XML Declaration

- XML documents should (but do not have to) begin with an XML declaration
- The XML declaration looks like a processing instruction with the name `xml` and the `version`, `standalone` and `encoding` attributes
- Example:

```
<?xml version="1.0" encoding="UTF-16"
  standalone="yes"?>
```

- If the XML is included in the document, it must be the first thing in the document

Valid XML Documents

- An XML document is said to be valid if it respects the format rules defined by an associated Document Type Definition (DTD)
- DTDs are written in a formal syntax that defines:
  - which elements may appear in an XML document
  - what is the allowed content for each element
  - what are the attributes each element may have
- A validating parser compares an XML document against its DTD and can tell if the DTD constraints are violated
- DTDs can be defined inside the XML document or in a external document (or in both ways, as we will see)

Well-formed XML documents

To be well-formed, an XML must adhere to the following rules:

- Every start-tag must have a matching end-tag
- Elements may nest, but not overlap
- There must be exactly one root element
- Attribute values must be quoted
- An element may not have two attributes with the same name
- Comments and processing instructions may not appear inside tags
- No unescaped < or & signs may occur in the character data of an element or attribute

Valid XML Documents (...)

DTDs are associated with XML documents by including a Document Type Declaration in the XML document prolog

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE person SYSTEM
  "http://someserver.org/person.dtd">
<person>
  ...
</person>
```

If the document resides on the same host as the DTD, a relative URL may be used:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE person SYSTEM "/dtds/person.dtd">
...
```

Valid XML Documents (...)

It is sometime useful (e.g., for debugging purposes) to have the DTD defined inside the XML document

XML allows the following syntax:

```
<?xml version="1.0"?>
<!DOCTYPE person [
  <!ELEMENT first_name (#PCDATA)>
  <!ELEMENT last_name (#PCDATA)>
  <!ELEMENT profession (#PCDATA)>
  <!ELEMENT name (first_name, last_name)>
  <!ELEMENT person (name, profession*)>
]>
<person>
  ...
</person>
```

Valid XML Documents (...)

It is sometime useful to mix an internal DTD with an external one. This is possible only if no conflicts arise between the two DTDs (neither one can override the element declarations the other makes (but entity declarations can be overridden)):

```
<?xml version="1.0"?>
<!DOCTYPE person SYSTEM "name.dtd" [
  <!ELEMENT profession (#PCDATA)>
  <!ELEMENT person (name, profession*)>
]>
<person>
  ...
</person>
```

DTD syntax: <!ELEMENT ...> (1/4)

```
<!ELEMENT name content-model>
```

`name` is a valid XML name

`content-model` can be :

- parsed character data, i.e. text:

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

- child elements:

```
<!ELEMENT fax (phone_number)>
```

- Sequences:

```
<!ELEMENT name (first_name, last_name)>
```

- Choices:

```
<!ELEMENT methodResponse (params | fault)>
```

DTD syntax: <!ELEMENT ...> (2/4)

Suffixes can be appended to an element name in a content specification to impose cardinality constraints

- ? zero or one occurence
- * zero or more occurences
- + one or more occurences

```
<!ELEMENT name (first_name, middle_name*, last_name?)>
```

This example imposes that each `name` element contains one `first_name` element followed by zero or more `middle_name` elements and zero or one `last_name` element.

DTD syntax: <!ELEMENT ...> (3/4)

- Mixed content (i.e. text together with child elements):

```
<!ELEMENT paragraph (#PCDATA | bold)*>
```

The #PCDATA keyword must always the first child in the list!

Ex:

```
<paragraph>
  This paragraph contains text that may be
  formatted using a <bold>bold font!</bold>
</paragraph>
```

- Empty elements:

```
<!ELEMENT image EMPTY>
```

DTD syntax: <!ELEMENT ...> (4/4)

- Any content:

  ```
  <!ELEMENT page ANY>
  ```

  Note: ANY does not allow you to use undeclared elements, useful in the initial design of a DTD, discouraged in a finished DTD.

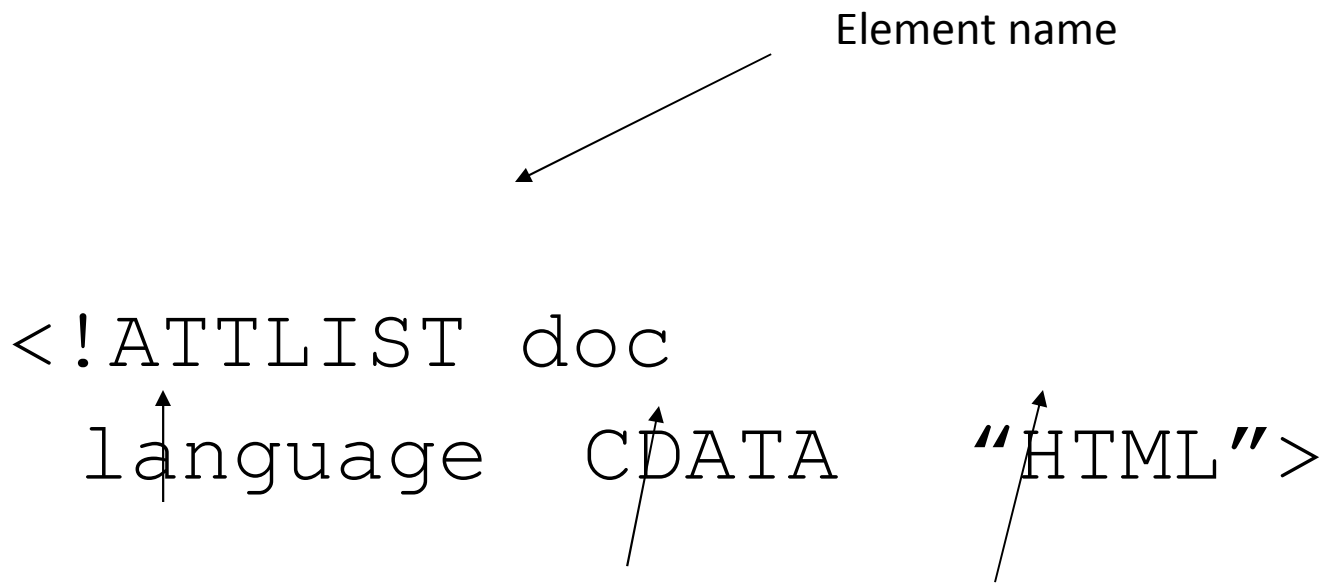
DTD syntax: <!ATTLIST ...> (1/4)

- Attributes are declared for elements using the ATTLIST keyword
- name is the name of the element

```
<!ATTLIST name
  attribute-name1  type  default-value
  attribute-name2  type  default-value
  ...>
```

DTD syntax: <!ATTLIST ...> (2/4)

- Character data: means any character (excluding < and “) but no elements or entities



DTD syntax: <!ATTLIST ...> (3/4)

- Enumeration: list of possible values separated by vertical bars

```
<!ATTLIST date
month  (jan|feb|mar|...|dec)
   #REQUIRED>
```

- Ex:

```
<date month="oct">...</date>
<date month="October">...</date>
  error!
```

DTD syntax: <!ATTLIST ...> (4/4)

- ID: an XML name that is unique within the XML document
- IDREF or IDREFS: a reference (or a whitespace separated list of references) to the ID type attribute of some element in the document
- NMTOKEN or NMTOKENS: a named toked (or a list of named tokens). Named tokens follow the same restrictions of XML names, but any of the allowed characters may be the first character of the token

ID example

```
<!ATTLIST credit-card
number ID #REQUIRED>
```

If we define two `credit-card` elements with the same value for the `number` attribute within the same document, we get a validation error

```
<credit-card number="5555-666-555-
  666"/>
<credit-card number="5555-666-555-
  666"/>
```

IDREF example

```
<!ATTLIST credit-card
number ID #REQUIRED>
<!ATTLIST order
  cardno IDREF #REQUIRED>
```

If we define an order that refers to an ID attribute that is not defined within the document we get a validation error

```
<credit-card number="5555-666-555-
  666"/>
<order cardno="5555-666-555-666"/>
<order cardno="1234-4321-5678-8765"/>
```

The default declaration

- `#IMPLIED`

  Optional attribute. May or may not be declared inside the element.

- `#REQUIRED`

  Required attribute. Each instance of the element must provide a value for the attribute.

- `#FIXED`

  The attribute value is constant and immutable.

- Literal

  The default value is given as a quoted string

An Example of DTD

```
<?xml version="1.0"?>
<!DOCTYPE person [
  <!ELEMENT person (name+,
  profession*)>
  <!ELEMENT name EMPTY>
  <!ATTLIST name first CDATA #REQUIRED
     last CDATA #REQUIRED>
  <!ELEMENT profession EMPTY>
  <!ATTLIST profession value CDATA
  #REQUIRED>
]>...
```

An Example of DTD

```
...
<person>
  <name first="Alan" last="Turing"/>
  <profession value="Computer
  scientist"/>
  <profession value="Mathematician"/>
  <profession value="Cryptographer"/>
</person>
```