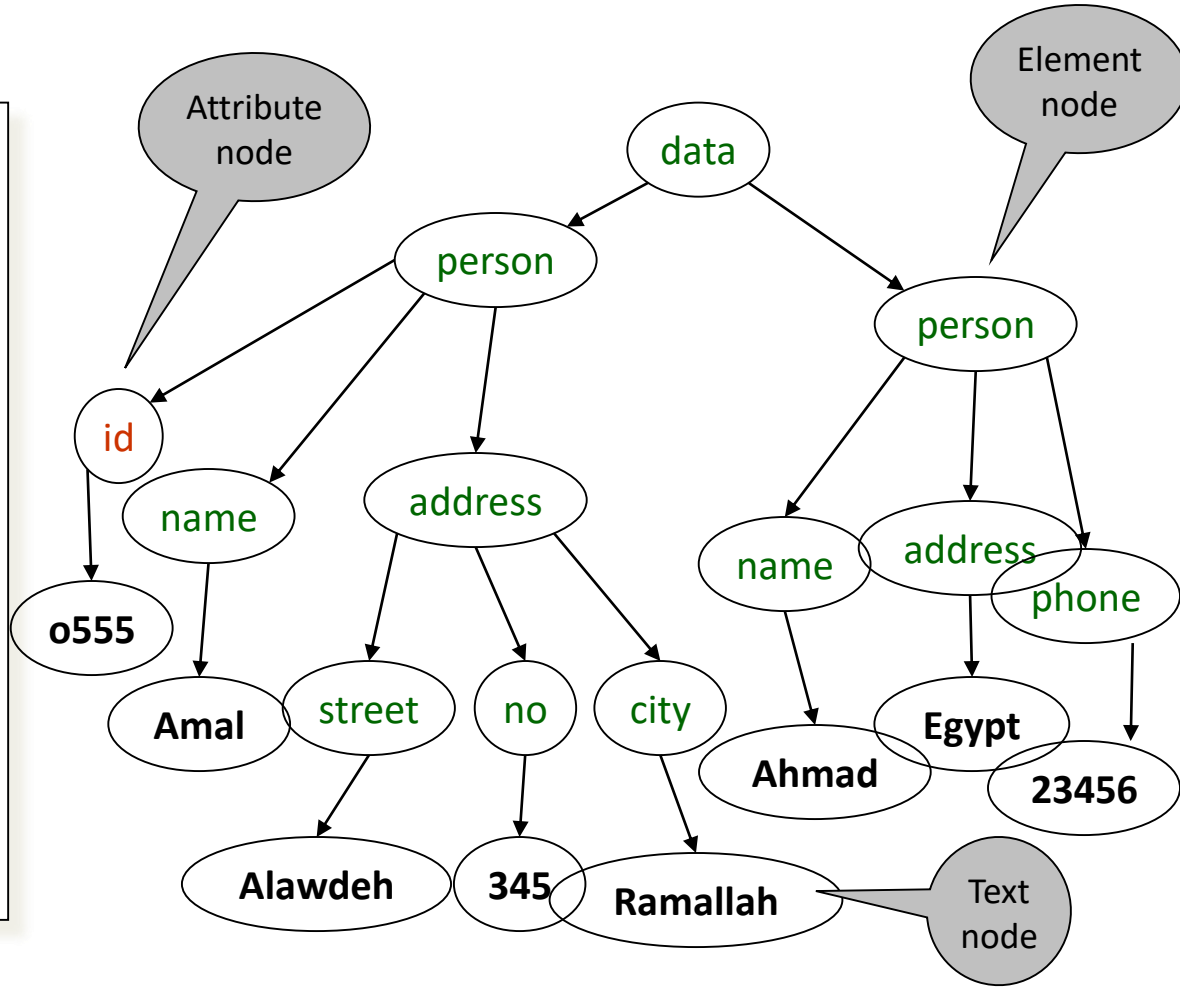
Document Type Definitions
DTD

XML Data: a Tree !

```
<data>
        <person id="o555" >
                <name> Amal </name>
                <address>
                        <street> Alawdeh </street>
                        <no> 345 </no>
                        <city> Ramallah </city>
                </address>
        </person>
        <person>
                <name> Ahmad </name>
                <address> Egypt  </address>
                <phone> 23456  </phone>
        </person>
</data>
```



Order matters !!!

Jason Formatter

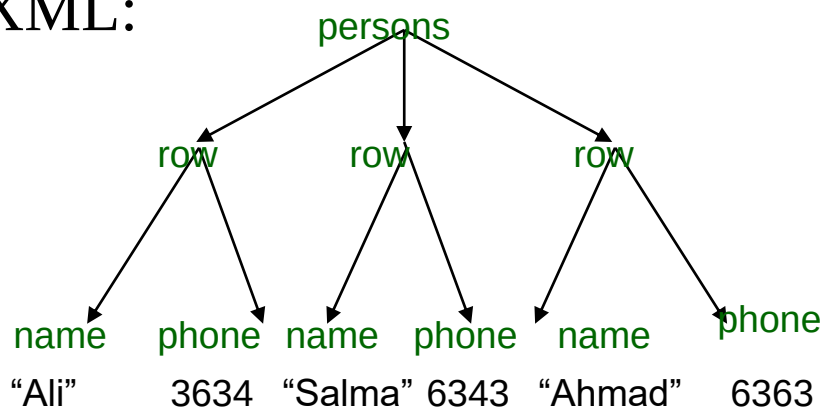
- https://jsonformatter.org/xml-viewer

From Relational Data to XML Data

persons

name	phone
Ali	3634
Salma	6343
Ahmad	6363

XML:




```
<persons>
    <row> <name>Ali</name>
          <phone> Ahmad</phone>
    </row>
     <row> <name>Salma</name>
          <phone> 6343</phone>
     <row> <name>Ahmad</name>
          <phone> 6363</phone>
    </row>
</persons>
```

Document Type Definitions
DTD

- Part of the original XML specification
- An XML document may have a DTD
- XML document:

  well-formed = if tags are correctly closed

  valid = if it has a DTD and conforms to it
- validation is useful in data exchange

Very Simple DTD

```
<!DOCTYPE company [
  <!ELEMENT company   ((person|product)*)>
  <!ELEMENT person  (ssn, name, office, phone?)>
  <!ELEMENT ssn        (#PCDATA)>
  <!ELEMENT name     (#PCDATA)>
  <!ELEMENT office     (#PCDATA)>
  <!ELEMENT phone    (#PCDATA)>
  <!ELEMENT product  (pid, name, description?)>
  <!ELEMENT pid    (#PCDATA)>
  <!ELEMENT description    (#PCDATA)>
]>
```

Very Simple DTD

Example of valid XML document:

```
<company>
   <person>  <ssn> 123456789 </ssn>
             <name> Waleed </name>
             <office> B432 </office>
             <phone> 1234 </phone>
   </person>
   <person>  <ssn> 987654321 </ssn>
             <name> Maher </name>
             <office> B123 </office>
   </person>
   <product> ...  </product>
   ...
</company>
```

What is a DTD

- Defines the structure of an XML document
- Only the elements defined in a DTD can be used in an XML document
- can be internal or external
- A DTD defines the structure of a “valid” XML document
- Processing overhead is incurred when validating XML with a DTD

An internal DTD

```
<?xml version=“1.0”?>
<!DOCTYPE invoice [
    <!ELEMENT  invoice (id, qty, desc, price) >
    <!ELEMENT  id      (#PCDATA) >
    <!ELEMENT  qty     (#PCDATA) >
    <!ELEMENT  desc    (#PCDATA) >
    <!ELEMENT  price   (#PCDATA) >
]>
        <invoice>
            <id>12345</id>
            <qty>55</qty>
            <desc>Left handed monkey wrench</desc>
            <price>14.95</price>
        </invoice>
```

An referenced external DTD

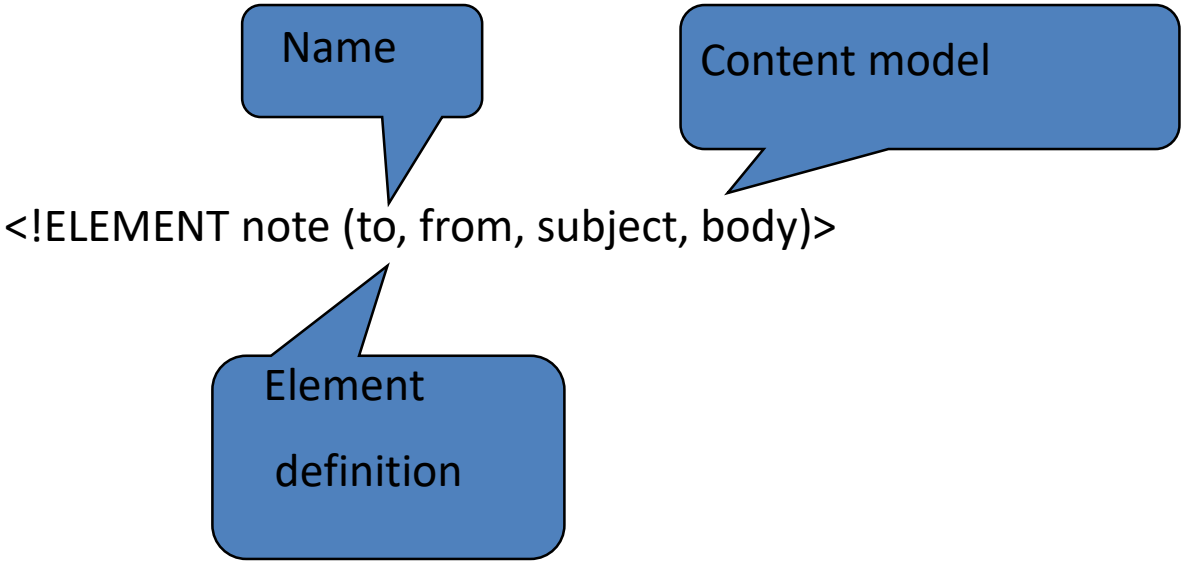
```
<?xml version=“1.0”>

<!DOCTYPE invoice SYSTEM “invoice.dtd”>

 <invoice>
   <id>12345</id>
   <qty>55</qty>
   <desc>Left handed monkey wrench</desc>
   <price>14.95</price>
 </invoice>
```

An external DTD (invoice.dtd)

```
<?xml version=“1.0”?>
<!ELEMENT  invoice (sku, qty, desc, price) >
    <!ELEMENT  id      (#PCDATA) >
    <!ELEMENT  qty      (#PCDATA) >
    <!ELEMENT  desc     (#PCDATA) >
    <!ELEMENT  price    (#PCDATA) >
```

Content Model

- Identify the name of the element and the nature of that element's content
- The example declares an element that then describes the document's content model



Element Type Declarations

- Three types of elements
  - EMPTY elements
  - ANY elements
  - MIXED elements

Empty Elements

- An element that can not contain any content
- The html image tag in xml would typically be empty, such as <image></image> or <image/>
- empty elements are more useful with the use of attributes

```
<!ELEMENT test EMPTY>

<!ELEMENT image EMPTY>

<!ELEMENT br EMPTY>
```

ANY Element

- An element that can contain any content
- it is recommended not to get into the habit declaring elements with the ANY keyword
- useful when transferring a lot of mixed or unknown data

```
<!ELEMENT  test  ANY >
```

Mixed Element

- Elements that can contain a set of content alternatives
- Separate the options with the “or” symbol “|”

```
<!ELEMENT  test   (#PCDATA) | name>
```

Data Types

- Parsed Character Data
  - #PCDATA
    - <!ELEMENT firstname (#PCDATA)>
    - <!ELEMENT lastname (#PCDATA)>

- Unparsed Character Data
  - CDATA
    - <firstname><![CDATA[Jim]]></firstname>
    - <lastname><![CDATA[Peters]]></lastname>

Structure Symbols

- Parenthesis (samp1, samp2) - The element must contain the sequence samp1 and samp2
- Comma (samp1,samp2,samp3) - The element must contain samp1,samp2 and samp3 in that order
- Or (samp1|samp2|samp3) - The element can contain samp1, samp2 or samp3
- ? samp1? - Element might contain samp1, if it does it can only do it once
- * samp1* - Element can contain samp1 one or more times
- + samp1+ - Element must contain samp1 at least once
- none samp1 - Element must contain samp1

Elements with more structure

<!ELEMENT email (to+ , from , subject? , body?)>

to: is reqd and can appear more than once

from: must appear only once

subject: optional, but if included can only appear once

body: optional, but if included can only appear once

Attribute Rules

- attribute values must be placed in “ “
  - in HTML this is only required id the attribute contains the space character
- attribute values are not processed by the XML parser
  - this means the values can’t be automatically checked by the parser

Attributes or Elements?

- Is it better to use attributes or to just make additional XML elements
  - there are no set rules when to use one over the other
    - experience is best teacher
  - but to help you decide:
    - attribute values are not parsed
      - can contain special characters that aren't allowed in elements
    - drawback - they cannot be validated by the parser
      - must be validated by additional code in the application

Attribute Defaults

- **#REQUIRED**
  - The attribute must have an explicitly specified value for every occurrence of the element in the document
- **#IMPLIED**
  - The attribute value is not required and no default value is provided. If a value is not specified the XMP processor must proceed without one.
- **“value”**
  - An attrubute can be given any legal value as a default. The attribute value is not required on each element of the document, and if it is not present it will appear to be the specified default
- **#FIXED “value”**
  - An attribute declaration may specify that an attribute has a fixed value. In this case, the attribute is not required, but if it occurrs, it must have the specified value. If it is not present, it will appear to be the specified defualt

A Code sample

```
<?xml version=“1.0” ?>
<!DOCTYPE email[
   <!ATTLIST email
         language  (english | french | spanish) “english”
        priority  (normal | high | low)  “normal” >
  <!ELEMENT   to          (#PCDATA)>
  <!ELEMENT   from       (#PCDATA) >
  <!ELEMENT    subject    (#PCDATA) >
  <!ELEMENT    message  (#PCDATA) > ]  >
<email  language=“spanish” priorit=“high”>
  <to>Peter Brenner</to>
  <from>Dick Steflik</from>
  <subject> Test Reminder</subject>
  <message>The exam is a week from today</message>
</email>
```

Excercise

- Setup a DTD to describe a course – course.dtd – consisting of the following elements:
- Root element: course
- To this:
  - name (course name) – 1 instance,
  - curriculum – 1 instance,
  - teacher – 1 to more instances,
  - Students – 1 to more instances.

- The DTD may look like:

<!--DTD to course at IHA-->

<!ELEMENT course (name, curriculum, teacher+, student+)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT curriculum (#PCDATA)>

<!ELEMENT teacher (#PCDATA)>

<!ELEMENT students (#PCDATA)>

Attribute Summary

- Attributes
  - cannot contain multipe values
  - cannot be validated
  - cannot describe structures like child elements can
- It is recommended to use attributes sparingly
- The following code would not be good form:

```
<?xml version="1.0" ?>
<email language="english" priority="high"
   to="you" from="me" subject="Reminder"
   message="The test is a week from today !" />
```

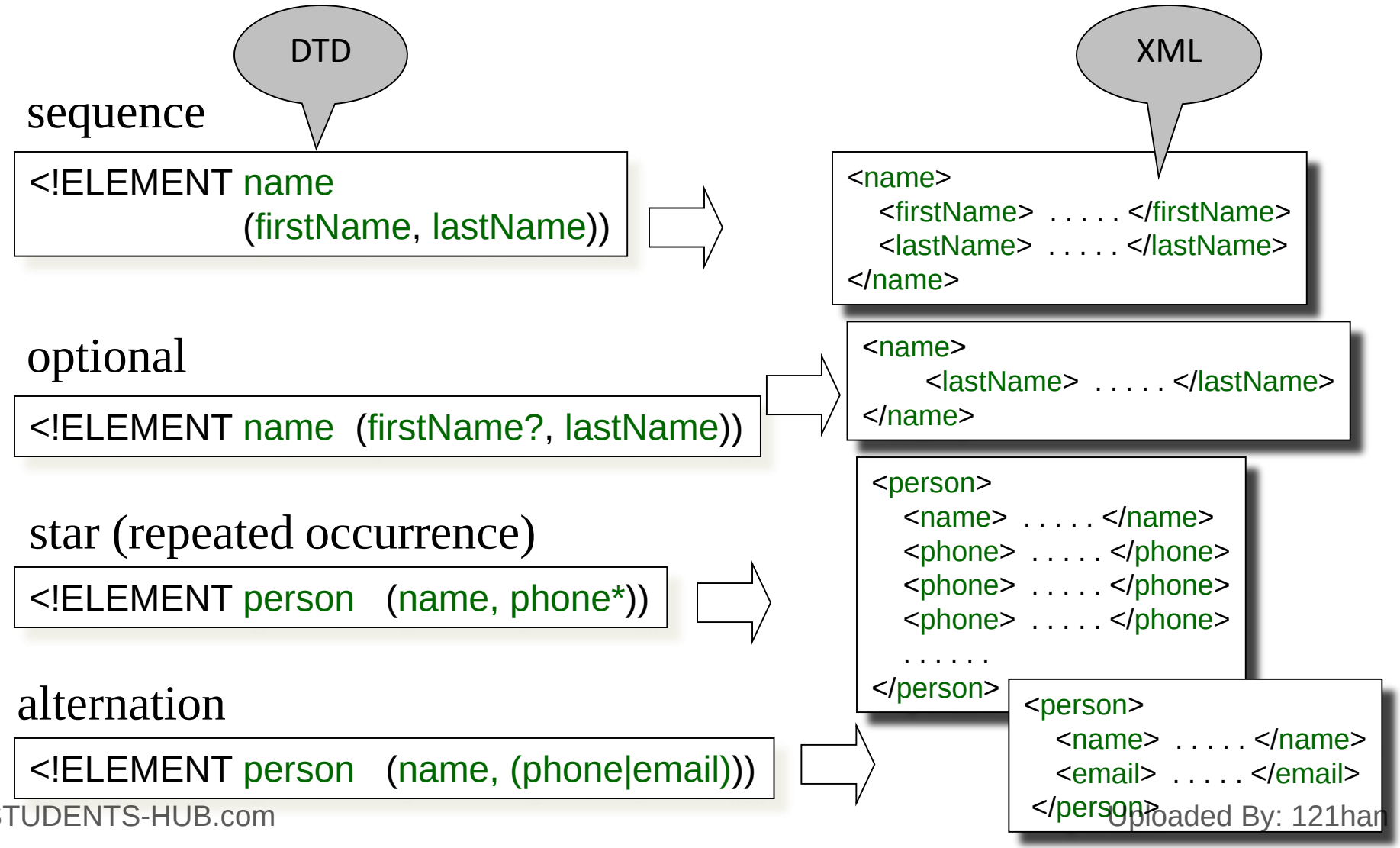
DTD: The Content Model

<!ELEMENT tag (*CONTENT*)>

content model

- Content model:
  - Complex = a regular expression over other elements
  - Text-only = #PCDATA
  - Empty = EMPTY
  - Any = ANY
  - Mixed content = (#PCDATA | A | B | C)*

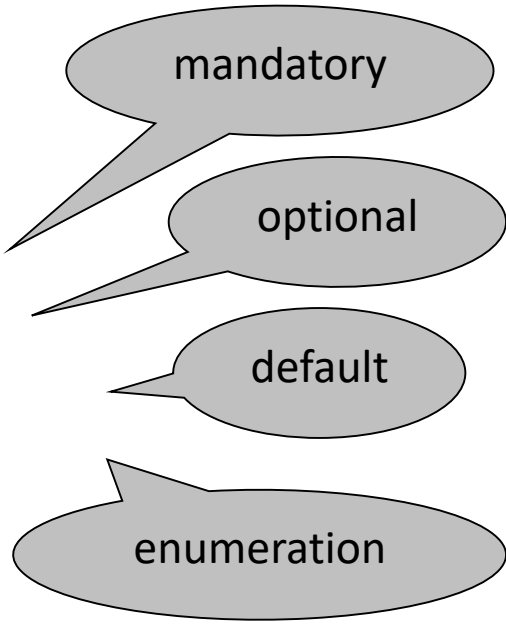
DTD: Regular Expressions



DTD: Attributes

- Document Type Definition

```
<!ELEMENT person (ssn, name, office, phone?)>
<!ATTLIST person  age CDATA #REQUIRED "18"
                  birthdate CDATA #IMPLIED
                  nationality CDATA #FIXED "CH"
                  gender (male|female) "female">
```



- Document

```
<person age="24" nationality="CH" gender="male">
  <ssn> … </ssn> …<phone> … </phone>
</person>
```