XML Terminology

As with all technologies, XML has its own terminology. Before we get into creating XML documents, it is important that you understand the terms that are going to be used throughout the rest of this course.

Tags

If you are already familiar with HTML, then you are already familiar with the concept of tags. If not, a tag is an identifier to an element, which we will discuss next. Just like HTML, tags are identified using less-than (<) and greater-than (>) symbols. These symbols are considered *markup* and the text between them is known as the tag. Take a look at the following HTML example:

```
<title>This is the title of my HTML page</title>
```

The tag we are using here is the "title" tag. Notice that it has a start tag (<title>), and a different version of the tag at the end (</title>). The slash in the second tag is called a *terminator tag*. The difference between HTML tags and XML tags is that in XML you can name the tags what you want (provided you follow some basic rules). In our example XML document, we have defined lots of tags. In the following example, there are three different tags that are defined: <name>, <first> and <last>.

```
<name>
  <first>John</first>
  <last>Doe</last>
</name>
```

It is very important to understand that in XML you have control of the tag names. There are only a few rules that you have to follow when it comes to naming your tags.

The tag name must contain one letter (A-Z or a-z).

The tag name can contain digits, but they cannot be the first character.

Providing that you adhere to rule 1, you can begin your tag name with an underscore (_) or a colon (:).

Spaces and/or tags are not allowed in tag names.

The only punctuation signs that you can use in your tag name are the hyphen (-) and a full stop (.). You cannot use underscores to separate long names (e.g. <this_is_my_long_tag_name>). Even though an underscore is allowed as a first character, it is not allowed anywhere else. However, you can use a hyphen or full stop to get the same effect (e.g. <this.is.my.long.tag.name> or <this-is-my-long-tag-name>). Use tag names that make sense. This isn't really a rule because it can not be enforced by an XML processor. However, I recommend that it be a self-imposed rule. One of the features of XML is its readability. The following XML would be well-formed, valid, and perfectly usable, but it is far from readable.

```
<ab>
  <cd>David</cd>
  <_>Doe</_>
</ab>
```

Other than these enforcing these simple rules, XML let's you name your tags as you please.

Elements

In XML documents, data is stored in elements. Elements are identified using a start tag and an end tag. The name of the tag is defined by the author. For those who are familiar with HTML, you immediately can see the difference between HTML and XML in the way that the tags are named. In HTML, you have a predefined set of tags to work with. In XML, you can create your own tag names so that it makes logical sense for what you are doing.

In our example document, we have several elements. For example, the following element called <position> contains a value of "programmer". Note that the end tag uses a backslash ("/") to denote the end of the element.

```
<position>Programmer</position>
```

Elements can be empty, meaning that they do not contain any data. For example, if we did not have a value for the <position> element it would look like this:

```
<position></position>
```

In XML, it is not necessary to use two tags to represent an empty element. Instead, you can use an *end tag*, which allows you to create a start and an end tag within one element tag. An end tag has a backslash incorporated into it as shown:

```
<position/>
```

is equivalent to:

```
<position></position>
```

Attributes

Element tags can contain *attributes*, which give further information about the elements they delimit. An example of an attribute can be seen in the <employee> tag of our example document.

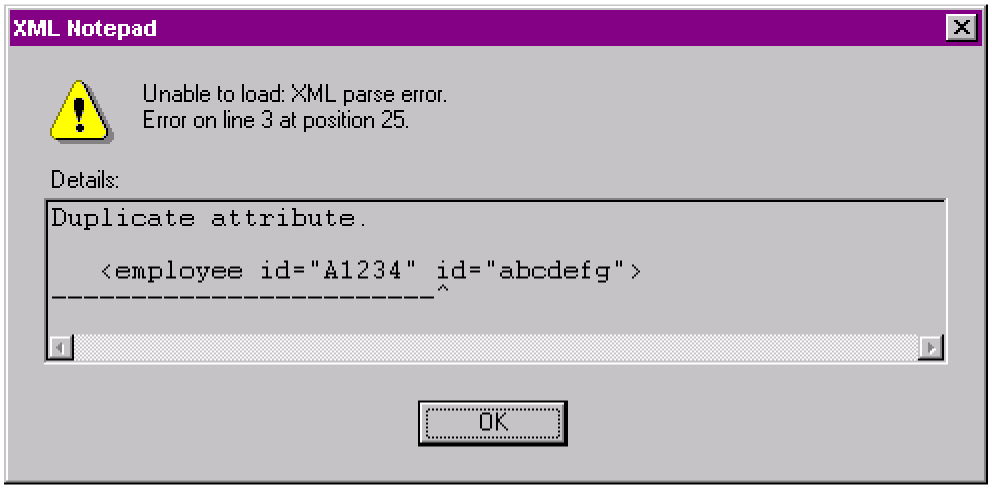
```
<employee id="A1234">
  . . . (contents omitted for brevity)
</employee>
```

In the above example, the <employee> tag has an attribute called "id". The "id" attribute contains a value of "A1234". Attributes can only be specified in the start tag of an element.

Duplicate attributes are allowed in XML. For example, the following statement is allowed, although it may not make much sense:

```
<employee id="A1234" id="abcdefg">
```

Unlike SGML and HTML, which consider this an error, the XML specification states
that this should be accepted and handled by merging the two
attribute values, and that the first declaration of an attribute is the one that can be referenced.
Based on the specification, processing of an XML
document should continue in this event. However, just like with HTML, viewers may not always implement the standard. Using the version of Microsoft XML Notepad that is available at the time of this writing, an error occurs when it encounters the above declaration.



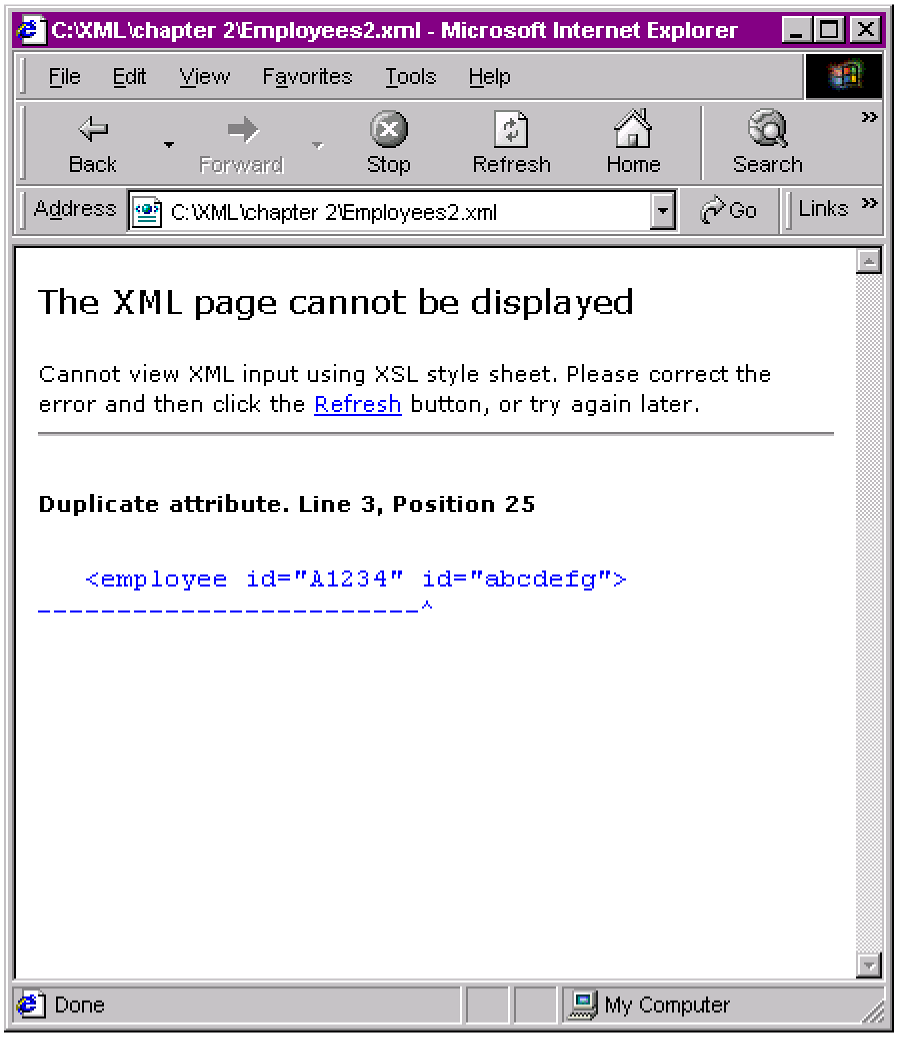
```
Unable to load: XML parse error.
Error on line 3 at position 25.
```

```
Duplicate attribute.

   <employee id="A1234" id="abcdefg">
------------------------^
```

File: Chapter 2\Employees2.xml in XML Notepad

The same is true when the document is loaded in IE5.


C:\XML\chapter 2\Employees2.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address C:\XML\chapter 2\Employees2.xml Go Links
Done
My Computer

The XML page cannot be displayed

Cannot view XML input using XSL style sheet. Please correct the error and then click the Refresh button, or try again later.

Duplicate attribute. Line 3, Position 25

```
   <employee id="A1234" id="abcdefg">
------------------------^
```

File: Chapter 2\Employees2.xml in IE5

In reality, it does not make sense to have multiple declarations of the same attribute. The point here is that the standard says that it should be allowed, even though certain XML processors may not adhere to the rules. This is not a new problem to SGML applications. This is very similar to the way that different browsers will interpret and display HTML documents slightly differently.

Entities

An *entity* has many applications in XML. The official definition of an entity is a *storage object*. The first XML entity that you have encountered so far is the XML document. The document as a whole is an entity. This entity is divided into elements.
There are other types of entities in XML as well. One of the most

powerful capabilities of XML is its ability to include *external entities*. In other words, you can reference (and thus include) another file inside of your XML document. You have probably seen this already if you have created web pages with HTML. The <IMG> tag references an external graphics file. The graphics file is not embedded inside of the HTML document, but it references it and the browser requests it from the server.

```
<IMG SRC=”./images/somepicture.jpg”>
```

We can do the same thing in XML.

```
<picture.1 source=”./images/somepicture.jpg”/>
```

In this example, the graphics file is considered an *unparsed entity* as it is not parsed by the XML processor.

You can also include other XML documents. The result is that the XML document is included in the document that referenced it and they are presented by the XML processor as one. This can be a very powerful feature, but beware its misuse. A good example is requesting a document that references another document that references another document, etc. You can easily see how things can get out of hand!

Tip: Through the use of external references in XML, you can request a small document and get back a lot more than you intended if it has lots of external entity references

Later, we are going to discuss *internal entities*, but for the remainder if this discussion, it is more important that you understand the external entity and that it is defined as a storage object.