XML Document Type Definitions (DTDs)

(extracted from material for the course “XML Data Management”, 2013/14)

Werner Nutt

based on slides by Sara Cohen, Jerusalem
Document Type Definitions

- Document Type Definitions (DTDs) impose structure on an XML document.
- Using DTDs, we can specify what a "valid" document should contain.
- DTD specifications require more than being well-formed, e.g., what elements are legal, what nesting is allowed.
- DTDs have limited expressive power, e.g., one cannot specify types.
What is This Good for?

• DTDs can be used to define special languages of XML, i.e., restricted XML for special needs

• Examples:
  – MathML (mathematical markup)
  – SVG (scalable vector graphics)
  – XHTML (well-formed version of HTML)
  – RSS ("Really Simple Syndication", news feeds)

• Standards can be defined using DTDs, for data exchange and special applications

  now, often replaced by XML Schema
Example: MathML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE math PUBLIC "-//W3C//DTD MathML 2.0//EN" "http://www.w3.org/Math/DTD/mathml2/mathml2.dtd">

<math>
  <mrow>
    <msup>
      <mi>x</mi>
      <mn>2</mn>
    </msup>
    <mo>&InvisibleTimes;</mo>
    <mi>y</mi>
  </mrow>
</math>
```
Example: SVG

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
 "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="250px" height="250px"
 xmlns="http://www.w3.org/2000/svg">
 <g fill="red">
  <text font-size="32" x="45" y="60">
   Hello, World!
  </text>
 </g>
 <g fill="blue">
  <text font-size="32" x="50" y="90">
   Hello, World!
  </text>
  <text font-size="32" x="58" y="98">
   Hello, World!
  </text>
 </g>
</svg>
```
Address Book DTD

- Suppose we want to create a DTD that describes legal address book entries.
- This DTD will be used to exchange address book information between programs.
- How should it be written?
- What is a legal address?
Example: An Address Book Entry

```xml
<person>
  <name>Homer Simpson</name>
  <greet>Dr. H. Simpson</greet>
  <addr>1234 Springwater Road</addr>
  <addr>Springfield USA, 98765</addr>
  <tel>(321) 786 2543</tel>
  <fax>(321) 786 2544</fax>
  <tel>(321) 786 2544</tel>
  <email>homer@math.springfield.edu</email>
</person>
```
Specifying the Structure

How do we specify exactly what must appear in a person element?

• A DTD specifies for each element the permitted content

• The permitted content is specified by a regular expression

• Our plan:
  – first, regular expression defining the content of person
  – then, general syntax
What’s in a **person** Element?

Exactly one name,
followed by **at most one** greeting,
followed by **an arbitrary number** of address lines,
followed by **a mix of** telephone and fax numbers,
followed by **at least one** email.

Formally:

```
name, greet?, addr*, (tel | fax)*, email+
```
name, greet?, addr*, (tel | fax)*, email+

**name** = there **must** be a name element

**greet?** = there is an **optional** greet element  
(i.e., 0 or 1 greet elements)

**name, greet?** = the name element is **followed**  
by an optional greet element

**addr** = there are **0 or more** address elements
What’s in a **person** Element? (cntd)

- **name**, greet?, addr*, (tel | fax)*, email+

  - `tel | fax` = there is a tel *or* a fax element
  - `(tel | fax)*` = there are 0 or more repeats of tel or fax
  - `email+` = there are 1 or more email elements
What’s in a **person** Element? (cntd)

name, greet?, addr*, (tel | fax)*, email+

Does this expression differ from:

```plaintext
name, greet?, addr*, tel*, fax*, email+
name, greet?, addr*, (fax|tel)*, email+
name, greet?, addr*, (fax|tel)*, email, email*
name, greet?, addr*, (fax|tel)*, email*, email
```
## Element Content Descriptions

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>element a</td>
</tr>
<tr>
<td>e1?</td>
<td>0 or 1 occurrences of expression e1</td>
</tr>
<tr>
<td>e1*</td>
<td>0 or more occurrences of expression e1</td>
</tr>
<tr>
<td>e1+</td>
<td>1 or more occurrences of expression e1</td>
</tr>
<tr>
<td>e1,e2</td>
<td>expression e2 after expression e2</td>
</tr>
<tr>
<td>e1</td>
<td>e2</td>
</tr>
<tr>
<td>(e)</td>
<td>grouping</td>
</tr>
<tr>
<td>#PCDATA</td>
<td>parsed character data <em>(i.e., after parsing)</em></td>
</tr>
<tr>
<td>EMPTY</td>
<td>no content</td>
</tr>
<tr>
<td>ANY</td>
<td>any content</td>
</tr>
<tr>
<td>(#PCDATA</td>
<td>a₁</td>
</tr>
</tbody>
</table>
addressbook as Internal DTD

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE addressbook [ 
  <!ELEMENT addressbook (person*)>
  <!ELEMENT person (name, greet?, address*, (fax | tel)*, email+)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT greet (#PCDATA)>
  <!ELEMENT address (#PCDATA)>
  <!ELEMENT tel (#PCDATA)>
  <!ELEMENT fax (#PCDATA)>
  <!ELEMENT email (#PCDATA)>
]>
How can we define the possible attributes of elements in XML documents?

General Syntax:

```xml
<!ATTLIST element-name
    attribute-name1 type1 default-value1
    attribute-name2 type2 default-value2
    ...
    attribute-name-n type-n default-value-n>
```

Example:

```xml
<!ATTLIST height dim CDATA "cm">
Attributes (cntd)

<!ATTLIST element-name
  attribute-name1 type1 default-value1
  ...
>

type is one of the following:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDATA</td>
<td>character data (i.e., the string as it is)</td>
</tr>
<tr>
<td>(en1</td>
<td>en2</td>
</tr>
<tr>
<td>ID</td>
<td>value is a unique id</td>
</tr>
<tr>
<td>IDREF</td>
<td>value is the id of another element</td>
</tr>
<tr>
<td>IDREFS</td>
<td>value is a list of other ids</td>
</tr>
</tbody>
</table>

... there are more possibilities (e.g., ENTITY or NM_TOKEN), which we don’t discuss
<!ATTLIST element-name
  attribute-name1 type1 default-value1
  ...>

default-value is one of the following:

<table>
<thead>
<tr>
<th>value</th>
<th>default value of the attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>#REQUIRED</td>
<td>attribute must always be included in the element</td>
</tr>
<tr>
<td>#IMPLIED</td>
<td>attribute need not be included</td>
</tr>
<tr>
<td>#FIXED value</td>
<td>attribute value is fixed</td>
</tr>
</tbody>
</table>
Example: Attributes

<!ELEMENT height (#PCDATA)>

<!ATTLIST height
  dimension (cm|in) #REQUIRED
  accuracy CDATA #IMPLIED
  resizable CDATA #FIXED "yes" >

Need not appear in the doc, will be automatically added by the XML processor

Typical usage:

  xmlns CDATA #FIXED "http://spam.com"
Adding a DTD to a Document

• A DTD can be *internal*
  – the DTD is part of the document file

• … or *external*
  – the DTD and the document are on separate files

• An external DTD may reside
  – in the local file system (where the document is)
  – in a remote file system (reachable using a URL)
Entities are XML macros. They come in four kinds:

- **Character** entities: stand for arbitrary Unicode characters, like: `<`, `;`, `&`, `©`, …

- **Named (internal)** entities: macros in the document, can stand for any well-formed XML, mostly used for text

- **External** entities: like named entities, but refer to a file with well-formed XML

- **Parameter** entities: stand for fragments of a DTD … and are referenced in a DTD
Character Entities

Macros expanded when the document is processed.

Example: Special characters from XHTML1.0 DTD

```xml
<!ENTITY mdash "&#8212;"> <!-- em dash, U+2014 ISOpub -->
<!ENTITY lsquo "&#8216;"> <!-- left single quotation mark, U+2018 ISOnum -->
<!ENTITY copy "&#169;"> <!-- copyright sign, U+00A9 ISOnum -->
```

Can be specified in decimal (above) and in hexadecimal, e.g.,

```xml
<!ENTITY mdash "&#x2014;"> (x stands for hexadecimal)
```
Named Entities

Declared in the DTD (or its local fragment, the “internal subset”)
• Entities can reference other entities
• … but must not form cycles (which the parser would detect)

Example:

```xml
<!ENTITY d "Donald">
<!ENTITY dd "&d; Duck">
```

Using `dd` in a document expands to

```
Donald Duck
```
Represent the content of an external file.
Useful when breaking a document down into parts.

Example:

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE book SYSTEM book.dtd [
  <!ENTITY chap1 SYSTEM "chapter-1.xml">
  <!ENTITY chap2 SYSTEM "chapter-2.xml">
  <!ENTITY chap3 SYSTEM "chapter-3.xml">
]
<!-- Pull in the chapters -->
<book>
  &chap1;&chap2;&chap3;
</book>
```
A document with a DTD is *valid* if it conforms to the DTD, that is,

- the document conforms to the regular-expression grammar,
- types of attributes are correct,
- constraints on references are satisfied.
How many children of the node `<a>` will a DOM parser find?
How many children of the node `<a>` will a DOM parser find now?
Not Every DTD Makes Sense

<DOCTYPE genealogy [
  <!ELEMENT genealogy (person*)>
  <!ELEMENT person (name, dateOfBirth, person, person)>
...
]> Is there a problem with this?
Not Every DTD Makes Sense (cntd)

```xml
<DOCTYPE genealogy [

  <!ELEMENT genealogy (person*)>  

  <!ELEMENT person ( 
      name,                         
      dateOfBirth,                
      person?,                    
      person? )> <!-- mother -->    
      person?  )> <!-- father -->   

  ... ]>

Is this now okay?
```
Weaknesses of DTDs

- DTDs are rather weak specifications by DB & programming-language standards
  - Only one base type: PCDATA
  - No useful “abstractions”, e.g., sets
  - IDs and IDREFs are untyped
  - No constraints, e.g., child is inverse of parent
  - Tag definitions are global

- Some extensions impose a schema or types on an XML document, e.g., XML Schema