

XML

Extensible Markup Language

- Used to describe data
 - common file/document structure
 - data sharing among applications
 - document validation
 - parsing libraries
 - generation libraries

Issues

- Syntax
- Document Type Definition/Schema
- Browser Support
 - CSS
 - XSL (Extensible Style Language)

XML Syntax

- Tags (like HTML)
 - tag names are not fixed (like in HTML).
 - tag names **are** case sensitive
 - **every** start tag must have a corresponding end tag.
 - can use things like `
`, ``
- Entire document is a single strict hierarchy.
- Attributes (like HTML)
 - attribute values **must be quoted**

[Student Record in XML \(viewable\)](#)

Document *elements*

- An element begins with a start tag and ends with the corresponding end tag.
- The element *contains* everything between the start and end tags
 - text (character data)
 - other elements

```
<foo>  
    Hello  
    <blah>Bye</blah>  
</foo>
```

Tag Names

- Characters can be

- alphabetic
- numeric
- ' '
- _

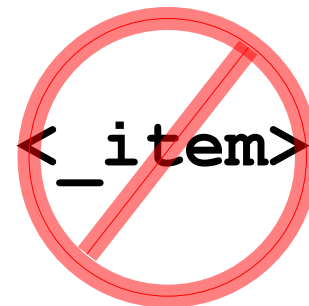
Other punctuation characters are legal, but best to avoid. Never use '-', '.' or ':' !!!

- Must start with alphabetic character

`<foo22>`

`<super_hero>`

`<stolen_song>`



More about Tag Names

- In general: use tag names that are descriptive.
- Never use the tag name 'xml', or in fact, any tag name that starts with 'xml'.
- You usually don't need long tag names
 - the context typically is also descriptive.

```
<student>
  <name>Dave</name>
  <grades>
    <tests>
      <average>99.5</average>
    </tests>
    <hw>
      <average>99.35</average>
    </hw>
  ...
XML
```

Attribute Values must be quoted

```
<student id="660123456">
```

```
<h3 align="center">
```

```
<table border=0 cellspacing=2>
```

```
<stolen_song name="Hey Jude">
```

Reserved Attribute Names

- **xml:lang**: used to specify the language used to represent the text of a tag.

```
<name xml:lang="en">Joe</name>
```

- **xml:space**: used to indicate whether whitespace is significant or not
 - The default is that whitespace is significant
 - This is different than HTML! (whitespace is not significant) in HTML.

Namespaces

- Used to avoid tag name *collisions*.
- tag name is preceded by a namespace:

```
<it:student>
```

```
<it:grade>
```

```
<it:term_project>
```

Document Prolog

- XML Declaration

- top line in any xml document
- indicates the xml version (use 1.0).
- Can also indicate document encoding.

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

- Document Type Definition

- declares valid tag and attribute names/values
- used for validation
- DTD is optional.

XML Document (Prolog) Example

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

```
<!DOCTYPE student SYSTEM "stu.dtd">
```

```
<student>
```

```
  <name>Dave</name>
```

```
  <grades>
```

```
    <tests>
```

```
      <average>99.9</average>
```

```
    </tests>
```

```
  ...
```

Document Type Definition

- Formal definition of the structure (hierarchy) of a specific kind of XML document.
- List of *legal* elements
- Lists of *legal* attributes.

domain specific information.

- Example: what a *student record* should look like as an XML document.

DTD

- A DTD can be *inline* or stored in an external document.
 - much like you can have a CSS style section in an HTML file, or refer to an external style sheet.

```
<?xml version="1.0">
<!DOCTYPE student [
  <!ELEMENT student (name,grades)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT grades (tests,hw)>
  ...
]>
```

DTD Syntax

- A DTD is not an XML document
 - there is an alternate format for specifying the same information that is in the form of an XML document
 - XML Schema Definition (XSD)
- DTD comes from SGML, which was around long before XML, HTML, the web, etc.
- Simple syntax, but it's very different than XML.

DTD and XML

- An XML document is made up of:
 - Elements, Tags, Attributes
 - Entities shortcuts, special chars
 - PCDATA parsed character data
 - CDATA character data
- DTD describes how these can be combined to create a specific kind of document.

DTD: Element definition

- Element name (tag that contains the element).
- what sub-elements must be/can be within the element.
- whether sub-elements can appear once, multiple times, etc.
- The order of subelements within the element.
- Whether PCDATA can appear within the element

DTD Element Examples

- `<!ELEMENT student (name,id,email)>`
 - must have these subelements (in order!)
 - `<name>...</name>`
 - `<id>...</id>`
 - `<email>...</email>`
- `<!ELEMENT name (#PCDATA)>`
 - name element must include only text (no subelements).
- `<!ELEMENT id (#PCDATA|ssn)>`
 - id element must contain either text, or a ssn element.

Other ELEMENT definitions

- You can also define elements that can contain
 - optional content (`grades?`) or (`grade*`)
 - mixed content (`test|hw|lab`) *
 - one or more instances of the same kind of subelement (`grade+`)
 -
- There are more ways to define elements...

Attributes

- For each element type, you can specify legal attributes and what kind of values each can have.
- There are many *types* an attribute can have (check a DTD reference for the full list).
- The type defines what values the attribute can have
 - could be CDATA (any text)
 - could be one of a list of possible values
 - could be a unique ID

Defining an Attribute

```
<!ATTLIST element_name  
  attribute_name attribute_type  
  default value>
```

Examples:

```
<!ATTLIST p color CDATA "black">
```

```
<!ATTLIST p size (small|large) "small">
```

Other attribute definition options

- You can also specify attributes as:
 - required (must be present)
 - implied (optional, no default value)
 - fixed (only one valid value)

Declaring an entity

```
<!ENTITY entity_name entity_value>
```

Examples declarations:

```
<!ENTITY me "Dave">
```

```
<!ENTITY semester "Fall 2007">
```

Example usage in XML document:

```
<course>
```

```
  <instructor> &me; </instructor>
```

```
  <term> &semester; </term>
```

Back to XML

- It's perfectly fine to create XML documents that do not have a DTD
 - you don't have any way to formally validate the document.
- XML documents can include comments, they look just like HTML comments

```
<!-- anything you want here -->
```

XML and browsers

- You can view an XML file in a browser
 - needs to be a relatively modern version.
- Typically you will see hierarchical view of the document including tag names, attributes, etc.
 - in some browsers you can collapse levels in the hierarchy.
- Sample XML document: [Student Record](#)

XML and browsers

- A browser has no way of knowing how to draw an XML document (other than showing the hierarchy):
 - the tags don't mean anything to the browser, they are not like HTML tags which the browser understands.
- We can tell the browser how to display the tags in our XML documents.
 - CSS style rules

Some useful CSS properties for XML documents

- The CSS property `display` can be used to tell the browser to display an element as a block:
 - rendered within a bounding box
 - separated from surrounding content.
- Example Rule:

```
name { display:block; }
```
- You can use any CSS property (`display` just turns out to be useful for XML).

Student with CSS style

- Same XML file, but now includes this in the *prolog*:

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

- This is how you tell the browser about an external style sheet.

[student with CSS applied](#)

[The CSS style file \(viewable version\)](#)

CSS limitations

- Using CSS, you can't control the display of an XML document very much
 - can't change the order
 - can't restrict display of some elements
 - can't add additional content (headings, etc).
- There is a much more powerful form of stylesheet used with XML documents:
 - XSL Extensible Stylesheet Language

XSL

- An XSL stylesheet is itself an XML document.
 - There is a DTD that defines what an XSL document should look like.
- XSL allows you to add content, rearrange elements, duplicate elements, etc.
- XSL is actually more of a programming language than a style declaration.
 - There is a significant learning curve for XSL...

XSL Example

- The Same XML file (student record):

[Student with XSL style](#)

[XSL Style sheet \(viewable\)](#)

[All the sample files](#)

Well Formed and *Valid* Documents

- A document that has proper XML syntax is called "Well Formed"
- A document that also has a DTD and obeys the rules in the DTD is called "valid"
- There are many stand-alone validation programs, as well as web-based validators.

`http://validator.w3.org/`