Comp 336/436 - Markup Languages

Fall Semester 2018 - Week 6

Dr Nick Hayward

XML - a few more demos and examples

- eHinman Collator
- regularise - soundex

XML - well-formed - intro

- HTML may often include errors and issues in the markup
  - *a browser helps resolve issues and errors in the markup*
- XML has strict rules
  - *XML document must be correctly structured*
  - *correct structure to enable machine parsing*
- XML specification prohibits XML parsers
  - *parsers may not try to fix and understand malformed documents*
- a parser may simply report the error

XML - well-formed - W3C definition

- W3C definition of a well-formed XML document
  - *unique opening and closing tag that enclose the whole document*
  - *all elements include a closing tag or correctly defined empty elements*
  - *all tags and attributes names adhere to case-sensitive rule*
    - e.g. the tag `<hello>` cannot be closed with `</hello>` &c.
  - *all elements properly nested*
    - i.e., an opening and closing tag
    - tags may not overlap &c.
  - *all attribute values always quoted correctly*
  - ...

n.b. these are the most important constraints - not an exhaustive list...

XML - validation

- *Well Formed* XML = correct syntax, e.g.
  - *root element*
  - *elements must have closing tags*
  - *tags are case sensitive*
  - *elements must be properly nested*
  - *attribute values must be quoted*
- *Valid* XML is *Well Formed* XML that conforms,
  - *e.g. to a DTD (document type definition)*

XML - validation - DTD

- DTD - Document Type Definition
- XML allows a developer to organise their own tags &c.
- XML documents often need to follow a specific grammar
- helps with sharing and reuse
  - *e.g. within a broader community and domain*
- the defined grammar will then become a known tool
  - *describes structure and context of all necessary topic specific XML documents*
- purpose of a Document Type Definition (DTD)
  - *to enable and permit such shared grammars*
- a DTD provides a framework for validating XML documents
  - *by defining legal building blocks of XML documents*
- a DTD outlines permitted elements in an XML document
  - *and any available attributes and sub-elements*
- DTD can be part of the XML document
  - *or it can be referred to by the XML document*
- external DTD is a simple text file with `.dtd` extension

XML - validation - DTD - element declarations

- element declarations describe a permitted set of elements within a document
  - *i.e. nature of declared elements, character data, &c.*
- elements in XML documents may
  - *enclose other elements*
  - *be empty*
  - *contain content*
  - *or be mixed (i.e. containing content and other elements...)*
- in a DTD, possible declarations for elements are as follows,
  - `<!ELEMENT element-name (child1,child2,...)>`
    - for an element containing other elements
  - `<!ELEMENT element-name EMPTY>`
    - for an empty element
  - `<!ELEMENT element-name (#PCDATA)>`
    - for an element containing content
  - `<!ELEMENT element-name (#PCDATA|child1|otherchild1)*>`
    - for a mixed element
  - `<!ELEMENT element-name ANY>`
    - for defining an element
    - no further specific detail provided
  - *a DTD must specify how the element may appear*
  - *i.e. in a given order, if they can be repeated...*

XML - validation - DTD - special characters

DTD special characters for element repetition and order.

character	definition
'	separate sequence items, indicate sequential order or items
|	choice separator, indicates selection of one item from list
( )	group elements
+	required occurrence with repetition
*	optional occurrence with repetition
?	optional occurrence

XML - validation - DTD - special characters

- if we create XML documents describing books
  - *with a title*
  - *multiple authors*
  - *different chapters*
- definition of the element book, e.g.

```
<!element book (title,author+,chapter+)>
```

- element book can contain
  - *only other elements*
  - *not content directly*
  - *a title (title)*
  - *one or more authors (author+)*
  - *successively one or more chapters (chapter+)*

XML - validation - DTD - ATTLIST declarations

- **ATTLIST** declaration
  - *XML element attributes are declared in the DTD*
- attribute-list declarations name
  - *a permitted set of attributes for each declared element*
  - *type of each attribute value,*
  - *explicit set of valid value(s) is not always necessary...*
- syntax for an attribute declaration, e.g.

```
<!ATTLIST elementName attributeName Type defaultValue>
```

XML - validation - DTD - ATTLIST declarations

value	definition
CDATA	character data value
ID	value is unique ID
IDREF	value is another element's ID
IDREFS	value is list of other IDs
NMTOKEN	value is valid XML name
NMTOKENS	value is list of valid XML names
ENTITY	value is entity
ENTITIES	value is list of entities
NOTATION	value is name of a notation
xml:	predefined value

XML - validation - DTD - add to XML

- a document type declaration
  - *added after the XML declaration*
  - *a mechanism for naming the document type for compliance*
  - *and for including its definition...*
- valid XML documents must declare the document type
  - *editors, browsers &c. can read* ***DTD***
  - *helps define the template structure...*
- a document type declaration names the document type
  - *references the root element of the document*
- it can reference an external DTD
  - *the external DTD subset*
- include the DTD internally
  - *in the internal DTD subset*
- or use both...
- document type declarations use the following form, e.g.

```
<!DOCTYPE NAME SYSTEM "file">
```

- e.g. reference an external DTD

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

XML - validation - DTD - issues

- DTD issues for XML include:
  - *DTDs do not make use of XML syntax*
  - *DTDs have no constraints on character data*
  - *e.g. if character data is allowed...any character data allowed*
  - *poor support for schema evolution, extension, or inheritance of declarations*
  - *DTDs provide simplistic attribute value models*
  - *DTDs provide a simple ID attribute mechanism*
  - *DTDs allow only default values for attributes, not for elements*

XML - validation - schema - intro

- DTDs inherited by XML from its predecessor SGML
- DTDs helped ease the transition from SGML to XML
- XML Schema became a W3C recommendation in 2004
    - *provides a rich and flexible mechanism for defining XML vocabularies*
- **Schemas** are written using XML syntax
- **Schemas** are referenced as external documents
- an XSL Schema (**XSD**) is itself an XML document

XML - validation - schema - a few benefits

- XML Schemas are now the successors to DTDs.
- a number of benefits to schemas, e.g.
  - *written in XML*
  - *support namespaces and data types*
  - *richer and more powerful than DTDs*
  - *extensible to future additions*

XML - validation - schema

- an XML schema defines elements and attributes allowed in a document, e.g.
  - *child elements*
  - *the order of child elements*
  - *the number of child elements*
  - *whether an element is empty or can include text*
  - *data types for elements and attributes*
  - *default and fixed values for elements and attributes*

XML - validation - schema - basics

- XML Schema starts with the document declaration
  - *continues by opening the root element* `<schema>`
  - *and by defining the specific namespace*
- within root element all specifications are defined
- schema ends with a closed root element `</schema>`
  - *as a well-formed XML document*
- XSD file - simple text file with `.xsd`
  - *basic outline, e.g.*

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  ...
</xsd:schema>
```

XML - validation - schema - body

- body of the schema contains element declarations
- four main schema elements
  - *`xsd:element` declares an element and assigns it a type*
  - *`xsd:attribute` declares an attribute and assigns it a type*
  - *`xsd:complexType` defines a new complex type*
  - *`xsd:simpleType` defines a new simple type*
- XML Schema provides a set of 19 primitive data types
  - *e.g. boolean, string, decimal, date, time...*
  - *use directly in an element or attribute definition, e.g.*

```
<xsd:element name="name" type="xsd:string" />
<xsd:attribute name="age" type="xsd:integer" />
```

XML - validation - schema - declarations

- `xsd:complexType` and `xsd:simpleType` are used to define new types
- simple declarations define elements
  - *do not have any children or attributes*
  - *may only contain text*
- complex declarations describe elements
  - *may have children and attributes*
  - *may contain text*
- declarations are not actually types
  - *they create an association between a name and defined constraints*
  - *constraints may govern appearance of a name*
  - *applied in documents governed by associated schema*

XML - validation - schema - example book element

- XSD example defining an element
  - *book of a user-defined complex type bookType*
- sub-elements in *bookType* definition
  - *a simple element of type string or a number (gYear)*
  - *gYear = one calendar year, e.g. 1997*
- element `<xsd:sequence>` identifies a sequence of elements

```
<xsd:element name="book" type="bookType"/>
<xsd:complexType name="bookType">
<xsd:sequence>
<xsd:element name="title" type="xsd:string" minOccurs="1" maxOccurs="1"/>
<xsd:element name="author" type="xsd:string" />
<xsd:element name="year" type="xsd:gYear" />
</xsd:sequence>
</xsd:complexType>
```

- XSD uses attributes `minOccurs` and `maxOccurs` to define cardinalities
- this example defines the element *title*
  - *may occur only one time within the element book*

XML - validation - schema - types

- XSD documents allow us to derive new simple types from existing types
    - *by using the `xsd:simpleType` element*
    - *basically defining a subtype*
- different type of elements can be used to define the subtype
- `xsd:restriction` child element
    - *derives a type - by restricting legal values of base type*
- `xsd:list` child element
    - *derives a type - a white space separated list of base type instances*
- `xsd:union` child element
    - *derives a type - by combining legal values from multiple base types*

XML - validation - schema - types example

```
<xsd:simpleType name="vehicle">
<xsd:restriction base="xsd:string">
<xsd:enumeration value="car"/>
<xsd:enumeration value="motorbike"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:element name="racer" type="vehicle" />
```

- a new simple type, `vehicle`
  - *defined as enumeration of possible string values*
  - *e.g.* `car` *and* `motorbike`
- and element `racer` defined of type `vehicle`

XML - transforming & rendering - intro

- XSL is a family of recommendations
  - *used for defining XML document transformation and presentation*
- XSL engine uses these stylesheets to transform XML documents into other documents
  - *formats output according to specific formatting templates*
- XSL family consists of three main sub-languages
  - *XSLT (XSL Transformations)*
    - XML-based language for transforming XML documents into other XML documents
    - e.g. XML, XHTML...
  - *XPath (XML Path Language)*
    - expression language used by XSLT
    - access or refer to parts of an XML document
  - *XSL-FO (XSL Formatting Objects)*
    - XML vocabulary for specifying formatting semantics
    - used for print publications, e.g. XML to PDF...
- in XSL
  - *input document is often called the source tree*
  - *output document the result tree*

XML - transforming & rendering - XSLT & XPath

- XSLT is a powerful language for transforming XML documents, e.g.
  - *a HTML document*
  - *another XML document,*
  - *a Portable Document Format (PDF) file*
  - *a Scalable Vector Graphics (SVG) file*
  - *a flat text file*
  - *...*
- XSLT stylesheet defines the rules for transforming an XML document
  - *chosen XSLT processor does the work and produces the output*
- XSLT relies on a technology called XPath
- XPath helps XSLT to identify nodes in XML documents
  - *e.g. elements, attributes, and other objects...*
- XPath also provides various functions for performing calculations...

XML - transforming & rendering - XSLT
Browser Support

- Mozilla Firefox
  - *Firefox supports XML, XSLT, and XPath from version 3*
- Internet Explorer
  - *Internet Explorer supports XML, XSLT, and XPath from version 6*
  - *Internet Explorer 5 is NOT compatible with the official W3C XSL Recommendation*
- Google Chrome
  - *Chrome supports XML, XSLT, and XPath from version 1*
- Opera
- Opera supports XML, XSLT, and XPath from version 9. Opera 8 supports only XML + CSS
- Apple Safari
  - *Safari supports XML and XSLT from version 3*

XML - transforming & rendering - XSLT Process Outline

- select the XML document you want to transform into XHTML
- create an XSL style sheet with a transformation template
- link the XSL style sheet to the XML document
- XSLT compliant browser will transform XML into XHTML

e.g. Agatha Christie - XML - part 1

- XSLT outputs
  - *document heading*
  - *book title*
  - *book author*

XML - XSLT working example - Agatha Christie

XML file - part 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="christie-full.xsl"?>
<catalogue>
  <book>
    <title>Evil Under the Sun</title>
    <author>Agatha Christie</author>
    <country>UK</country>
    <publisher>Collins Crime Club</publisher>
    <price>7</price>
    <year>1941</year>
  </book>
</catalogue>
```

XML - XSLT tests - initial XML

Exercise - part 1

- create an initial XML document for your own preferred *catalogue* of items
  - *e.g. music albums, toys, books, art, household items, sports, &c.*
- add at least two examples
  - *e.g. two or more books, albums, artworks &c.*
- add *title*, *author/creator/editor* &c. for each item
- add other details such as
  - *date, value, location, owner &c.*

10 minutes...

XML - transforming & rendering - XSLT basic usage

- XSLT transformation file is an XML document
- it follows the same syntax of any other XML
- using XSLT language, we have to define the appropriate namespace
- example file,

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
[… other directives …]
</xsl:stylesheet>
```

- in the above example,
  - *first line is the XML declaration*
  - *second defines the root element* `<xsl:stylesheet>` *and attributes*
- stylesheet ends with the root element closing tag
  - *as a well-formed XML document...*

XML - transforming & rendering - XSLT basic usage

- a few other necessary directives for XSLT
- need to intercept the root element and apply a stylesheet template
  - *taking the associated content*
  - *rewriting it as content of the HTML tag* `<h1>`
  - e.g.

```
<xsl:template match = "/" >
<html><body>
<h1><xsl:value-of select="book" /></h1>
</body></html>
</xsl:template>
```

- define a template and apply it to matched XML

XML - transforming & rendering - XPath basic usage

- XPath language is used to locate the desired element
  - *e.g. the expression* `"/"` *identifies the root element*
- once root element has been selected
  - *extract content using another XSLT directive,* `<xsl:value-of…>`
  - *attribute* `select` *contains another XPath expression*
  - *extract the content inside the element* `book`
- wrap extracted content with HTML elements
  - *create basic HTML page*

Demos

- Various
  - *eHinman Collator*
  - *regularise - soundex*
- XML Agatha Christie - XML - part 1

References

- Oxygen XSLT Processors
- W3C - XML well formed
- W3C - XSLT 1.0