Comp 336/436 - Markup Languages

Fall Semester 2018 - Week 4

Dr Nick Hayward

XML - recap

- first version of XML became a W3C Recommendation in 1998
- a useful format for data storage and exchange
  - *config files, data storage, data exchange...*
- XML is a simple, very flexible text format
  - *used for the description of marked-up electronic content*
- XML is classified as extensible because
  - *allows user to define markup elements*
  - *e.g. elements for a document, project, domain...*
- XML is a meta-language
  - *a means of formally describing a language*
  - *e.g. a markup language*
- XML useful for
  - *sharing data*
  - *abstraction of data and presentation*
  - *document reuse*
  - ...
- XML provides a basic syntax

XML - recap - origins

- XML emerged as a way to overcome the shortcomings of its two predecessors
  - *Standard Generalised Markup Language (SGML)*
  - *HyperText Markup Language (HTML)*
- HTML is too limited, while SGML is too complex
- XML is software and hardware independent
  - *light and simple tool for carrying information*
- XML helps science, industries, companies
  - *specify how to store specific data*
  - *store data in a machine compatible form*
  - *allows applications to run on any platform*
  - *easily import and process this data*
  - ...

XML - recap - usage

- XML has some important characteristics
  - *extensible so it does not contain a fixed set of tags*
  - *documents must be well-formed according to a strict set of rules*
  - *may be formally validated using DTDs or XML Schemas*
- HTML documents can contain errors (and often do)
  - *mdern browsers will still render the pages as well as possible*
- XML focuses on the meaning of data, not its presentation
- XML is not a replacement for HTML
- XML is used to transport data
- HTML is used to format and display the data for the Web &c.

XML - syntax

- XML is a formal specification for markup languages
  - *formal language specifications have an associated syntax*
- an XML document consists of the following:
  - *a prolog*
    - includes XML declaration
    - optional reference to external structuring documents
  - *the body*
    - consisting of a number of elements which may also contain attributes
- *prolog* is the declaration
  - *informs the machine that the document &c. is XML*
  - *includes any relevant additional information such as the encoding...*
- other components may be inserted in the *prolog*
  - *associated schemas (either DTDs or XML Schema)*
  - *or attached stylesheets (in CSS or XSL)*
- in the *body*
  - *elements are organized in a hierarchical structure (a tree)*
  - *one root element (Document element)*
  - *all other elements within the root*

XML - example structure

xml

```
<?xml version="1.0" standalone='yes'?>
<library>
  <book category="non-fiction">
    <title>The Discovery of the Tomb of Tutankhamen</title>
    <author>Howard Carter</author>
    <year>1923</year>
  </book>
</library>
```

tree

```
 /library
    |-book
       |---title
       |---author
       |---year
```

XML - structure - elements

```
<library>
  <book category="non-fiction">
    <title>The Discovery of the Tomb of Tutankhamen</title>
    <author>Howard Carter</author>
    <year>1923</year>
  </book>
</library>
```

- XML element
  - *everything from start to end tag*
  - *inclusive of the tags*
- element can contain
  - *other elements*
  - *text*
  - *attributes*
  - *or a mix of all of the above...*
- example listing - a bit of fun

XML - structure - elements - naming rules

There are some naming rules for use with XML elements, e.g.

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

XML - structure - elements - best practices

There are also some best practices for use with XML elements, e.g.

- make element names descriptive and easy to read
- try to avoid hyphenated words
  - *some software may perceive this as a subtraction*
- try to avoid using a `.` period
  - *some software may perceive this as a defined property...*
  - *e.g. property of an object...*
- avoid colons in the element name
  - *reserved for the namespace in an XML document*

XML - structure - elements - extensible

- XML elements are also extensible
- extend an element and it should not break an application
- XML is extensible
  - *not a fixed format like HTML*
- XML is a metalanguage
  - *a language for describing other languages*
- describe your own languages, e.g.
  - *RDF, SVG...*
  - *list of XML markup languages*

XML - structure - attributes

```
<book type="print">Hannibal's Footsteps</book>
```

- additional information about the element
- attribute values must be quoted
- double or single quotes

```
&quot;
```

XML - structure - attribute usage

- some of the problems with using attributes include,
  - *attributes cannot contain multiple values (elements can)*
  - *attributes cannot contain tree structures (elements can)*
  - *attributes are not easily expandable (for future changes)*
- metadata is a common use of attributes for the element content
  - *e.g. an id or reference number*

XML - structure - attribute vs element

- different discussions amongst various scientific and technical communities
  - *when and why to encode information into attributes or as content in elements*
- there is not a prescribed or specific rule for this choice
- the choice will often depend on the developer or designer
  - *domain influence as well*
  - *company practice and preferences*
  - *parser or tools for transforming XML*
  - ...
- using attributes at all is often a matter of choice and expediency
- e.g. what's the difference between the following?

```
<title><primary>The Discovery of the Tomb of Tutankhamen</primary></title>
```

or

```
<title type="primary">The Discovery of the Tomb of Tutankhamen</title>
```

XML - structure - attribute usage

- XML attributes are normally used to describe elements
  - *provide additional information about elements*
- metadata (data about data) commonly stored as attributes
- data stored as elements
- styles and patterns of usage will also develop with experience
  - *often informed by best practices in a given domain or community*

e.g.

```
<library>
  <book category="non-fiction">
    <title type="primary">The Discovery of the Tomb of Tutankhamen</title>
    <author>Howard Carter</author>
    <year>1923</year>
  </book>
</library>
```

XML - test 1

Think about how you might encode the following information:

- a car
- a sports team
- 2 Musical CDs including each song per album per CD (e.g. each album has 5 songs)

XML - test 1 - example solution

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<vehicle>
  <car id="1" type="compact" manufacturer="Ford">
    <model>Focus</model>
    <variant>Hatchback</variant>
    <doors>5</doors>
    <trim>Sport</trim>
    <colour>Black</colour>
    <year>2014</year>
    <fuel>Unleaded</fuel>
    <transmission>Manual</transmission>
    …
  </car>
  <boat id="1" type="motor" manufacturer="Princess">
    ...
  </boat>
</vehicle>
```

- demo

To the Lighthouse

(Here Mr. Carmichael, who was reading Virgil, blew out his candle.)

3

But what after all is one night? A short space, especially when the darkness dims so soon, and so soon a bird sings, a cock crows loudly, or a faint green quickens, like a turning leaf, in the hollow of the wave.

It seemed now as if, touched by human penitence and all its tool, divine goodness had parted the curtain and displayed behind it, single, distinct, **the hare erect**; the wave falling; the boat rocking, which, did we deserve them, should be

XML - test 2 - example solution

```
<?xml version="1.0" encoding="UTF-8"?>
<novel>
  <page no="198">
    <line no="1">
      <title align="centre" style="underline" colour="black">To the Lighthouse</t
    </line>
    <line no="2" type="empty"></line>
    <para no="1">
      <line no="3">(Here Mr. Carmichael, who was reading</line>
      …
    </para>
    <line no="6">
      <section align="centre" style="heading" colour="blue">3</section>
    </line>
    ...
    <line no="11"><rend style="highlight" colour="green">green</rend>...</line>
    ...
  </page>
</novel>
```

- demo

XML - processing instructions

- an XML document can also contain processing instructions
- processing instructions encode application-specific data
  - *e.g. document declaration which opens every XML document*
- each instruction is enclosed in <? and ?> delimiters
- the target identifies the application
  - *an application should ignore unrecognised processing instructions*
- processing instructions allow a developer to enter directives
  - *not part of the XML document's content*
  - *instructions are passed up to different ad-hoc applications*

XML - entities

- an XML document can use also entities
  - *similar to macros for programmers*
  - *or aliases for more complex functions*
- a single entity name can replace lots of text
  - *e.g.* `<!ENTITY editor "yvaine">`
- an entity can be recalled in the content of the document
  - *e.g.* `&editor;`
- use in an XML document, e.g.

```
<person>The editor's name is &editor;</person>
```

XML - CDATA

- we may also use CDATA elements in an XML document
- a CDATA element tells the XML parser not to interpret or parse characters
- the content may now be parsed
  - *even though it contains a character as part of an instruction...*
  - *parser expects an entity*

```
<editors><![CDATA[tristan & yvaine]]></editors>
```

XML - namespaces - intro

- in XML, element names are defined by developers
  - *different organisations may use the same tag*
  - *and then markup content with different semantics*
- XML also designed to enable interoperability and data exchange
  - *method to combine XML sources without ambiguity*
- *namespaces* designed to provide uniquely named elements and attributes
- as defined by the W3C a namespace is
  - *a collection of XML elements and attributes*
  - *identified by an Internationalised Resource Identifier (IRI)*
  - *often referred to as an XML vocabulary*

XML - namespaces - usage

- namespaces are declared as an attribute of an element
  - *using the xmlns name attribute in an element's start tag*
- not mandatory to declare namespaces only at the root element
  - *may be declared at any element in the XML document*
- namespace scope:
  - *begins at the element where it has been declared*
  - *applies to the entire content of that element*
  - *unless overridden by another namespace declaration*

```
<editor xmlns:editor="http://www.mynamespace.com" />
```

- a namespace declaration has the following syntax, e.g.
  - `xmlns:prefix="URI"`

XML - namespaces - structure

- URI is a string of characters, which identifies an Internet Resource
  - *most common URI is the Uniform Resource Locator (URL)*
  - *a less common type of URI is the Uniform Resource Name (URN)*
  - *e.g.* `urn:isbn:0030818516`
- a specific namespace identifies a collection of names
- a collection of element names
  - *e.g. standard XHTML*
- a collection of attribute names
  - *e.g. XLink*
- a namespace can collect names of properties
  - *e.g. FOAF (Friend of a Friend)*
  - *dictionary of named properties and classes*
  - *uses W3C's RDF technology*
- a namespace may describe a set of functions
  - *e.g. the XPath Data Model*

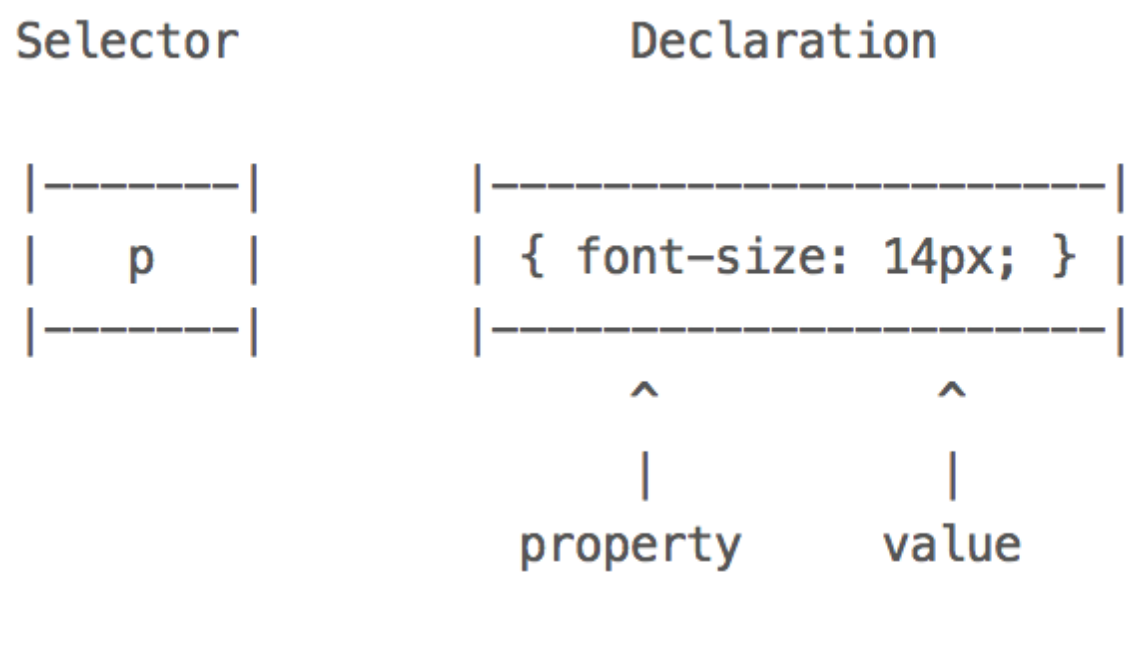
XML - rendering - css styling

- XML was created to structure content, not to display it for users &c.
- render XML with styles in a browser
  - *a CSS stylesheet can be applied similar to HTML files*
- CSS is a language used to describe the presentation of a document
  - *a document written in a markup language*
  - *e.g. SVG has its own CSS properties and values...*
- CSS document is basically constituted by a set of rules
- a ruleset consists of two parts: a selector and a declaration
  - *property and a value, e.g.*

```
selector {property1:value; property2:value;}
```

- selector is a reference to the element to be rendered
- declaration defines applied styles, effects...

Image - CSS Syntax



CSS Syntax

XML - rendering - css stylesheets

- with XML documents
  - *attach external stylesheets using `xml-stylesheet` processing instruction*
  - *add to the prolog of the XML document, e.g.*

```
<?xml-stylesheet href="style.css" type="text/css"?>
```

- there can be multiple XML stylesheet processing instructions
- possible to attach multiple stylesheets to an XML document
- possible attributes are
  - `type` *- specific type, e.g.* `text/css`
  - `medium` *- display medium type, e.g. print, screen...*
  - `title` *- specify local name, a title...*

XML - rendering - css stylesheets example

XML

```
<?xml version="1.0" standalone="yes" ?>
<library>
<title>Example with CSS</title>
<body>body in the library...</body>
</library>
```

XML & CSS stylesheet

```
<?xml version="1.0" standalone="yes" ?>
<?xml-stylesheet href="style1.css" type="text/css" ?>
<library>
<title>Example with CSS</title>
<body>body in the library...</body>
</library>
```

- XML document is still visible to the browser - i.e. data open and accessible
  - *tags rendered and styled with CSS*

XML - viewing examples

- view without styling in a web browser, editor...
  - *demo*
- render with CSS by associating a stylesheet with a given XML file
  - *demo*
- transform the document using XSLT
- access, query and manipulate XML using JavaScript
- parse XML using PHP, render HTML, manipulate with JavaScript, and style with CSS
  - *demo 1 - TEI Parser simple*
  - *demo 2 - TEI Parser full*
  - *Winchester XML*

Demos

- TEI Parser - simple
- TEI Parser - full
- XML Basic - cars
- XML Basic - literary text
- XML Basic - CD catalogue
- XML Basic - CD catalogue with CSS
- XML Sample - Winchester

Resources

- MDN - CSS
- W3C - CSS
- W3 Schools - CSS
- list of XML markup languages