HTML table or a dining table.
- Without any information about how to display the data, most browsers will just display the XML document as it is.
- In the next chapters, we will take a look at different solutions to the display problem, using CSS, XSLT and JavaScript.

XML Basics

You have by now created a root element and one child element.

```
<?xml version="1.0"?>
<ancient_wonders>
  <wonder>Colossus of Rhodes</wonder>
<ancient_wonders>
```

In the this example you can see how elements are nested. <title> is nested within <book>.

```
<?xml version="1.0"?>
<book_store>
  <book>
     <title>Everyday Italian</title>
     <author>Ciada De Laurentis</author>
  </book>
</book_store>
```

XML Basics

Often when creating your XML document, you will want to break down your data into smaller pieces. In XML, you can create child elements of child elements of child elements, etc.

The ability to nest multiple levels of child elements enables you to identify and work with individual parts of your data and establish a hierarchical relationship between these individual parts.

If we didn’t nest our child elements our XML file would look something like this:

```
<?xml version=“1.0”?>
<book_store>
  <book>Everyday Italian</book>
  <author>Ciada De Laurentis</author>
  <book>Harry Potter</book>
  <author>J.K. Rowling</author>
</book_store>
```

What is wrong with doing it this way?

XML Basics

When you are nesting child elements you must ensure to abide by the rules of XML that the order of opening and closing nested tags are correct.

Open the first or outer element, then open the child or inner element , create the content for that child element, close the child element, repeat the last three steps as desired, then finally close the outer element.

```
<?xml version="1.0"?>
<ancient_wonders>
  <wonder>
    <name>Colossus of Rhodes</name>
    <location>Rhodes, Greece</location>
    <height>107</height>
  </wonder>
<ancient_wonders>
```

XML Basics

Adding attributes:

An attribute stores additional information about an element, without adding text to the element’s content itself. Attributes are known as “name-value” pairs, and are contained within the opening tag of an element.

To add an attribute:

> Before the closing > of the opening tag, type **attribute=**, where **attribute** is the word that identifies the additional data. Then type **“value”**, where value is that additional data. The quotes are required.

No two attribute names in the one element can have the same name.

Attribute values must be in quotes.

If the value is to contain quotes then start with double quotes and use single quotes in the value, ending with double quotes..

XML Basics

```
<name language=“English”>Colossus of Rhodes</name>

<title comment=“it’s open”>Door</title>
```

Attributes should be used as “metadata”; that is, data about data. Attributes should be used to store information about the element’s content, and not the content itself.

There are no rules about when to use attributes or when to use elements. Attributes are handy in HTML. In XML my advice is to avoid them. Use elements instead.

XML Basics

XML Elements vs. Attributes

Take a look at these examples:

```
 <person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
<person>
  <gender>female</gender>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

In the first example gender is an attribute. In the last, gender is an element. Both examples provide the same information.

XML Basics

Attributes vs Elements

Some of the problems with using attributes are:

Attributes cannot contain multiple values (elements can)

Attributes cannot contain tree structures (elements can)

Attributes are not easily expandable (for future changes)

Attributes are difficult to read and maintain.

Use elements for data.

Use attributes for information that is not relevant to the data.

Don't end up like this:

```
<note day="10" month="01" year="2008"
   to="Tove" from="Jani" heading="Reminder"
   body="Don't forget me this weekend!">
   </note>
```

XML Basics

XML Attributes for Metadata

Sometimes ID references are assigned to elements. These IDs can be used to identify XML elements in much the same way as the id attribute in HTML. This example demonstrates this:

The id attributes above are for identifying the different notes. It is not a part of the note itself.

Metadata (data about data) should be stored as attributes, and the data itself should be stored as elements.

XML Basics

Attributes vs Elements

```
 <messages>
 <note id="501">
   <to>Tove</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this weekend!</body>
 </note>
 <note id="502">
   <to>Jani</to>
   <from>Tove</from>
   <heading>Re: Reminder</heading>
   <body>I will not</body>
 </note>
</messages>
```

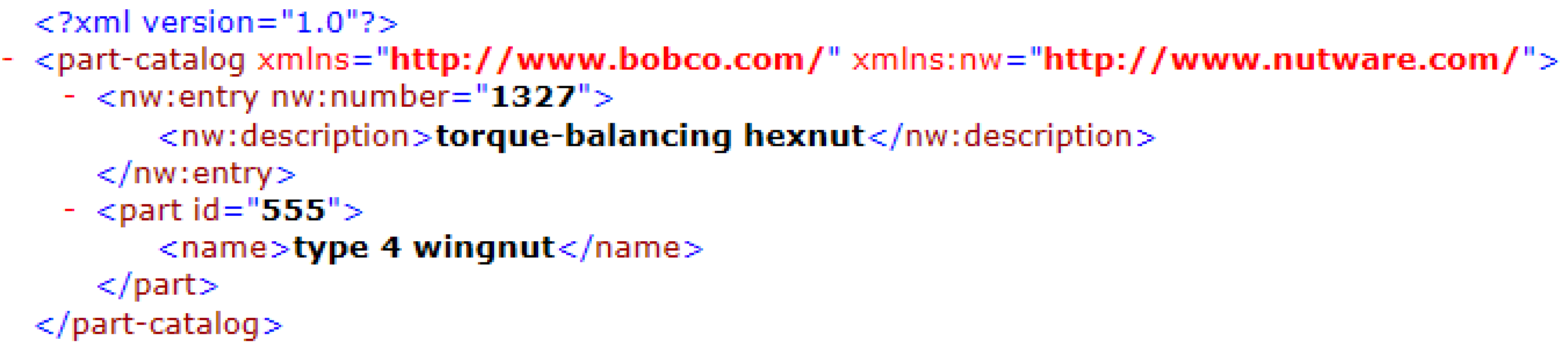
XML Basics

Namespaces

Namespaces are a mechanism by which element and attributes names can be assigned to groups. They are most often used when combining different vocabularies in the same document.

The following example shows that the part-catalog element contains two namespaces which are declared by the attributes xmlns:nw and xmlns. The elements inside part-catalog and their attributes belong to one or other namespace. Those in the first namespace can be identified by the prefix nw.

XML Basics



```
  <?xml version="1.0"?>
- <part-catalog xmlns="http://www.bobco.com/" xmlns:nw="http://www.nutware.com/">
  - <nw:entry nw:number="1327">
      <nw:description>torque-balancing hexnut</nw:description>
    </nw:entry>
  - <part id="555">
      <name>type 4 wingnut</name>
    </part>
  </part-catalog>
```

The attributes of part-catalog are called namespace declarations. The general form of a namespace declaration is to start with the keyword xmlns: followed by the namespace prefix, and equals sign, and a namespace identifier in quotes.

XML Basics

In a special form of the declaration, the colon and namespace prefix are left out, creating an implicit (unnamed) namespace. The first namespace declaration in the previous example is an implicit namespace.

part-catalog and any of its descendants without the namespace prefix nw: belong to the implicit namespace.

To include an element or attribute in a namespace other than the implicit namespace, you must do so as follows:

XML Basics

Fully qualified name:

To the left of the colon is the namespace prefix, and to the right of the colon is the local name.

namespace prefix : local name

<nw:description>

Namespaces only affect a limited area in the document. The element containing the declaration and all of its descendants are in the scope of the namespace.

XML Basics

Namespace identifiers are assigned a URL. This is not a requirement, however. The XML parser does not lookup any information located at that site. The site may not even exist.

So why use the URL?

The namespace must have a unique identifier. URLs are unique. They often contain information about the company or organisation so it makes a good candidate.

Still many have made the point that URLs are not really meant to be used as identifiers. Resources are often moved and URLs change. But since there is currently no alternative it looks like the practice is here to stay.

XML Basics

Using empty elements

Empty elements are elements that do not have any content of their own. Instead, they will have attributes to store data about the element. For example, you might have a main_image element with an attribute containing the filename of an image, but it has no text content at all.

To write an empty element with a single opening/closing tag.

```
<main_image file="colossus.jpg" w="528" h="349"/>
```

To write an empty element with separate opening and closing tags.

```
<main_image file="colossus.jpg" w="528" h="349"></main_image>
```

In XML, both the single empty element and the separate open and closing tags for an empty element are the same. It is personal preference as to which technique you prefer to use.

XML Basics

Writing comments

It is often useful to annotate your XML documents so that you know why you used a particular element, or what a piece of information specifically means.

As with HTML, you can insert comments into your XML documents, and they will not be parsed by the processor.

To write a comment:

<!– write comment here -->

Comments can contain spaces, text, elements, and line breaks, and can therefore span multiple lines of XML.

No spaces are required between the double hyphens and the content of the comment.

You may not use a double hyphen within a comment.

XML Basics

Writing comments

You may not nest comments.

You may use comments to hide a piece of your XML code during development or debugging. The elements within the commented section along with any errors they may contain, will not be processed by the XML processor.

Commenting are also useful for documenting the structure of an XML document, in order to facilitate changes and updates in the future.

XML Basics

Predefined Entities – five special symbols

Unlike HTML, there are only five predefined entities in XML:

&amp creates an ampersand & character
&lt creates a less than sign <
&gt creates a greater than sign >
&quot creates a double quotation mark “
&apos creates a single quotation mark ‘

XML Basics

Predefined Entities

Predefined entities exist in XML because each of these characters have specific meanings. For example if you used a < within the text value of an element or attribute, the XML parser would think you were starting a new element.

You may not use < or & anywhere in your XML document, except to begin a tag or an entity respectively. If you need to use one of these characters within the text value of an element or attribute, you must use one of the predefined entities.

You may use a ‘, “ or > within the text value of an element or attribute. However, when using “ or ‘, be sure that you do not match an existing quote. It is advisable to use the predefined entity for > to avoid any confusion.

XML Basics

Displaying elements as text:

If you want to write about XML elements and attributes in your XML documents, you will want to keep the XML processor from interpreting them, and instead just display them as regular text. To do this, you enclose the information in a CDATA section.

```
<![CDATA[
  <ancient_wonders>
    <wonder>
      <name language="English"> colossus of Rhodes</name>
      <location>Rhodes, Greece</location>
    </wonder>
</ancient_wonders>
]]>
```

XML Basics

Displaying elements as text:

Two other common uses for the CDATA section are to enclose HTML and JavaScript so that they are not parsed by the XML parser.

CDATA stands for (unparsed) Character Data, meaning that the CDATA content will not be interpreted by the XML processor. This is opposed to PCDATA which is parsed character data – we will cover later.

The predefined entities will not work within the CDATA section so if you want display an & then type &, as &amp will not work.

You may not nest CDATA sections.

CDATA can be used anywhere within the root element of an XML document.