Lecture 7 – XML

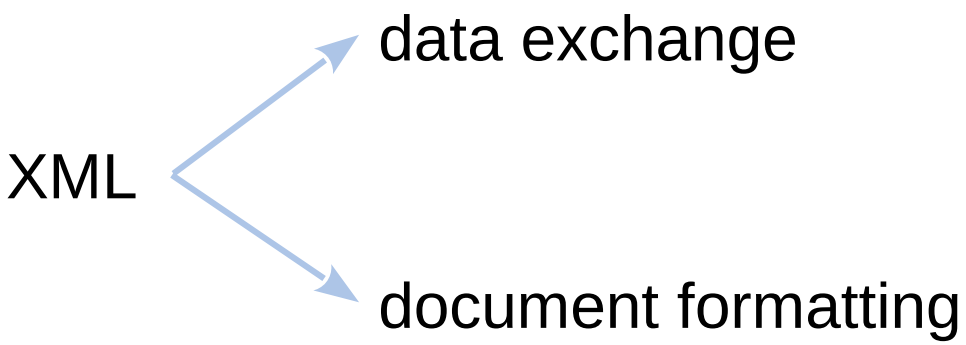
Open Data Management & the Cloud

(Data Science & Scientific Computing / UniTS – DMG)

XML

eXtensible Markup Language

language for encoding documents in a format that is both human-readable and machine-readable.



Markup Language

Annotating (insert metadata) in a text with a distinguishable syntax

Presentational

Hidden from human users: binary codes embedded within document to produce the WYSIWYG ("what you see is what you get") effect.

(e.g Word, …)

Procedural

Embedded in text to provide instructions for programs that process the text.

(e.g. TeX, PostScript)

Descriptive (or **logical**)

Label parts of the document decoupling the structure from its rendering.

(e.g. HTML, XML…)

From Text to Markup for DATA

ASCII text

- simple
- universal
- easy to archive
- can NOT divide data in units

CSV

- simple
- non-data elements ‘,’
- only tables
- no meaning on data fields

XML

- readable
- flexible
- based on tags
- clear meaning of data fields

```
Martedì 17 14:00

Giovedì 19 14:00
```

```
Martedì, 17, 14:00

Giovedì, 19, 14:00
```

```
<lecture>
     <day>Martedì</day>
     <number>17</number>
     <time>14:00</time>
</lecture>

<lecture>
     <day>Giovedì</day>
     <number>19</number>
     <time>14:00</time>
</lecture>
```

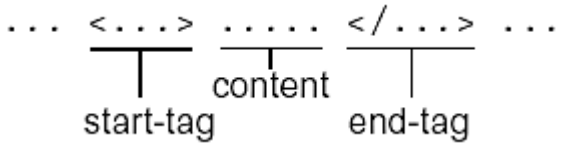
XML Syntax

The first line of an XML document should be

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

XML → documents decomposed into small elements

Element → 3 parts: **start-tag**, data **(element content)** , **end-tag**



start-tag

markup characters **<** and **>** and a *name* : e.g `<tagname>`

end-tag

markup characters **<** and ***/***and **>** and a *name* : e.g `</tagname>`

Comments → ignored by the interpreter

```
<!-- This is a comment -->
```

Elements

- **no predefined element names:** each document define its owns
- **instructions** followed **in** strict **order** (i.e like reading a book)
- **elements** are **not only containers** but can be placeholders

  ```
  ...the page ends here <pageBreak></pageBreak> The next page starts here…
  ```

  in this case the start-tag, end-tag is equivalent to

  ```
  ...the page ends here <pageBreak/> The next page starts here…
  ```

- special characters represented via “escape-code”

  ```
  <code>if ( x < y ) { ... } </code>  → <code> if ( x &lt;y ) { ... }
  </code>
  ```

  “`<`” is substituted into “<” (… and the “&” character? Use “`&`”)

- **elements can be repeated**: same element name, same operation on it
- elements are Case Sensitive: name != Name

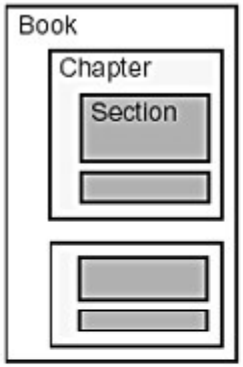
  ```
  <name>XML </Name>
  ```

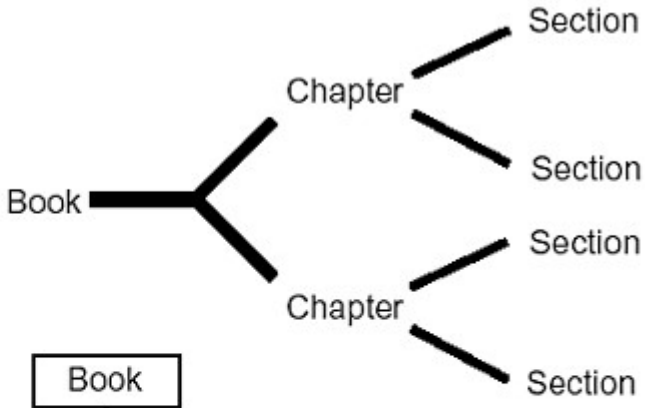
  WRONG!

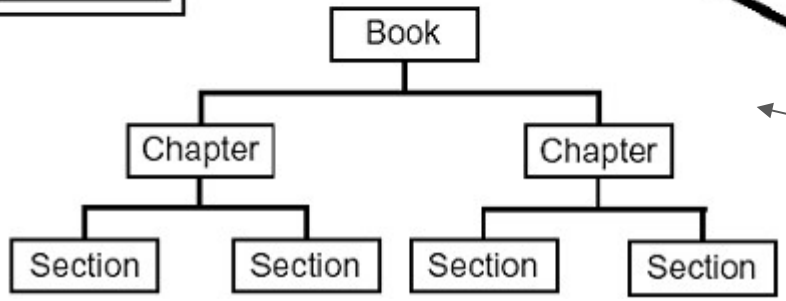
Hierarchies

elements are processed in order and can contain other elements
(e.g. a book that contains many chapter, sections, paragraphs, ...)

box of boxes visualization







tree visualization

a document must be a **single element** called **root**

Layout & Constraints

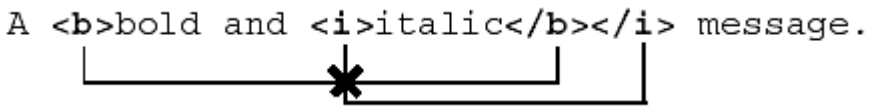
Embedded elements can be on the same line

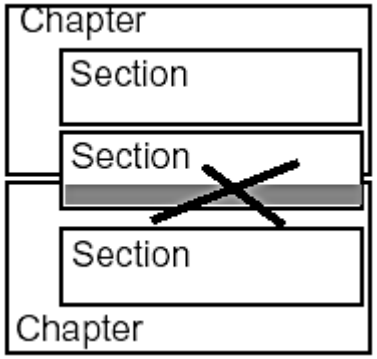
```
<book><chapter><section>...</section></chapter></book>
```

or indented more lines (for clarity)

```
<book>
  <chapter>
    <section>...</section>
    <section>...</section>
  </chapter>
</book>
```

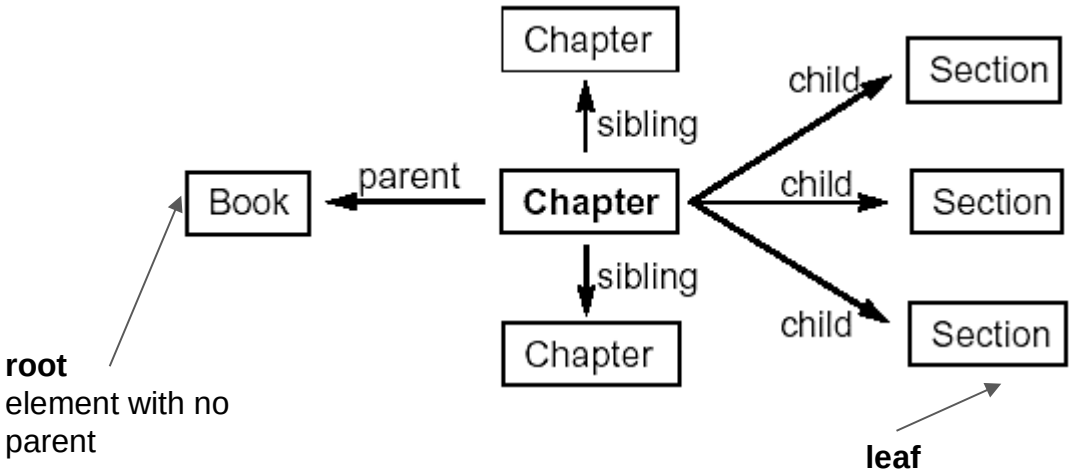
Hierarchical structures are strictly enforced: elements must be completely embedded within another element, or must be completely outside of that other element.





Terminology

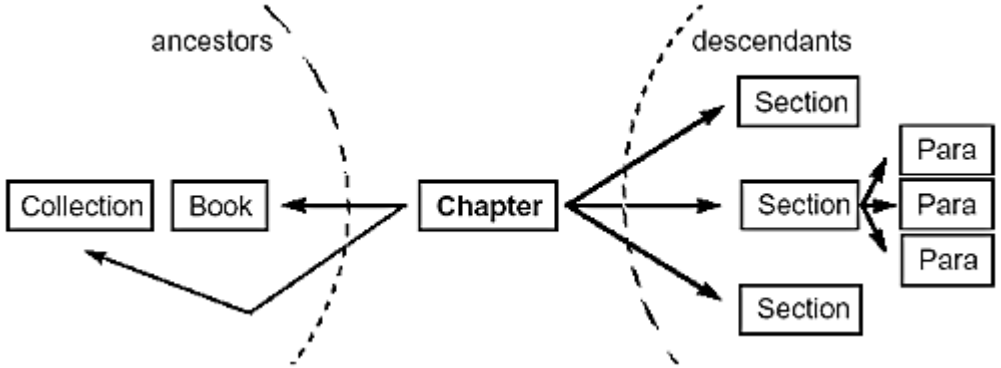
Relation between element **target** and nearby elements



root
element with no parent

leaf
element with no child

```
<parent>
  <sibling>...</sibling>
  <target>
    <child>...</child>
    <child>...</child>
    <child>...</child>
  </target>
  <sibling>...</sibling>
</parent>
```

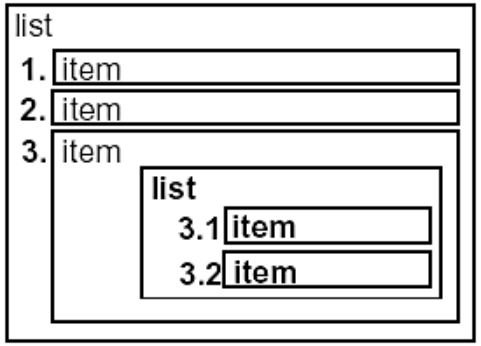



```
<ancestor>
  <ancestor>
    <target>
      <descendant>...</descendant>
      <descendant>
        <descendant>...</descendant>
        <descendant>...</descendant>
        <descendant>...</descendant>
      </descendant>
      <descendant>...</descendant>
    </target>
  </ancestor>
</ancestor>
```

Element Content

Content:

- **data**: `<para> some text </para>`
- **mixed**: `<para> some <pageBreak/> text <name> HERE </name> </para>`
- **recursive**: instances of same type



- Additional information (metadata) as **attributes**
  `<element name="value"> ... </element>`

Legal and Illegal Elements

LEGAL ELEMENT	REASON	ILLEGAL ELEMENT	REASON
`<myElement> </myElement>`	Spaces are allowed after a name.	`<my Element />`	Names cannot contain spaces.
`<my1stElement/>`	Digits can appear within a name.	`<1stName />`	Names cannot begin with a digit.
`<myElement />`	Spaces can appear between the name and the forward slash in a self-closing element.	`< myElement />`	Initial spaces are forbidden.
`<my-Element />`	A hyphen is allowed within a name.	`<-myElement/>`	A hyphen is not allowed as the first character.
`<όνομα />`	Non-roman characters are allowed if they are classified as letters by the Unicode specification. In this case the element name is *forename* in Greek.	`<myElement> </MyElement>`	Start and end tags must match case-sensitively.

Attributes

Additional information about element: metadata `<chapter author="J.J. Abrams">`

- case sensitive
- set in the start-tag
- composed by **name=“**value**”** (key/value pair)

LEGAL ATTRIBUTE	REASON	ILLEGAL ATTRIBUTE	REASON
`<myElement value="Joe's attribute" />`	Single quote inside double quote delimiters.	`<myElement 1stAttribute= "value" />`	Attribute names cannot begin with a digit.
`<myElement value='"a quoted value"' />`	Double quotes inside single quote delimiters.	`<myElement value= 'Joe's attribute' />`	Single quote inside single quote delimiters.
		`<myElement name="Joe" name="Fawcett" />`	Two attributes with the same name is not allowed.
		`<myElement name='Joe" />`	Mismatching delimiters.

Attributes vs. Elements (1)

- Attributes can store information like elements

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <gender>female</gender>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

- Attributes cannot contain multiple values (elements can)
- Attributes cannot contain tree structures (elements can)
- Attributes are not easily expandable (for future changes)

```
<note date="2008-01-10">
  <to>Anna</to>
  <from>John</from>
</note>
```

```
<note>
  <date>2008-01-10</date>
  <to>Anna</to>
  <from>John</from>
</note>
```

```
<note>
  <date>
    <year>2008</year>
    <month>10</month>
    <day>01</day>
  </date>
  <to>Anna</to>
  <from>John</from>
</note>
```

Attributes vs. Elements (2)

- This is not wrong, but please don't do it!

```
<note day="10" month="01" year="2008" to="Anna" from="John"
heading="Reminder" body="Don't forget me this weekend!">
</note>
```

- Metadata should be stored as attributes, and the data itself should be stored as elements.

```
<messages>
  <note id="501">
    <to>Anna</to>
    <from>John</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</from>
  </note>
  <note id="502">
    <to>John</to>
    <from>Anna</from>
    <heading>Re: Reminder</heading>
    <body>I will not!</from>
  </note>
</messages>
```

Attributes vs. Elements (3)

Principle of core content:

- Attribute: information useful to process data
- Element: information useful as data

Principle of readability

If information is intended to be read and understood by a person, use elements

Principle of element/attribute binding

Use an element if you need its value to be modified by another attribute.

Name Conflicts

XML describing musical compositions rating (elements: author, score) and the performance of a musician at a competition (elements: performer, song, score)

```
<competition>
        <author>J Smith</author>
        <score>professional</score>
        <performer>J Smith</performer>
        <song>Piano Song</song>
        <score>57</score>
</competition>
```

There is a name conflict for the `<score>` element!

A user or an XML application will not know how to handle it.

Solving the Name Conflict

Name conflicts in XML can be avoided using a name prefix:

- Use the `c:` prefix for the composer's elements
- Use the `p:` prefix for the performance elements

```
<competition>
        <c:author>J Smith</c:author>
        <c:score>professional</c:score>
        <p:performer>J Smith</p:performer>
        <p:song>Piano Song</p:song>
        <p:score>57</p:score>
</competition>
```

There will be no conflict because the two `<score>` elements have different names.
When using prefixes in XML, a namespace for the prefix must be defined.

Using Namespaces (1)

- Namespaces are defined using attribute with syntax `xmlns:prefix="URI"`
- Namespace can be defined in the start tag of an element or in the XML root element

HTML example

```
<html xmlns="http://www.w3.org/TR/REC-html40">
  ...<p>An HTML paragraph.</p>...
</html>
```

- Namespaces can have an alias (prefix)

HTML alias example

```
<x:html xmlns:x="http://www.w3.org/TR/REC-html40">
  ...<x:p>An HTML paragraph.</x:p>...
</x:html>
```

Using Namespaces (2)

- Namespace locations can be defined within any element

location example

```
<document>
   ...<description>...</description>...
  <X:table xmlns:X="http://www.w3.org/TR/REC-html40">
    ...<X:td>An HTML table cell.</X:td>...
  </X:table>
  ...<summary>...</summary>...
</document>
```

- Any element may declare more than one namespace

multiple declaration example

```
<D:document xmlns:D="file:///DTDs/document.dtd"
            xmlns:X="http://www.w3.org/TR/REC-html40">
  ...<D:description>...</D:description>...
  ...<X:td>An HTML table cell.</X:td>...
  ...<D:summary>...</D:summary>...
</D:document>
```

Using Namespaces (3)

- Attributes from one namespace can be used in elements from another. The attribute names contain the same prefixes as the elements

mixed attribute example

```
<property:house property:style="Georgian" html:style="color:red">
  ...
</property:house>
```

Using Namespaces (4)

- Attribute **xmlns** without prefix is the default namespace for the document

```
<document xmlns="file:///DTDs/document.dtd" xmlns:X="http://www.w3.org/TR/REC-
html40">
   ...<description>...</description>...
   ...<X:td>An HTML table cell.</X:td>...
</document>
```

- Default namespace can be changed at any point in the document hierarchy

```
<document xmlns="file:///Schemas/document.xsd"
          xmlns:X="http://www.w3.org/TR/REC-html40">
  ...
  ...<para>A normal paragraph.</para>...
  ...<X:td>An HTML table cell.</X:td>...
  ...<description>...</description>...

  ...<html xmlns="http://www.w3.org/TR/REC-html40">
     ...<td>An HTML table cell.</td>...
  ...</html>
  ...
  <summary>...</summary>
</document>
```

Note: Differently from elements, attributes with no prefix are NOT considered to belong to the default namespace.

Entities

Markup → divide document into logical components (elements)

Entities → manage components

With entities

- create escape-codes for significant markup characters
- representing characters not available in keyboard or standard sets
- divide long documents
- create re-usable components shared by many documents
- include by reference external binary data, such as images
- assist in the construction of DTDs (Document Type Definitions, next lecture)

Entity are referenced via **&name;**

Built-in Entities

Avoid markup characters confusion:

- **<** for '**<**'
- **>** for '**>**'
- **&** for '**&**'
- **'** for ' **'** ' (in attribute values)
- **"** for ' **"** ' (in attribute values)

Represent any character knowing its number in the set

&#nnn;

With **nnn** a number from 1 to 255 for ASCII set, ISO 8859/1
256 to 65536 for Unicode/ISO10646

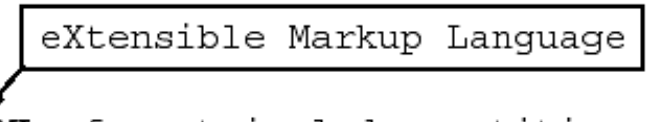
`The caf&#233; is open.` → The café is open.

Entity Declaration

- ENTITY keyword followed by name and definition or reference to entity

```
!DOCTYPE MyDoc [
    <!ENTITY entity1-name "entity-value">
    <!ENTITY entity2-name "entity-value">
    ...
]>
```

- Define constant text




```
<!ENTITY XML "eXtensible Markup Language"> → The &XML; format includes entities.
```

- Represent structured content

```
<!ENTITY Water 'H<sub>2</sub>O'>  → &Water; is water → H2O is water
```

- Large entities can be referred as external code

```
<!ENTITY MyEntity SYSTEM "file:///user/folder/myentity.xml">
```

- Separate binary data from XML data

```
<!ENTITY MyPicture SYSTEM "file:///user/folder/picture.png">
```

XML Navigation

The ability to navigate through XML documents is the key of any XML file

- The meaning of an element can depend on its contextual location
- Every element in XML document has unique contextual location
- Any element in the document is identified by the steps it would take to reach it

e.g. to get “Smith”: `<book><front><author><name><last>`

```
<book>
  <front>
    <author>
      <name>
        <first>John</first>
        <last>Smith</last>
      </name>
    </author>
  </front>
</book>
```

XPath (www.w3.org/TR/XPath) is one of the standards used to walk through a XML file

Patterns in XML Data

XML fragment:

```
<pixel_reading>
      <device>Camera</device>
      <patch>cyan</patch>
      <RGB resolution="8">
            <red>0</red>
            <green>255</green>
            <blue>255</blue>
      </RGB>
</pixel_reading>
```

The information context list from the XML is:

```
<pixel_reading><device> Camera
<pixel_reading><patch> cyan
<pixel_reading><RGB resolution="8"><red>   0
<pixel_reading><RGB resolution="8"><green> 255
<pixel_reading><RGB resolution="8"><blue>  255
```

XML Design Pitfalls

Inadequate context describing what a data element is (incomplete use of tags)

Inadequate instructions on how to interpret data elements (incomplete use of attributes)

Use of attributes as data elements (improper use of attributes)

Use of data elements as metadata instead of using tags (indirection through use of name/value pairings)

Unnecessary, unrelated, or redundant tags (poor hierarchy construction)

Attributes that have nothing to do with data element interpretation (poor hierarchy construction or misuse of attributes)

Attributes as Data

```
<pixel_reading>
      <device>Camera</device>
      <patch>cyan</patch>
      <RGB resolution="8" red="0" green="255" blue="255">
</pixel_reading>
```

resolution=”8”

True attribute. The value “8” has no meaning by itself → metadata

red=”0”

Should be data: it’s a reading and to be interpreted requires resolution attribute

green=”255”

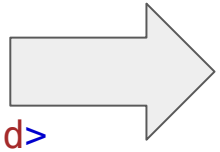
same as red

blue=”255”

same as red

Data as Metadata

```
<pixel_reading>
      <device>Camera</device>
      <patch>cyan</patch>
      <RGB>
         <item>
             <band>red</band>
             <value>0</value>
         </item>
         <item>
             <band>green</band>
             <value>255</value>
         </item>
         <item>
             <band>blue</band>
             <value>255</value>
         </item>
      </RGB>
</pixel_reading>
```



```
1 <pixel_reading><device>Camera
2 <pixel_reading><patch>cyan
3 <pixel_reading><RGB><item><band>red
4 <pixel_reading><RGB><item><value>0
5 <pixel_reading><RGB><item><band>green
6 <pixel_reading><RGB><item><value>255
7 <pixel_reading><RGB><item><band>blue
8 <pixel_reading><RGB><item><value>255
```

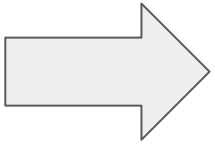
1, 2 → ok

3,5,7 → no data, just metadata of 4,6,8

4,6,8 → data but missing information (color?)

Inadequate Tags

```
<pixel_reading>
    <device>Camera</device>
    <patch>cyan</patch>
    <mode>RGB</mode>
    <band>red</band>
    <value>0</value>
    <band>green</band>
    <value>155</value>
    <band>blue</band>
    <value>255</value>
</pixel_reading>
```



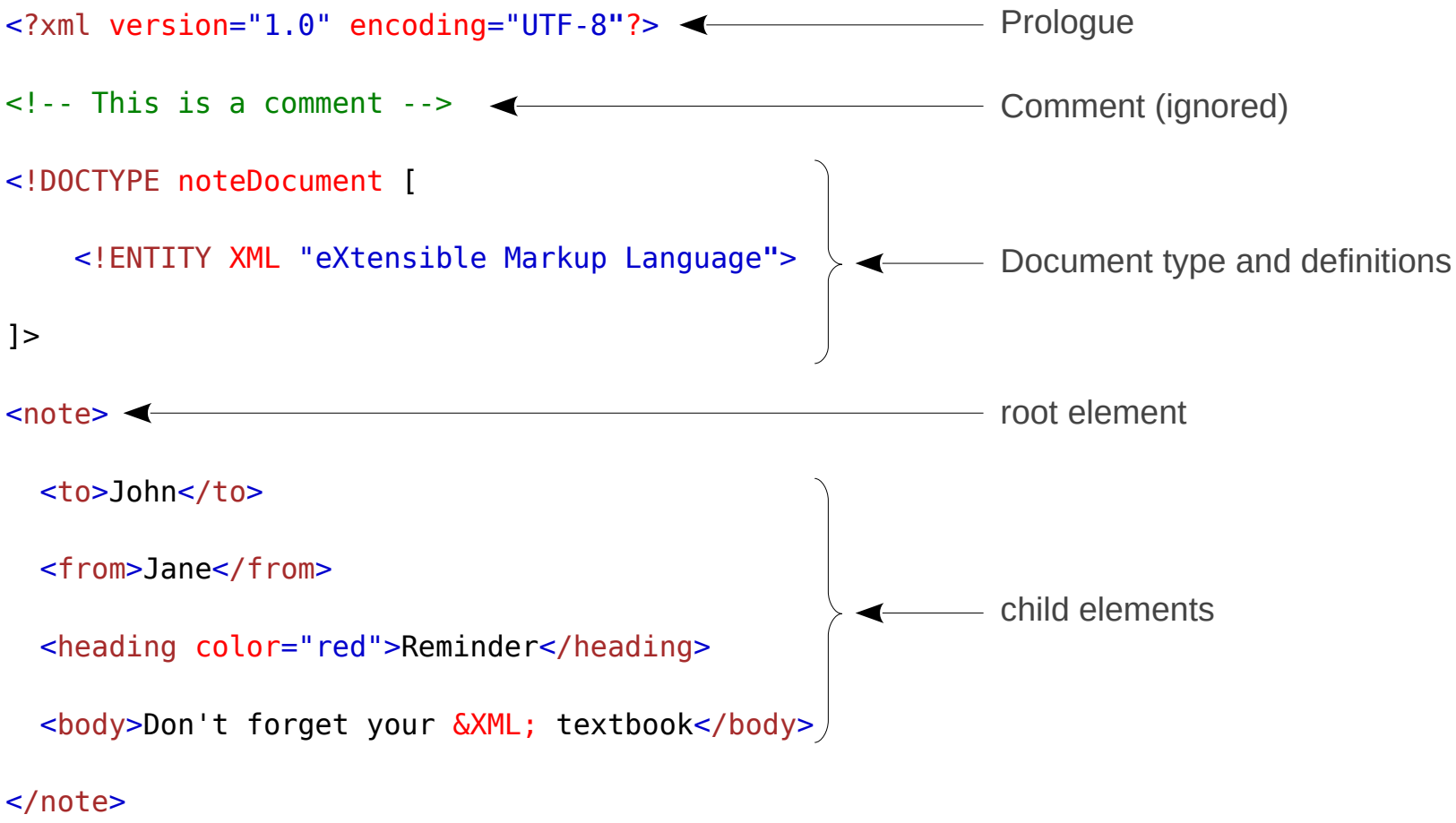
```
1 <pixel_reading><device>Camera
2 <pixel_reading><patch>cyan
3 <pixel_reading><mode>RGB
4 <pixel_reading><band>red
5 <pixel_reading><value>0
6 <pixel_reading><band>green
7 <pixel_reading><value>255
8 <pixel_reading><band>blue
9 <pixel_reading><value>255
```

1, 2 → ok

3,4,5,7 → no data, just metadata of 5,7,9

5,7,8 → data but missing information (color?)

XML file


<?xml version="1.0" encoding="UTF-8"?>
Prologue
<!-- This is a comment -->
Comment (ignored)
<!DOCTYPE noteDocument [
<!ENTITY XML "eXtensible Markup Language">
]>
Document type and definitions
<note>
root element
<to>John</to>
<from>Jane</from>
<heading color="red">Reminder</heading>
<body>Don't forget your &XML; textbook</body>
child elements
</note>