Introduction to the eXtensible Markup Language

To understand the way in which a computer processes electronic texts, it is essential to realise firstly that the words and the characters in the texts are fundamentally meaningless to the machine. Similar to the way in which the meaning of a Chinese text eludes a person who cannot read Chinese characters, the computer essentially sees an electronic text as a seemingly arbitrary sequence of characters. For (English-speaking) human beings, the word "Shakespeare" may conjure up images or works by a well-known poet and playwright, but for the computer, this word is nothing more than eleven collocated characters. Subsequently, the computer can certainly not be expected to be capable of performing more advanced or more intelligent tasks such as finding book titles or geographic names in this text. Before this can be possible, the machine would first need to be told precisely where such information is to be found within the text.

One of the major techniques that have been developed to help the computer to overcome its basic incomprehension of human text is the eXtensible Markup Language (XML). The technique was developed by the World Wide Web consortium (W3C), and is currently a vital standard for data exchange on the web, both within the commercial and the non-commercial sector. As is also expressed by the final two letters of the name of the standard, XML is used to 'mark up' specific aspects of a textual document. In general, it can be said that mark up is used to describe certain aspects of a text explicitly. Marking up a text entails two things: (1) selecting or situating a certain logical or structural component within the text and (2) giving information about this selected text. If the communication of such information is standardised, to such an extent that there is a fixed vocabulary and an agreed way to make use of this vocabulary, we may speak of a mark up language.

XML Elements

XML is used to describe specific aspects of a text explicitly. To do this, XML makes use of terms which are known as elements. They could be terms such as "personalName" or "bookTitle". In addition, 'marking up' a piece a text requires that we insert this descriptive term in angular brackets, both before and after this text fragment. These additions are referred to as XML tags. The closing tag differs from the opening tag, in that there is a forward slash in front of the element name. If we want to say in XML that the words "Camera Obscura" together form a title, we can use the following syntax:

```
<title>Camera Obscura</title>
```

In this example, we may speak of the XML element `<title>`. An element always consists of two parts: an opening and a closing tag which classifies or identifies the text that they surround. Adding such tags to a text is also referred to as 'encoding' a text. By adding such elements, all kinds of aspects can be made explicit. If all the titles that occur in the text are encoded with the `<title>` element, we can instruct the computer to retrieve all the titles at a later stage. The use of such XML elements thus provides a way to communicate our knowledge of the words in the text to the machine. It allows us to describe aspects of the texts that are not provided literally by the words alone.

XML elements can contain not only raw texts, but also other elements. In other

words, elements can be nested within other elements. If a title consists of a main title and a subtitle, for instance, this may be encoded as follows:

```
<title>
<mainTitle>The Importance of being Earnest</mainTitle>:
<subTitle>a Trivial Comedy<subTitle>
</title>
```

The main title and the subtitle can be distinguished on the basis of the encoding, and if the computer is prompted to provide the full title, this is evidently also possible.

Elements may sometimes have no contents. This may sound puzzling, since it had been explained earlier that XML elements describe certain characteristics of a text. Nevertheless, elements can also be used to describe a certain position within a text. Such elements are referred to as 'milestone' elements. Outside the world of computing, milestones are markers that can be found along a path to indicate the geographic position of a traveller. In a sense, milestone tags have the same function in XML documents. They may be used, for instance, to record the occurrence of a line break in a letter.

```
In kind reply to your letter of 11 past<lb></lb>we beg
to inform you it is our intention<lb></lb>to give a
translation of Eliot's "Im-<lb></lb>pressions" etc.
```

There is also an abbreviated syntax for empty elements. If the forward slash is written at the end of element name in the opening tag, this notation will be understood as an empty element. The following example is completely identical to the previous example.

```
In kind reply to your letter of 11 past<lb/>we beg to
inform you it is our intention<lb/>to give a transla-
tion of Eliot's "Im-<lb/>pressions" etc.
```

Listing 1 is an example of a ful XML docment. The example shows a literature list that is encoded in XML.

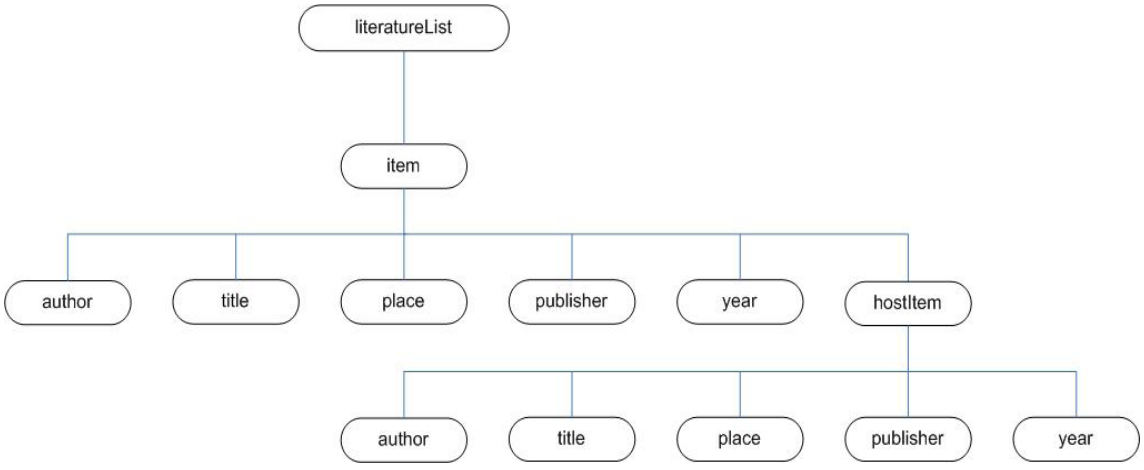
```
<?xml version="1.0" encoding="UTF-8"?>
<literatureList>
    <item>
        <author>Asa Briggs</author>
        <author>Peter Burke</author>
        <title>A Social History of the Media, from
Gutenberg to the Internet</title>
        <place>London</place>
        <publisher>Polity Press</publisher>
        <year>2005</year>
    </item>
    <item>
        <author>Robert Darnton</author>
        <title>What is the History of Books?</title>
        <hostItem>
        <title>Books and Society in History</title>
        <editor>Kenneth E. Carpenter</editor>
```

```
            <place>New York</place>
            <publisher>Bowker</publisher>
            <year>1983</year>
                <pages>3-26</pages>.
        </hostItem>
        </item>
    </literatureList>
```

On the very first line, there is a so-called XML declaration. This line is needed to tell the application that what follows needs to be interpreted as an XML document. The sample `<literatureList>` contains two `<item>`s. For each `<item>`, it is possible to record the author, title, publisher and a year and place of publication. If a publication is part of a larger unit, such as in the case of a journal article, it is also possible to use a `<hostItem>` element which refers to the journal or the book in which this text appeared. The same bibliographic aspects can be recorded for this host item as for the main item.

To clarify the structure of an XML document, it may often help to draw a tree diagram. This tree diagram gives an overview of the elements that can be used within a document. It also clarifies the hierarchical relations that can exist between these elements.

One of the rules of XML is that the names of XML elements cannot contain spaces. This may pose a problem for compound nouns, as these are mostly written with a space between the constituents. In such situations, the convention is to use the so-called camel case notation. The different words are joined, and only the first letter of a new word is capitalised. This notation is used in the elements `<literatureList>` and `<hostItem>`.



Attributes

As can be seen from listing 1, there are different kinds of items in the literature list. The first item is a monograph, and the second item is an article that has appeared in a book. They have both been listed as `<item>`s to be able to process them in a uniform manner. Nevertheless, it may also be useful in certain situations to be able to select only the monographs or only the articles from the list. The current document does not allow

us to do this. If we want to be able to distinguish different types of item elements, we could use XML attributes. An XML element with an attribute looks as follows: `<item type="article">`. Attributes are used to further modify or qualify an element. They describe a certainadditional property of the element. In this example, the article specifies that we are dealing with a special type of item, namely, an article. The monograph can be identified as such by using `<item type="monograph">`. Attributes consist of a property name and a value. To create an attribute, leave a space after the element name, give the property name followed by an equals-sign and the value for this property in double quotes.

Other forms of mark-up

Next to XML elements, there are a number of other forms of mark-up. This introduction will only discuss entities and comments. Entities, firstly, are basically codes that represent other characters. An entity reference always begins with the ampersand ("&") and ends with a semi-colon (";"). The code that is included in between these two characters will be replaced by a different literal value when the XML document is presented on a screen. Entities need to be used if characters such as the less-than sign or the greater-than sign are part of the element content, since the XML processor would otherwise consider these characters part of the mark up. The standard entities < and > can be used to represent the less-than and the greater-than signs respectively.

```
<p>The sentence is given in the &lt;p&gt; element.</p>

<formula>x &gt; y</formula>
```

Entities references also need to be used for character encoding. To secure the portability and the interoperability of documents, it is best to ensure that XML documents contain ASCI characters only. When special characters need to be used, i.e. characters that are not part of the ASCI character table, these need to be represented by a code value from the Unicode Character Code Charts. Unicode is a universally acknowledged encoding system which provides a unique number for each character. It is maintained by the Unicode Consortium. Code values can be found in the or through the Character Name Index on the Unicode website. Unicode values must be preceded by a hash ("#"). If the hexadecimal notation is used to represent the Unicode value, an "x" must also be used.

There is also a special form of mark up that can be used to insert comments into the XML document. Although an XML document is primarily intended to be processed by computers, it may sometimes be useful to add certain messages for human beings who attempt to read the XML file. All text that is included between the special tags "<!--" and "-->" will be ignored by XML processors. Here is an example of a comment which can be used to ease the human interpretation of an XML document.

```
<!-- The next section contains the transcription -->
```

Well-formedness

XML can be used to encode a wide variety of document types. They can be used in traditional text documents, such as poems and letters. Listing 1 was an example of a document which primarily captures data. In this situation, the document functions similarly

to a table in a database. Regardless of the document type, however, XML documents should always comply with a number of rules. The most important rules are given below:

- Each XML opening tag must have a matching closing tag

```
<p>This is correct</p>
<p>this is not correct, as no closing tag is used.
```

- XML elements must be nested properly

```
<!-- The next line is correct -->
<p>Nicolaas Beets was the author of <title>Camera Ob-
scura</title></p>

<!-- The next line contains an incorrect nesting -->
<p>Nicolaas Beets was the author of <title>Camera Ob-
scura</p></title>
```

- An XML document can only have a single root element

The root element is the element that contains all the other elements. In a tree diagram representing the XML document, the document element is always shown at the very top. The root of the document that is given as listing 1 is <literatureList>. This document root contains all the other documents and is not contained within any of the other elements itself.

- The names of XML elements are case sensitive

In the following example, there is a mistake, since the <item> opening tag is not closed correctly by a matching closing tag. The </ITEM> tag is considered another element.

```
<item><title>What is the History of Books?</
title><ITEM>
```

- Attribute values must be provided in quotation mark

If the value of an attribute is not provided in quotaion marks, this will be flagged as incorrect.

```
<!-- The folowing line contains an error -->
<item type=monograph><title>Books in the Digital Age</
title><item>
```

- A particular attribute name can only be used once within the same opening tag

The next line will not be accepted by XML parsers.

```
<item type="PDF" type="article"><title>What is the His-
tory of Books?</title></item>
```

If the fact that the text is available as a PDF file needs to be recorded in a different attribute.

If an XML document meets all these requirements, we can say that it is well-formed.

Validity

To summarise, XML is a technique which allows us to insert explicit descriptive terms into a text document. But how do we know which terms we can use to encode a text? And how do we know which attributes can be used? The answer is that the element names and the attribute names that are permitted in a document are normally all listed in another document that is attached to the XML document. This can be either a Document Type Definition (DTD) or an XML Schema. This text shall only discuss DTDs. The structure of an XML document can be visualised on the basis of a tree diagram. This diagram contains all the elements that are allowed and it also indicates the position on which an element may occur. In figure 1, it can be seen that the element `<hostItem>` occurs within `<item>`. A DTD can be seen as a document which expresses this same information in such a way that it can be understood by a computer. In a DTD, the notion that a `<literatureList>` can contain an `<item>` is explained as follows:

```
<!ELEMENT literatureList ( item)+>
```

The plus sign after the parenthesised item element indicates that the item element is required and repeatable. In other words, it must occur at least once within `<literatureList>`. The DTD thus gives more specific information about the structure than the tree diagram, which does not indicate which elements are mandatory and if elements are repeatable. The DTD syntax allows two other occurrence indicators, namely the question mark and the asterisk. Their meaning is provided in the table below.

Component	Description
?	optional occurrence indicator (may or may not occur: 0 or 1)
+	required and repeatable (must occur at least once: 1 or many)
*	optional and repeatable (may occur any number of times: 0, 1 or many)

The tree diagram also shows that an item element can contain the following elements: author, title, place, publisher, year and hostItem. In the DTD, this can be expressed as follows:

```
<!ELEMENT item ( author* | title | place* | publisher*
| year* | hostItem* ) *>
```

DTDs can also provide information on the attributes that may be used with certain elements. The following line specifies that the the item element can take a type attribute

with the value “article” or “monograph”.

```
<!ATTLIST item type ( article | monograph ) >
```

Listing 2 gives the full DTD for the XML document that was given as listing 1.

```
<!ELEMENT literatureList ( item)+>
<!ELEMENT item ( author | title | place | publisher |
year | hostItem )*>
<!ELEMENT author (#PCDATA)>
<!ELEMENT editor (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT pages (#PCDATA)>
<!ATTLIST item type ( article | monograph ) "monograph"
>
<!ELEMENT hostItem (#PCDATA | title | editor | place |
publisher | year | pages)*>
<!ELEMENT year (#PCDATA)>
<!ELEMENT place (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
```

This DTD can be saved as a file named “literatureList.dtd”, for instance. This DTD can then be connected to the XML document, by adding a link directly after the XML declaration. The reference to the DTD may look as follows:

```
<!DOCTYPE literatureList SYSTEM "literatureList.dtd">
```

This line is called a document type declaration. Once an XML document is linked to a DTD, an XML application can compare the structure of the document that follows to the rules as laid down in the DTD. If the document does not violate these rules, the document is considered valid. Well-formedness is only one aspect of validity. There are two requirements: (1) it must be well-formed, and (2) the document must comply with the rules as laid out in the DTD. If a document is valid, this implies that it is well-formed. The statement is not necessarily true vice versa. A well-formed XML document may still be invalid.

The ‘x’ in the abbreviation XML refers to the ‘eXtensibility’ of the standard. Users of XML are free to define their own DTDs or XML schemas. The ‘eXtensible’ mark up language is strictly speaking not a markup language itself. The standard XML should be seen as a set of rules that users can aply to make actual mark up languages. Since XML can be used to generate new languages while it is not a language itself, it is also referred to as a ‘metalanguage’. The standard does not prescribe any element or attribute names, but it enables individuals and communities to create their own DTDs and Schemas, and thus to use the terms that they consider most useful or most attractive. Nevertheless, XML is emerging more and more as a technique for the exchange of information. Certain communities have taken efforts to define standard vocabularies that can be used to descibe and encode specific document types. If individual users of XML adhere to the rules as defined in such standard DTDs, this means that their data can be disseminated effectively within this larger community. One of the most important standards in the humanities is the Text Encoding Initiative (TEI).