XML

CPSC 315 – Programming Studio

These are slides from Dr. John Keyser's 315 lecture

Consistent Data Transfer

- Transfer of data has become increasingly important
- Can't assume control of all ways data is created and used
  - Cross-platform, cross-system, etc.
  - People want to access data for their own purposes
  - People want to use data from several sources
- Data may be more complicated than "traditional" formats would support
  - E.g. ASCII text only good for some text documents
- Need a more universal means of transferring data

Markup Languages

- Idea is to "tag" information to give a sense of its meaning/semantics
- How that is handled is up to reader
- Usually separates presentation from structure
- Examples:
  - HTML: standard web page information, interpreted by browsers
  - TeX/LaTeX: document specification, style descriptions determine how it is laid out

XML

- eXtensible Markup Language
- Extensible: able to define additional "tags"
  - Specific tags and the semantics associated with them allow specifications of different languages
- Developed by the World Wide Web Consortium (W3C) to help standardize internet information transfer
- Now used as the basis for many specialized languages
  - Each has its own semantic requirements

XML Characteristics

- Straightforward to use on the internet
- Easily processed/parsed
- Human-readable
- Capable of expressing wide range of applications
  - Including hierarchies, tables
- Can be very large/verbose

XML Document Text

- Intermingled character data and markups
- Markups:
  - Start/End tags (and empty element tags)
  - Entity/Character references
  - Comments
  - CDATA delimiters
  - Processing Instructions
  - XML/Text declarations
  - Document type declarations

Basic XML Syntax

- Some prolog/header
  - Possibly describing/referring to type of XML
- Single root element
- More elements forming a tree
  - Elements fully “nest” inside each other
  - Can have any number of children elements
- Elements begin with a start tag, end with an end tag
  - `<Elem>`Stuff in element`</Elem>`

Tag Format

- Starting Tags can declare attributes
  - `<TagName Attr1=“…” Attr2=‘…’>`
  - Note that attributes can use “ or ‘
- Ending Tags match starting tag name, but with a / preceding
  - `</TagName>`
- Character data (and maybe other elements) in between start/end tags
- Empty element:
  - `<Elem/>`
  - Equivalent to `<Elem></Elem>`

Entity/Character References

- Note: Some character patterns are “reserved”
  - <, >, &, ‘, “
- An entity reference is a name given to a character or set of characters
  - Used for any other things to be repeated
    - General entity form: `&Whatever;`
  - Used for the “reserved” chacters
    - `&lt;` <, `&gt;` >, `&amp;` &, `&quot;` “, `&apos;` ‘

Character References

- Character References are specialized
- Use the form `&#…;` where the … is a reference to a character in an ISO standard
  - `&#38;` is an &

Comments

- Begin with `<!--`
- End with `-->`
- Everything in between is ignored

```
<!-- This is a comment -->
```

CDATA sections

- Used to note a section that would otherwise be viewed as markup data
- `<![CDATA[ … ]]>`

```
<![CDATA[ <b>This <a>is</b>not</a>bad ]]>
```

Processing Instructions

- Allow documents to contain instructions for applications reading them
  - "Outside" the main document
- `<? Target … ?>`
- Target is the target application name
  - Any other instructions follow

```
<? MyReader -o3 -f input.dat ?>
```

XML/Text Declarations

- Documents should start with declaration of XML type used, in a prolog:
  - `<?xml version="1.0" ?>`
- Other documents "included" should also have such a prolog, as the first line

XML Semantics

- Semantics must be declared to determine what is valid syntax
  - Tags allowed and their attributes, entities
  - Does not say how it is processed
- Can be located in XML document itself
- Can be contained in separate Document Type Declaration (DTD)
- Newer XML Schema definitions, which capture semantics in an XML-like document
  - But drawbacks, including difficulty to use, not as universally implemented, large size, etc.

Document Type Declaration: **DTD**

- Defines constraints on the structure of the XML
- Comes before first element
- Either defines or points to external definition of Document Type Definition (DTD)
- External: `<!DOCTYPE Name SYSTEM url>`
- Internal: `<!DOCTYPE Name […]>`
- The DTD can be standalone (no further external references) or not

Element Declarations

- Define elements and allowed content (character data, subelements, attributes, etc.)
- `<!ELEMENT Name Content>`
  - `Name` is the unique name
  - `Content` describes that type of element
- Options for Content:
  - `EMPTY` – nothing allowed in the element
  - `ANY` – no restrictions
  - Children elements only
  - Mixed character and children elements

Element Declarations: Child element content

- When an element has (only) child elements within it
- Specify using:
  - Parentheses `()` for grouping
  - The `,` for sequencing
  - The `|` for "choice of"
  - The `+` (one or more), `*` (zero or more), or `?` (zero or one) modifiers.
    - If no modifier, means "exactly once"

Example of Child elements

```
<!Element book (
   title,
   coverpage,
   tableofcontents?,
   editionnote*,
   preface?,
   (chapternumber, chaptertitle, chaptertext)+,
   index?
)>
```

Element Declarations: Mixed element content

- When an element can contain both character and child elements
- The character text is denoted as a kind of special element name: `#PCDATA`

```
<!ELEMENT story (#PCDATA|a|b|c)*>
```

Attribute Declarations

- Define allowed attribute names, their types, and default values
- `<!ATTLIST ElementName Attribute*>`
  - `ElementName` is the name of the element those attributes belong to
  - Repeat attribute definition as many times as needed

Attribute Declaration: Types

- `Name Type DefaultValue`
- `Name` is the attribute name
- `Type`:
  - `CDATA` : string
  - Enumerated: specified via a comma-separated list in parentheses
  - Tokenized: a limited form, specified by some other rule defined in the DTD
  - Several variations

Attribute Declaration: Defaults

- Specify a default value
  - Also specify whether attribute is needed in the element
- `#REQUIRED`
  - This attribute must be specified each time (no default)
- `#IMPLIED`
  - No default is specified
- Otherwise, use the default value given
  - Precede by `#FIXED` if it must always take that default

Attribute Declaration Example

```
<!ATTLIST Book
  title      CDATA  #REQUIRED
   author     CDATA  "anonymous"
   publisher CDATA  #IMPLIED
   category  (fiction,nonfiction) "fiction"
   language  CDATA  #FIXED 'English'
>
```

Entity Declarations

- Entity References should be declared
- Internal Entity:
  - `<!ENTITY Name ReplacementText >`

```
<!ENTITY CR "Copyright 2008">
…
&CR;
```

- External Entity:
  - `<!ENTITY Name SYSTEM url >`

```
<!ENTITY BP SYSTEM "http://this.com/BP.xml">
…
&BP;
```

  - There are also other variations on external entities

Parameter Entities

- Like general entities, but refer to entities to be used in the Document Type Declaration
- Use a % instead of an &

```
<!ENTITY % newdef SYSTEM
  "http://this.com/newdef-xml.entities">
…
%newdef;
```

Conditionals (in the DTD)

- Used in the DTD to apply different rules
- `<![Condition[…]]>`
  - If Condition is `INCLUDE` then keep
  - If Condition is `IGNORE` then skip
- Combine with parameter entities:

```
<!ENTITY % addborder 'INCLUDE'>
…
<![%addborder;[
… (stuff to draw border) …
]]>
```

XML Namespaces

- Different XML definitions could define the same element name.
- If we want to use both, could have conflict.
- Can distinguish using namespaces.

```
<a:book>…</a:book>
<b:book>…</b:book>
```

Defining XML Namespaces

- xmlns attribute in definition of element

xmlns:prefixname="URL"

```
<a:book
  xmlns:a=http://this.com/adef>
```

- Can be defined in first use of element or in XML root element.
- Can define a "default"
  - No prefix needed, leave off : also

Summary/More Information

- XML has become a standard way of transferring information, especially over the internet
- Provides flexibility to represent a wide range of data.

- Many texts/online tutorials about XML
- W3C "official" pages:

  http://www.w3.org/XML/

  See in particular the XML 1.0 specs (more than the 1.1 specs)