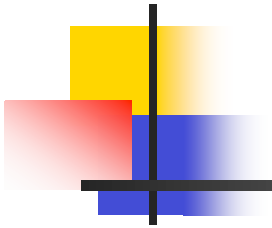
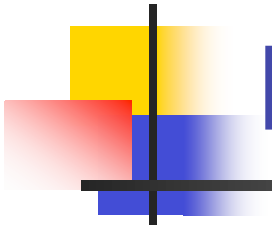


B336 Internet Systems Programming

The XML Document

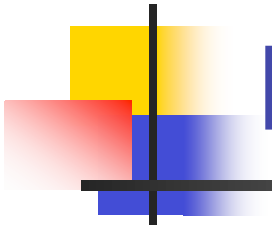
(Week 5 Lecture 2-1)



Lecture Objectives

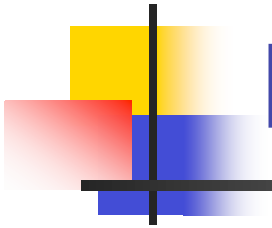
- Understand the role of the XML document format in XML technologies.

- Know the format and syntax of an XML document, as specified by W3C's XML 1.0 Recommendations.



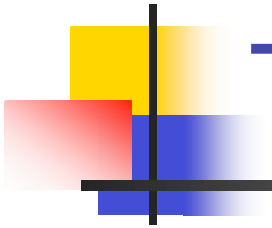
Learning Objectives

- In the scheme of what we are doing in this unit:
  - We are studying XML as an important set of Internet technologies to use as solutions in different areas.
  - The XML document format is the first and most basic step in understanding how all the different XML technologies work.



Lecture Outline

- How important is the XML 1.0 document format specifications?
- Definition of a well-formed XML document.
- Components of a well-formed XML document.

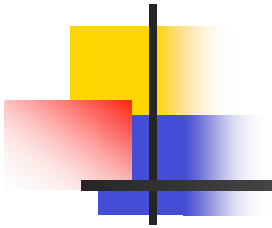


The XML Document Specification

- In the last lecture, we discussed two key factors to the success of XML as a set of technologies:

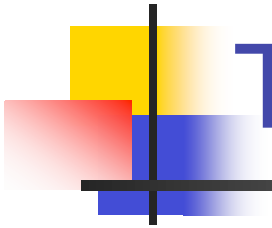
  1. Standardization
  2. Ease of creating new languages

- These two factors are built on how good the specifications of the basic XML document format is.



Reference

- The current base XML 1.0 Recommendations (2nd Edition):
  - http://www.w3.org/TR/REC-xml
  - The material in the Sybex textbook covers the details of the specifications quite extensively - you can use it as a reference.

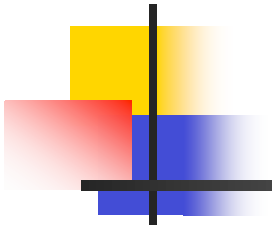


The XML Document Specification

- The W3C's XML 1.0 Recommendations specifies two sets of constraints for an XML document:

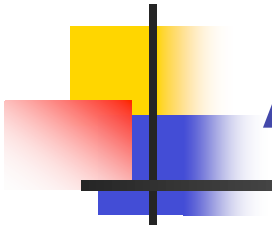
  1. A **well-formed** document: The XML document conforms to the basic syntax rules.
  2. A **valid** document: Having a DTD to to specify the allowed components (eg. tags, structure) in the XML document.

- In this lecture, we will concentrate on what makes a document "well-formed".
  - We will deal with "valid" documents in the next lecture.



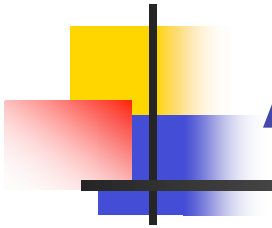
The Importance of XML Document Specification

- Having well-defined documents will make it:
  - Easy for document publishers and information content creators to create new documents.
  - Easy for software to parse and process the documents.
  - Easy for people to read and understand the documents.
- These are the core design goals of XML (see the last lecture).



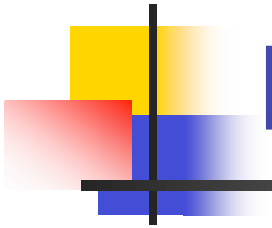
A Well-Formed XML Document

- **ALL** XML documents **MUST** be well-formed.
  - That is, they MUST conform to all syntax definitions in XML 1.0 Recommendations.
  - Unlike HTML, XML software are not allowed to "correct" errors. Non-well-formed documents will always cause unrecoverable errors in all XML software.



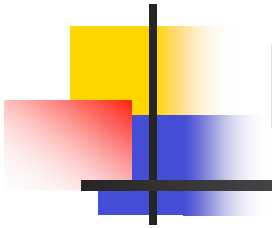
A Well-Formed XML Document

- A well-formed XML document consists of 3 parts:
  - Prolog (optional)
  - A root element
  - Miscellaneous parts (optional) - misc parts can exist within the Prolog and the root element parts as well.



Root Element

- The root element is where all the "actual" data resides.
- There can be **one and only** one root element.
- All child elements must be **properly nested**.
- The effect of these conditions means that the elements forms a **tree**.
  - The root element (obviously) is at the root of the tree.



Miscellaneous Parts

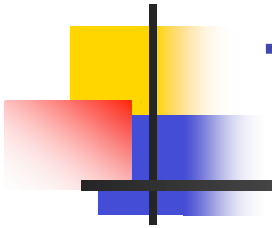
- The “Miscellaneous parts” of an XML document can consist of:
  - Comments. Eg.

    ```
    <!-- Start of the main tag -->
    ```

  - Processing Instructions. Eg.

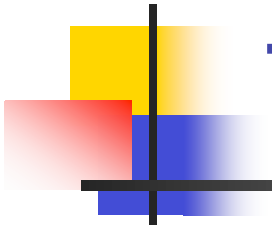
    ```
    <?xml stylesheet type="text/css" href="mycss.css"?>
    ```

  - White Spaces



The Prolog

- The prolog exists at the beginning of an XML document. It can consist of:
  - An XML declaration, followed by
  - Miscellaneous parts (as described before), followed by
  - A Document Type Declaration (DTD) - more on this in the next lecture.
- Although formally, the prolog **can** be empty, the W3C (and the specifications itself) recommends that no documents leave out the declaration part.
  - Most XML documents on the net do have an XML declaration.

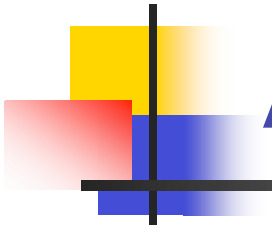


The XML Declaration

- Some example declarations:

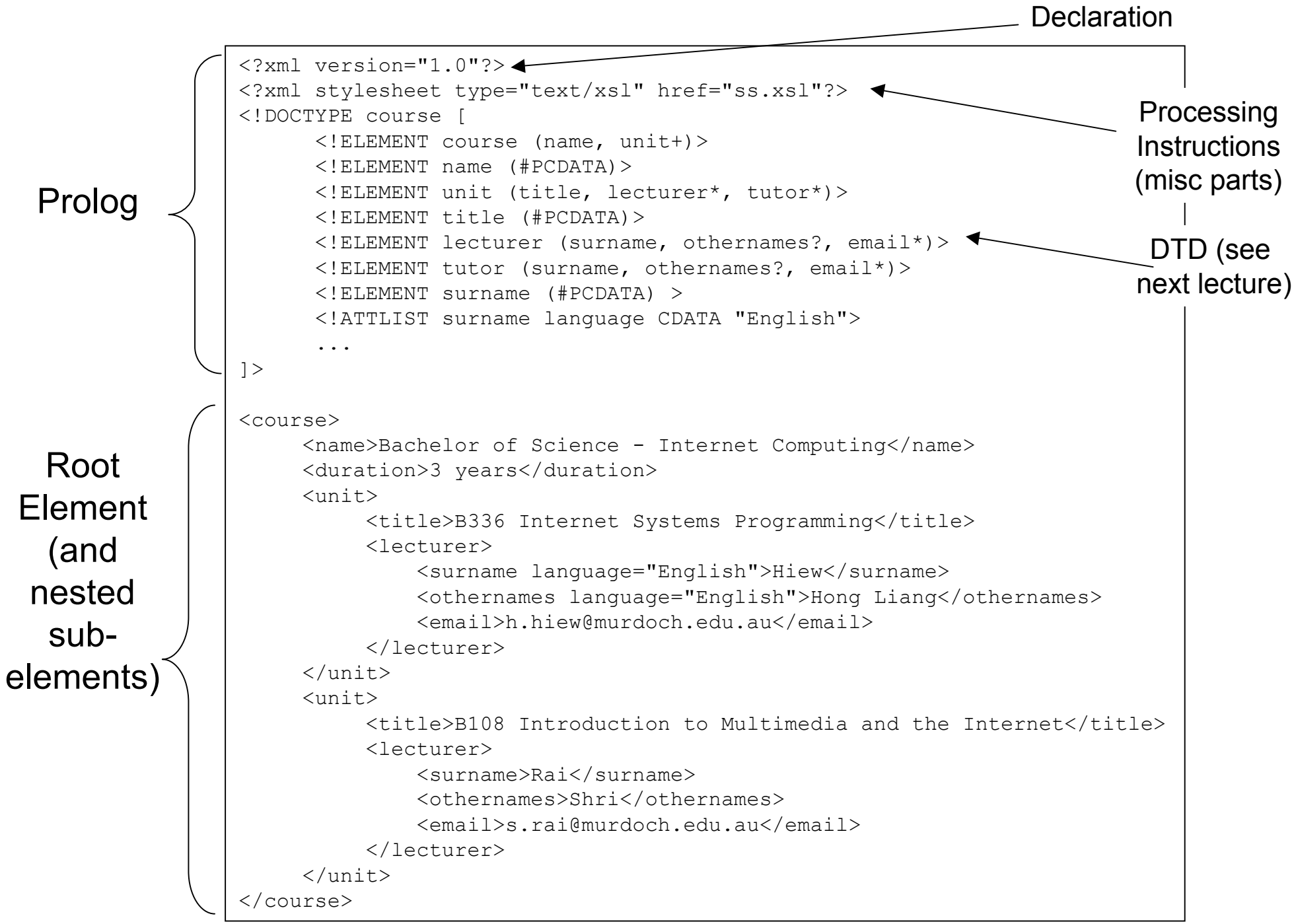
```
<?xml version="1.0"?>

<?xml version="1.0" standalone="yes"?>

<?xml version="1.0" standalone="yes"
 encoding="UTF-8"?>
```

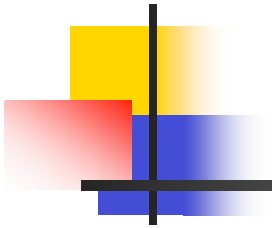
- The declaration must have at least the “xml” keyword and the “version” attribute.



A summary so far...

- So a well-formed XML document consists of (in order):
  - (Optional) Prolog with
    - a <?xml …?> declaration
    - Miscellaenous parts
    - Document Type Definition
  - A required root element
    - where all other elements exists within
  - (Optional) Miscellaneous parts


Declaration
Processing Instructions (misc parts)
DTD (see next lecture)
Prolog
Root Element (and nested sub-elements)
<?xml version="1.0"?>
<?xml stylesheet type="text/xsl" href="ss.xsl"?>
<!DOCTYPE course [
<!ELEMENT course (name, unit+)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT unit (title, lecturer*, tutor*)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT lecturer (surname, othernames?, email*)>
<!ELEMENT tutor (surname, othernames?, email*)>
<!ELEMENT surname (#PCDATA) >
<!ATTLIST surname language CDATA "English">
...
]>
<course>
<name>Bachelor of Science - Internet Computing</name>
<duration>3 years</duration>
<unit>
<title>B336 Internet Systems Programming</title>
<lecturer>
<surname language="English">Hiew</surname>
<othernames language="English">Hong Liang</othernames>
<email>h.hiew@murdoch.edu.au</email>
</lecturer>
</unit>
<unit>
<title>B108 Introduction to Multimedia and the Internet</title>
<lecturer>
<surname>Rai</surname>
<othernames>Shri</othernames>
<email>s.rai@murdoch.edu.au</email>
</lecturer>
</unit>
</course>



Elements in XML

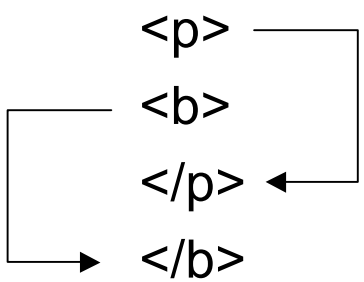
- The content of an XML document (the information the author wants to convey) are broken up into units called *elements*.

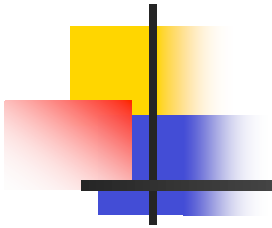
- Different types elements are given different names, and tagged with a **start-tag** and **end-tag** with that name.
  - Eg. In the element `<StudentID>12345678</StudentID>`, the information “12345678” of type “StudentID” is marked-up with the tag `<StudentID>`.

Elements in XML

- **All** basic information text must be tagged in an XML file.

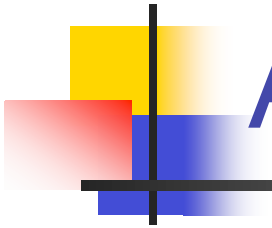
- All elements must exist **properly** nested within the root element.
  - Can't have overlaps like this





Empty Elements

- Some elements do not have closing tags. These are called “empty” elements.
  - You can see the concept of empty elements in HTML, from tags like <hr>, <br>, <img>, etc.
- ... BUT in XML, tags of empty elements must end with a “/”.
  - Eg, in XHTML, the new XML compliant version of HTML, we have <hr /> and <br /> tags.

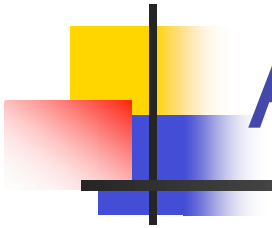


Attributes of Elements

- Elements may also contain attributes.
- The attribute **names** and **values** are defined in the start-tag. Eg.

```
<Student ID="12345678" status="enrolled"
      workrate="dead lazy" />
```

- All attribute values must be enclosed in quotes - unlike HTML.



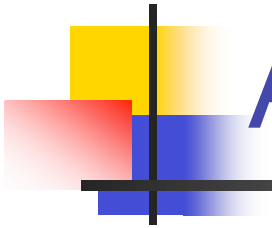
Attributes vs Child Elements

- In any element, you can define information about the element by using attributes, or using child elements.

- Eg. the following contains the same information:

```
<lecturer>
   <surname>Hiew</surname>
   <othernames>Hong Liang</othernames>
   <email>h.hiew@murdoch.edu.au</email>
</lecturer>
```

```
<lecturer surname="Hiew"
        othernames="Hong Liang"
        email="h.hiew@murdoch.edu.au" />
```



Attributes vs Child Elements

- Sometimes there are obvious technical reasons for using one versus another.
  - Eg. Wanting to use a default value - easier with an attribute than an element.
- In many cases the decision will be a judgement call.
- There should at least be differences between the nature of the information in the attributes, compared to the child elements.

Pre-defined Entity References

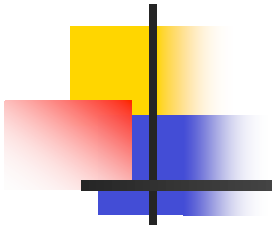
- Since some characters like “<“ and “>” are special in XML, you cannot use them in some places. Eg.

```
<number attribute=">5"> 4 </number>
```
- Wrong syntax!

```
<number attribute="&gt;5"> 4 </number>
```
- Correct!

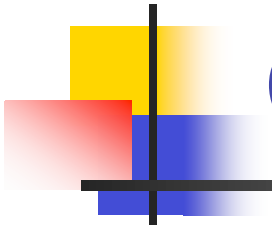
- There are 5 pre-defined entity references to alleviate this:

&lt;	The < character
&gt;	The > character
&amp;	The & character
&apos;	The ‘ character
&quot;	The “ character



Defining New Entities

- Besides the predefined entities, you may also define new entities in the XML document.
- This can be useful for example for:
  - Reference to commonly used names.
  - Multi-language support.
  - Managing binary files,
  - Etc.

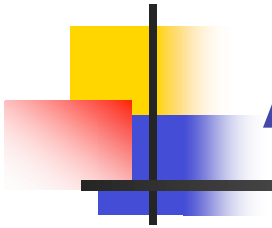


CDATA Sections

- Since XML is very sensitive to characters like “<“ and “&”, you can define character data sections in an XML document which are **not** suppose to be parsed.

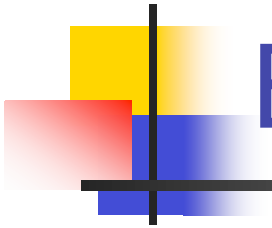
- Eg.

```
<![CDATA
        I’m free to use any of my own special
        characters like <&*@!]% in here!!!
]>
```



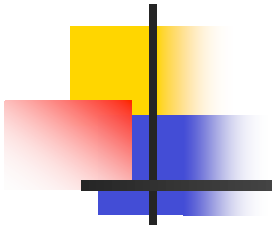
A Summary

- Things to watch out for when constructing a well-formed XML document:
  - Should have an XML declaration at the beginning.
  - Include at least the root document.
  - Include both start and end tags for non-empty elements.
  - Use “/” for empty elements tags.
  - The root element must contain all other elements.
  - Nest all elements properly - no overlaps.
  - Use unique attribute names.
  - Use quotes for attribute values.
  - Use the pre-defined entity references instead of the original characters.



Extra Reading

- Required Reading:
  - Chapters 1-4 in the Sybex textbook - for details of creating XML documents.
- Official Reference
  - XML 1.0 Recommendation
    - http://www.w3.org/TR/REC-xml



Next lecture...

- What makes a **valid** document?