

## Hypermedia and the Web – Lecture Notes

### XML DTD Language

Goal of Document Type Definitions (DTDs) is to make it possible to machine-check some important aspects of the syntactic correctness of XML documents.

The simplest form of syntactic correctness is *well-formedness*. A well-formed XML document has correctly formatted XML elements (angle-brackets and slashes in the right places, attributes having correct syntax), and every start element has a corresponding end element. However, a well-formed XML document could have elements that are incorrectly positioned relative to other elements, or could contain a corrupted tree structure. DTDs permit checking these properties.

DTDs can check for:

- correct document tree structure
- correct lists of attributes
- whether a specific element should belong in a given XML document at all

DTDs can also specify default values for attributes, and some value checking on attribute values.

DTDs do not:

- perform type checking on element values
- syntax checking on element values
- handle extensible documents very well, where arbitrary elements can appear at places in the document structure

### Specifying Elements

`<!ELEMENT element_name content_specification>`

The element name can be any legal XML name.

There are several choices for content specification:

`#PCDATA` – parsed character data :: can have character data (contents), but no child elements

Child elements:

`(child_elem)` – a single child element

`(child_elem1, child_elem2)` – two child elements, where `elem1` must come before `elem2`

Example:

`<!ELEMENT date (month, day, year)>`

`<!ELEMENT month #PCDATA>`

`<!ELEMENT day #PCDATA>`

```
<!ELEMENT year #PCDATA>
<date>
  <month>May</month>
  <day>5</day>
  <year>2004</year>
</date>
```

This is valid with respect to the DTD.

```
<date>
  <year>2004</year>
  <month>May</month>
  <day>5</day>
</date>
```

This is **not** valid, since the year element comes before month, which differs from the DTD specification.

### **Number of children:**

- ? - Zero or one element instances allowed
- \* - Zero or more element instances allowed
- + - One or more element instances allowed

### **Choice among elements:**

Can also specify that you have a choice among elements:

```
( elem_choice1 | elem_choice2 | elem_choice3 | ... )
```

```
<!ELEMENT library_item ( book | periodical | CD | DVD )>
```

Each `library_item` contains either one book, or one periodical, or one CD, or one DVD element. Cannot have, say, a book and a DVD as children of the same `library_item` element.

### **Empty elements:**

Specify that the element must always be empty (is being used as a value in an enumeration, or only has content stored in attributes).

```
<!ELEMENT elem_name EMPTY>
```

### **Any elements:**

Specify that a specific XML element can contain any kind of child element, so long as they are defined in the DTD.

```
<!ELEMENT elem_name ANY>
```

### Specifying Attributes

ATTLIST declarations define XML attributes that can appear on XML elements.

```
<!ATTLIST elem_name attr_name attribute_type attribute_defaults>
```

Example:

```
<!ATTLIST image source CDATA #REQUIRED>
```

The image element must have a source attribute defined on it, of type CDATA (character data).

Can also combine them:

```
<!ATTLIST image source CDATA #REQUIRED  
width CDATA #REQUIRED  
height CDATA #REQUIRED>
```

Avoids having to repeat the element name, and makes groupings of attributes to elements more clear.

### Attribute Types:

Common attribute types:

CDATA – text strings

Enumerations – choice from a list of possible values

Example:

```
<!ATTLIST date month (Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov |  
Dec) #REQUIRED>
```

<date month="Jan"> is valid

<date month="January"> is not valid (value doesn't exactly match "Jan")

<date month="1"> is also not valid (value doesn't exactly match "Jan")

ID/IDREF

An ID type attribute contains an identifier that is unique within the document. An IDREF must hold the value of one of these IDs. Permits the establishment of relationships among elements in a document that go beyond tree structures.

### **Attribute Defaults**

#IMPLIED – the attribute is optional. Instances may or may not provide a value for the attribute

#REQUIRED – the attribute is required. Instance must provide a value of the attribute

#FIXED – the value is a constant, and cannot be changed. The attribute has the given value, whether or not the attribute is explicitly defined on an element

*value* – A default value (a quoted string)

\*\*\* Examples on transparencies \*\*\*