

Semi-structured Data

4 - Document Type Definitions (DTDs)

Outline

- DTDs at First Glance
- Validation
- Document Type Declaration
- Internal DTD Subsets
- Element Declarations
- Attribute Declarations
- Entity Declarations (by Example)
- Namespaces and DTDs
- Limitations of DTDs

DTDs at First Glance

- Agreement to use only certain tags - **interoperability**
- Such a set of tags is called **XML application** - application of XML on a particular domain (e.g., phonebook, real estate, etc.)

```
<person>
  <name>
    <first> Wolfgang </first>
    <last> Dvorak </last>
  </name>
  <tel> 18441 </tel>
  <fax> 918441 </fax>
  <email> dvorak@dbai.tuwien.ac.at </email>
</person>
```

```
<house>
  <address>
    <street> Bräuhausegasse </street>
    <number> 49 </number>
    <postcode> A-1050 </postcode>
    <city> Vienna </city>
  </address>
  <rooms> 3 </rooms>
</house>
```

DTDs at First Glance

- **Schema** - the markup permitted in a particular application
- Many different XML **schema languages** available:
 - Document Type Definitions (DTDs)
 - W3C XML Schema
 - REgular LAnguage for XML Next Generation (RELAX NG)
 - Schematron
 - ...
- In the context of this course we are going to see **DTDs** and **W3C XML Schema**

...but for the moment let us focus on DTDs

DTDs at First Glance

- A DTD lists all the elements and attributes the document uses

```
<!ELEMENT person (name, tel, fax, email+)>
```

```
<!ATTLIST person id_number ID #REQUIRED>
```

```
<!ELEMENT name (first, last)>
```

```
<!ELEMENT first (#PCDATA)>
```

```
<!ELEMENT last (#PCDATA)>
```

```
<!ELEMENT tel (#PCDATA)>
```

```
<!ELEMENT fax (#PCDATA)>
```

```
<!ELEMENT email (#PCDATA)>
```

ATTENTION: The order of the declarations is not significant

DTDs at First Glance

- The order of the declarations is not significant

```
<!ELEMENT person (name, tel, fax, email+)>  
<!ATTLIST person id_number ID #REQUIRED>  
<!ELEMENT name (first, last)>  
<!ELEMENT first (#PCDATA)>  
<!ELEMENT last (#PCDATA)>  
<!ELEMENT tel (#PCDATA)>  
<!ELEMENT fax (#PCDATA)>  
<!ELEMENT email (#PCDATA)>
```

```
<!ELEMENT last (#PCDATA)>  
<!ELEMENT fax (#PCDATA)>  
<!ELEMENT email (#PCDATA)>  
<!ELEMENT first (#PCDATA)>  
<!ELEMENT tel (#PCDATA)>  
<!ELEMENT person (name, tel, fax, email+)>  
<!ELEMENT name (first, last)>  
<!ATTLIST person id_number ID #REQUIRED>
```

Both DTDs are equivalent

Validation

- When a document matches a schema is **valid**; otherwise, is **invalid**

<!ELEMENT person (name, tel, fax, email+)>

<!ATTLIST person id_number ID #REQUIRED>

<!ELEMENT name (first, last)>

<!ELEMENT first (#PCDATA)>

<!ELEMENT last (#PCDATA)>

<!ELEMENT tel (#PCDATA)>

<!ELEMENT fax (#PCDATA)>

<!ELEMENT email (#PCDATA)>

<person id_number="E832740">

<name>

<first> Wolfgang </first>

<last> Dvorak </last>

</name>

<tel> 18441 </tel>

<fax> 918441 </fax>

<email> wolfgang.dvorak@tuwien.ac.at </email>

<email> dvorak@dbai.tuwien.ac.at </email>

</person>



Validation

- When a document matches a schema is **valid**; otherwise, is **invalid**

<!ELEMENT person (name, tel, fax, email+)>

<!ATTLIST person id_number ID #REQUIRED>

<!ELEMENT name (first, last)>

<!ELEMENT first (#PCDATA)>

<!ELEMENT last (#PCDATA)>

<!ELEMENT tel (#PCDATA)>

<!ELEMENT fax (#PCDATA)>

<!ELEMENT email (#PCDATA)>

<person id_number="E832740">

<name>

<first> Wolfgang </first>

<last> Dvorak </last>

</name>

<fax> 918441 </fax>

<tel> 18441 </tel>

<email> wolfgang.dvorak@tuwien.ac.at </email>

<email> dvorak@dbai.tuwien.ac.at </email>

</person>



Validation

- **Validating parsers** - check both for well-formedness and validity
- **Validating errors** may be ignored (unlike well-formedness errors)
- Whether a validity error is serious depends on the application


ATTENTION: Validity errors are not necessarily fatal

Document Type Declaration


- A valid document contains a URL indicating where the DTD can be found
- This is done via the **document type declaration** - after the XML declaration

```
<!DOCTYPE person SYSTEM "http://www.mysite.com/dtds/person.dtd">
```

root element
of the document



where the DTD
can be found



ATTENTION: DTD = Document Type Definition (not Declaration)

Document Type Declaration

- **Relative URL** - if the document and the DTD reside in the same base site

```
<!DOCTYPE person SYSTEM "/dtds/person.dtd">
```

- **Just the file name** - if the document and the DTD are in the same directory

```
<!DOCTYPE person SYSTEM "person.dtd">
```


Document Type Declaration: Public IDs

```
<!DOCTYPE person SYSTEM "http://www.mysite.com/dtds/person.dtd">
```

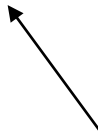
- The keyword **SYSTEM** is use for DTDs defined by the user
- For official, publicly available DTDs, the keyword **PUBLIC** is used

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "xhtml11.dtd">
```

Public ID
uniquely identifies
the XML application in use



Backup URL
in case the public ID
is not recognizable



... but public IDs are not used very much in practice

Internal DTD Subsets

- A DTD can be directly given in the document (between [])

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

```
<!DOCTYPE person [  
  <!ELEMENT person (name, tel, fax, email+)>  
  <!ATTLIST person id_number ID #REQUIRED>  
  <!ELEMENT name (first, last)>  
  <!ELEMENT first (#PCDATA)>  
  <!ELEMENT last (#PCDATA)>  
  <!ELEMENT tel (#PCDATA)>  
  <!ELEMENT fax (#PCDATA)>  
  <!ELEMENT email (#PCDATA)>  
]
```

```
>  
<person id_number="E832740">  
  <name>  
    <first> Wolfgang </first>  
    <last> Dvorak </last>  
  </name>  
  <tel> 18441 </tel>  
  <fax> 918441 </fax>  
  <email> wolfgang.dvorak@tuwien.ac.at </email>  
  <email> dvorak@dbai.tuwien.ac.at </email>  
</person>
```

standalone document



Internal DTD Subsets

- Only part of the DTD can be directly given in the document (between [])

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE person SYSTEM "person_text.dtd" [
  <!ELEMENT person (name, tel, fax, email+)>
  <!ATTLIST person id_number ID #REQUIRED>
  <!ELEMENT name (first, last)>
]>
<person id_number="E832740">
  <name>
    <first> Wolfgang </first>
    <last> Dvorak </last>
  </name>
  <tel> 18441 </tel>
  <fax> 918441 </fax>
  <email> wolfgang.dvorak@tuwien.ac.at </email>
  <email> dvorak@dbai.tuwien.ac.at </email>
</person>
```

person_text.dtd:

```
<!ELEMENT first (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

not a standalone
document

Internal DTD Subsets

- DTD = internal DTD subset [external DTD subset

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE person SYSTEM "person_text.dtd" [
  <!ELEMENT person (name, tel, fax, email+)>
  <!ATTLIST person id_number ID #REQUIRED>
  <!ELEMENT name (first, last)>
]>
<person id_number="E832740">
  <name>
    <first> Wolfgang </first>
    <last> Dvorak </last>
  </name>
  <tel> 18441 </tel>
  <fax> 918441 </fax>
  <email> wolfgang.dvorak@tuwien.ac.at </email>
  <email> dvorak@dbai.tuwien.ac.at </email>
</person>
```

internal DTD subset

person_text.dtd:

```
<!ELEMENT first (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

external DTD subset

ATTENTION: The two subsets must be compatible - no multiple declarations


Up to Now

- **DTDs at First Glance**
- **Validation**
- **Document Type Declaration**
- **Internal DTD Subsets**
- Element Declarations
- Attribute Declarations
- Entity Declarations (by Example)
- Namespaces and DTDs
- Limitations of DTDs

Element Declarations

- Every element used in a valid document must be declared
- This is done via an **element declaration**

<!ELEMENT element-name content-specification>



indicates what children the
element must or may have,
and in which order

Element Declarations: #PCDATA

- An element may only contain **parsed character data**

<!ELEMENT name (#PCDATA)>

Valid:

```
<name>
  Wolfgang Dvorak
</name>
```

Invalid:

```
<name>
  <first> Wolfgang </first>
  <last> Dvorak </last>
</name>
```

Element Declarations: Child Elements

- An element must have **one child element**

```
<!ELEMENT person (name)>  
<!ELEMENT name (#PCDATA)>
```

Valid:

```
<person>  
  <name> Wolfgang Dvorak </name>  
</person>
```

Invalid 1:

```
<person>  
  <name> Wolfgang Dvorak </name>  
  <tel> 18441 </tel>  
</person>
```

Element Declarations: Child Elements

- An element must have **one child element**

```
<!ELEMENT person (name)>  
<!ELEMENT name (#PCDATA)>
```

Valid:

```
<person>  
  <name> Wolfgang Dvorak </name>  
</person>
```

Invalid 2:

```
<person>  
  <name> Wolfgang Dvorak </name>  
  Tel: 18441  
</person>
```

Element Declarations: Sequences

- An element has **multiple child element**

```
<!ELEMENT name (first, last)>  
<!ELEMENT first (#PCDATA)>  
<!ELEMENT last (#PCDATA)>
```

Valid:

```
<name>  
  <first> Wolfgang </first>  
  <last> Dvorak </last>  
</name>
```

Invalid 1:

```
<name>  
  <last> Dvorak </last>  
</name>
```

Element Declarations: Sequences

- An element has **multiple child element**

```
<!ELEMENT name (first, last)>  
<!ELEMENT first (#PCDATA)>  
<!ELEMENT last (#PCDATA)>
```

Valid:

```
<name>  
  <first> Wolfgang </first>  
  <last> Dvorak </last>  
</name>
```

Invalid 2:

```
<name>  
  <last> Dvorak </last>  
  <first> Wolfgang </first>  
</name>
```

Element Declarations: Sequences

- An element has **multiple child element**

```
<!ELEMENT name (first, last)>  
<!ELEMENT first (#PCDATA)>  
<!ELEMENT last (#PCDATA)>
```

Valid:

```
<name>  
  <first> Wolfgang </first>  
  <last> Dvorak </last>  
</name>
```

Invalid 3:

```
<name>  
  <first> Wolfgang </first>  
  <middle> J. </middle>  
  <last> Dvorak </last>  
</name>
```

Element Declarations: Number of Children

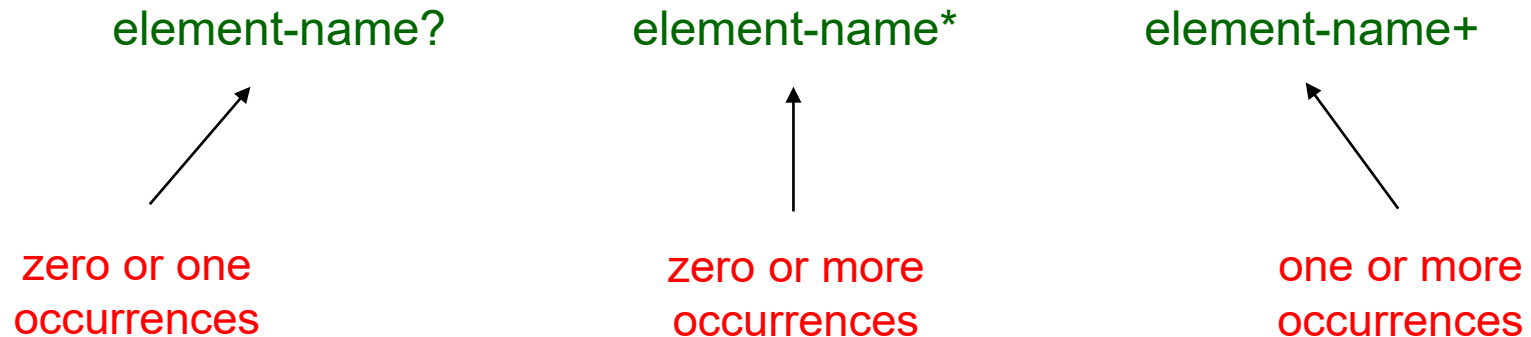
- Not all instances of an element have the same children

| | | |
|---|---|--|
| <pre><name> <first> Wolfgang </first> <last> Dvorak </last> </name></pre> | <pre><name> <first> Wolfgang </first> <middle> J. </middle> <last> Dvorak </last> </name></pre> | <pre><name> <first>Pippilotta</first> <middle> V. </middle> <middle> R. </middle> <middle> P. </middle> <middle> E. </middle> <last>Langstrumpf</last> </name></pre> |
|---|---|--|

- Sequences are not enough to make all the above documents valid
- Occurrence indicators (?,*,+)

Element Declarations: Number of Children

- Occurrence indicators (?,*,+)



ATTENTION: DTDs cannot specify the exact number of occurrences, or say at most k or at least k occurrences

Element Declarations: Number of Children

```
<name>
  <first> Wolfgang </first>
  <last> Dvorak </last>
</name>
```

```
<name>
  <first> Wolfgang </first>
  <middle> J. </middle>
  <last> Dvorak </last>
</name>
```

```
<name>
  <first>Pippilotta</first>
  <middle> V. </middle>
  <middle> R. </middle>
  <middle> P. </middle>
  <middle> E. </middle>
  <last>Langstrumpf</last>
</name>
```

<!ELEMENT name (first, middle*, last)>

<!ELEMENT first (#PCDATA)>

<!ELEMENT middle (#PCDATA)>

<!ELEMENT last (#PCDATA)>

Element Declarations: Choices

- Exactly one child element from a predefined list of elements

```
<!ELEMENT day (Mon | Tue | Wed)>  
<!ELEMENT Mon (#PCDATA)>  
<!ELEMENT Tue (#PCDATA)>  
<!ELEMENT Wed (#PCDATA)>
```

Valid:

```
<day>  
  <Mon> Monday </Mon>  
</day>
```

Invalid:

```
<day>  
  <Mon> Monday </Mon>  
  <Wed> Wednesday </Wed>  
</day>
```

ATTENTION: The separator | is interpreted as exclusive OR

Element Declarations: Parentheses

- Individual elements, sequences, ?, *, + and choices are **rather limited**
- E.g., we cannot say a name element may contain:
 - Just a first name,
 - Just a last name, or
 - A first and a last name with an arbitrary number of middle names
- Combine the above features in an arbitrary way - **(nested) parentheses**

Element Declarations: Parentheses

```
<!ELEMENT person (name, (tel | email))>
```

```
<!ELEMENT name (first, last)>
```

```
<!ELEMENT first (#PCDATA)>
```

```
<!ELEMENT last (#PCDATA)>
```

```
<!ELEMENT tel (#PCDATA)>
```

```
<!ELEMENT mail (#PCDATA)>
```

A person element contains a name element, and either a tel or an email

Element Declarations: Parentheses

```
<!ELEMENT books-catalogue ((title, author, year?)+)>
```

```
<!ELEMENT title (#PCDATA)>
```

```
<!ELEMENT author (#PCDATA)>
```

```
<!ELEMENT year (#PCDATA)>
```

A books-catalogue element consists of a non-empty list of triples of the form title, author, year, with the year being optional

Element Declarations: Parentheses

```
<!ELEMENT name (last  
                | (first, ((middle+, last) | last?))  
                )>
```

```
<!ELEMENT first (#PCDATA)>
```

```
<!ELEMENT middle (#PCDATA)>
```

```
<!ELEMENT last (#PCDATA)>
```

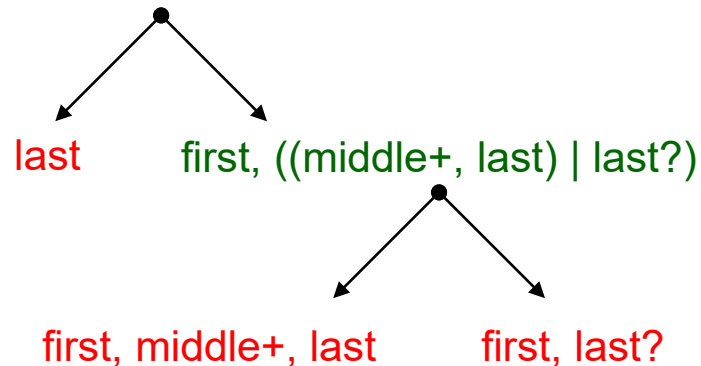
Element Declarations: Parentheses

```
<!ELEMENT name (last  
                | (first, ((middle+, last) | last?))  
                )>
```

```
<!ELEMENT first (#PCDATA)>
```

```
<!ELEMENT middle (#PCDATA)>
```

```
<!ELEMENT last (#PCDATA)>
```



A name element may contain:

- **Just a first name,**
- **Just a last name, or**
- **A first and a last name with an arbitrary number of middle names**

Element Declarations: Mixed Content

- An element may contain both child elements and character data

<definition>

The term <term> Semi-structured Data </term> refers to a form of structured data that does not conform with the formal structure of relational data

</definition>

- Mixed content - (non-whitespace) text and elements

always first



```
<!ELEMENT definition (#PCDATA | term)*>  
<!ELEMENT term (#PCDATA )>
```

ATTENTION: This is the only way to declare mixed content

Element Declarations: Empty Content

- **Empty elements**, i.e., without a content, are declared as

`<!ELEMENT element-name EMPTY>`

Valid: `<element-name></element-name>`
 or
 `<element-name/>`

Invalid: `<element-name>` `</element-name>`

Element Declarations: Any Content

- We can say that an element **simply exists**, without any restrictions

`<!ELEMENT element-name ANY>`

- It is useful during the designing phase of a DTD
- In general, it is a bad design to use ANY in finished DTDs

ATTENTION: ANY does not allow undeclared child elements

Up to Now

- **DTDs at First Glance**
- **Validation**
- **Document Type Declaration**
- **Internal DTD Subsets**
- **Element Declarations**
- Attribute Declarations
- Entity Declarations (by Example)
- Namespaces and DTDs
- Limitations of DTDs

Attribute Declarations

- Every attribute used in a valid document must be declared
- This is done via an **attribute declaration**

<!ATTLIST element-name attr-name₁ attr-type₁ attr-default₁

...

attr-name_n attr-type_n attr-default_n>

data type
of the attribute

default value
of the attribute

ATTENTION: The order of the attributes is not significant

Attribute Declarations: Attribute Types

- Up to now, attribute values can be any string of text
 - ... except the symbols < and & - we need to use < and &
 - DTDs can make stronger statements about the attribute values - **attribute type**
 - There are **ten attribute types** in XML:
 - CDATA
 - NMTOKEN
 - NMTOKENS
 - Enumeration
 - ID
 - IDREF
 - IDREFS
 - ENTITY
 - ENTITIES
 - NOTATION
- details follow
- check out the textbook (XML in a Nutshell, Chapter 3)

Attribute Types: CDATA

- An attribute may contain **any text acceptable in a well-formed document**

`<!ATTLIST book price CDATA #REQUIRED>`

`<!ELEMENT book (#PCDATA)>`

- A price is in the form €20.00 - only CDATA allows for such values

Attribute Types: NMTOKEN

- **XML name token** - legal XML name, but can start with any allowed character
- Recall that XML names can start only with a letter or underscore
- **NMTOKEN** - an attribute can take XML name tokens

```
<!ATTLIST course date NMTOKEN #REQUIRED>  
<!ELEMENT course (#PCDATA)>
```

Valid: <course date="05-03-2015"> SSD </course>

Invalid: <course date="05/03/2015"> SSD </course>

Attribute Types: NMTOKENS

- An attribute may contain a **list of XML name tokens** (separated by whitespace)

```
<!ATTLIST course date NMTOKENS #REQUIRED>  
<!ELEMENT course (#PCDATA)>
```

Valid: <course date="05-03-2015 12-03-2015"> SSD </course>

Invalid: <course date="05/03/2015 12-03-2015"> SSD </course>

Attribute Types: Enumeration

- List of possible values (separated by |)

```
<!ATTLIST course day (Mon | Thu) #REQUIRED>  
<!ELEMENT course (#PCDATA)>
```

Valid: <course day="Thu"> SSD </course>

Invalid: <course day="Sun"> SSD </course>

ATTENTION: The only attribute type that is not an XML keyword

Attribute Types: ID

- An attribute must contain an XML name (not name token) that is **unique**
- Each element has at most one ID attribute - **ID of an element**

```
<!ATTLIST person id_number ID #REQUIRED>  
<!ELEMENT person (#PCDATA)>
```

Valid: <person id_number="_832740"> Wolfgang Dvorak
 </person>

Invalid: <person id_number="832740"> Wolfgang Dvorak
 </person>

Attribute Types: IDREF

- An attribute must contain the **value of some ID type attribute** in the document

```
<!ATTLIST employee emp_id ID #REQUIRED>
<!ATTLIST project proj_id ID #REQUIRED>
<!ATTLIST manager mgr_id IDREF #REQUIRED>
<!ELEMENT employee (#PCDATA)>
<!ELEMENT project (#PCDATA)>
<!ELEMENT manager (#PCDATA)>
```

Valid:

```
<employee emp_id="e1"> E </employee>
<project proj_id="p1"> P </project>
<manager mgr_id="e1"> E </manager>
```

Attribute Types: IDREF

- An attribute must contain the **value of some ID type attribute** in the document

```
<!ATTLIST employee emp_id ID #REQUIRED>
<!ATTLIST project proj_id ID #REQUIRED>
<!ATTLIST manager mgr_id IDREF #REQUIRED>
<!ELEMENT employee (#PCDATA)>
<!ELEMENT project (#PCDATA)>
<!ELEMENT manager (#PCDATA)>
```

?

```
<employee emp_id="e1"> E </employee>
<project proj_id="p1"> P </project>
<manager mgr_id="p1"> E </manager>
```

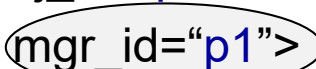
Attribute Types: IDREF

- An attribute must contain the **value of some ID type attribute** in the document

```
<!ATTLIST employee emp_id ID #REQUIRED>
<!ATTLIST project proj_id ID #REQUIRED>
<!ATTLIST manager mgr_id IDREF #REQUIRED>
<!ELEMENT employee (#PCDATA)>
<!ELEMENT project (#PCDATA)>
<!ELEMENT manager (#PCDATA)>
```

Valid:

```
<employee emp_id="e1"> E </employee>
<project proj_id="p1"> P </project>
<manager mgr_id="p1"> E </manager>
```



although conceptually wrong
(manager is a project)

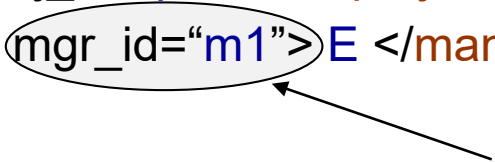
Attribute Types: IDREF

- An attribute must contain the **value of some ID type attribute** in the document

```
<!ATTLIST employee emp_id ID #REQUIRED>
<!ATTLIST project proj_id ID #REQUIRED>
<!ATTLIST manager mgr_id IDREF #REQUIRED>
<!ELEMENT employee (#PCDATA)>
<!ELEMENT project (#PCDATA)>
<!ELEMENT manager (#PCDATA)>
```

Invalid:

```
<employee emp_id="e1"> E </employee>
<project proj_id="p1"> P </project>
<manager mgr_id="m1"> E </manager>
```



m1 is not the value of an ID type attribute

Other Attribute Types

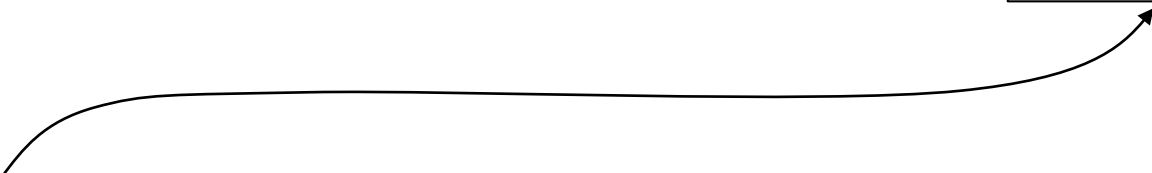
- **IDREFS** - list of IDs occurring in the document
- **ENTITY** – unparsed entity declared in the DTD
- **ENTITIES** - list of unparsed entities declared in the DTD
- **NOTATION** - name of a notation declared in the DTD

... for more details, check out the textbook (XML in a Nutshell, Chapter 3)

Attribute Declarations: Attribute Defaults

- Recall how an attribute declaration looks like

```
<!ATTLIST element-name attr-name1 attr-type1 attr-default1  
...  
attr-namen attr-typen attr-defaultn>
```



| | |
|--------------|---|
| #IMPLIED | optional, no default value |
| #REQUIRED | required, no default value |
| #FIXED value | attribute value is constant and immutable |
| value | the actual default value is given |

Attribute Defaults: #FIXED

```
<!ATTLIST tuwien website CDATA #FIXED "http://www.tuwien.ac.at">
```

Valid:

`<tuwien website="http://www.tuwien.ac.at"> ... </tuwien>`
or
`<tuwien>.. </tuwien>`



even if the attribute is not explicitly stated,
it has the specified value

Invalid:

`<tuwien website="www.tuwien.ac.at"> ... </tuwien>`

Attribute Defaults: Default Value

<!ATTLIST course elective (yes | no) "no">

Valid: <course elective="yes"> ... </course>
or
<course elective="no"> ... </course>
or
<course> ... </course> - the value of elective is no

Invalid: <course elective="true"> ... </course>

Entity Declarations: Example

- Recall that XML predefines five entities (lt, gt, amp, quot, apos)
- DTDs can define more entities via an **entity declaration**
- The following defines the entity ssd:

```
<!ENTITY ssd "Semi-structured Data" >
```

- We can use **&ssd;** anywhere we need to type “Semi-structured Data”

... check out the textbook (XML in a Nutshell, Chapter 3)

Namespaces in DTDs

```
<!-- Students' and University's Evaluation -->
```

```
<course
```

```
  xmlns="http://www.oeh.ac.at"
```

```
  xmlns:univ= "http://www.tuwien.ac.at">
```

```
<title> SSD </title>
```

```
<assessment> Fair </assessment >
```

```
<univ:assessment> Elective </univ:assessment >
```

```
</course>
```

Namespaces are completely
independent of DTDs

```
<!ELEMENT course (title, assessment, univ:assessment)>
```

```
<!ATTLIST course xmlns CDATA #FIXED "http://www.oeh.ac.at">
```

```
<!ATTLIST course xmlns:univ CDATA #REQUIRED>
```

```
<!ELEMENT title (#PCDATA)>
```

```
<!ELEMENT assessment (#PCDATA)>
```

```
<!ELEMENT univ:assessment (#PCDATA)>
```

ATTENTION: The validator does not care about namespaces - some element and attribute names happen to contain colons (:)

Check for Validity

- Easy way: **online validator** - <http://www.xmlvalidation.com/>
- Recommended: **xmllint** - <http://xmlsoft.org/>
 - Portable C library for Linux, Unix, MacOS, Windows, ...
 - Command line call:
 - **xmllint --valid <xml-file-name>**
(for files with Document Type Declaration)
 - **xmllint --dtdvalid <dtd-file-name> <xml-file-name>**
 - Check the TUWEL course: Übung/Tools für die Übung

Limitations of DTDs

- **Not in XML syntax**
 - Different parsers for the document and the DTD
- **A weak specification language**
 - No control on the exact number of child elements
 - Limited selection of data types
 - The notion of inheritance does not exist
- **No explicit support of namespaces**
 - The validator is completely unaware of the existence of namespaces

... W3C XML Schema