

## HTML Limitations

- Fixed set of tags
  - No standard way to create new formatting tags
- HTML falls short for other Web uses
  - Semantic meaning to content
    - Better search engines
  - Standard way to represent content/data
    - B2B transactions
    - AJAX
  - Standard way to invoke services
    - Encode function names and arguments

## XML (eXtensible Markup Language)

- Language for creating markup languages (Meta-language)
- Standard way to create new document types, tags, and attributes
- Specification available at:
  - <http://www.w3.org/TR/REC-xml>
- Supports all above new applications
- Examples:
  - XHTML
  - VoiceXML (for speech)
  - SOAP
  - WSDL

## XML Syntactic Elements

- Processing Instruction (PI)
 

```
Instructions for parser/ viewer/processor
<?xml .... ?>
<?xmlstylesheet type="text/css" href="resume.css"?>
```
- Elements
  - tags and attributes
 

```
<tagname attribute="value" ...>
</tagname>
```
- Entity reference
  - similar to macros
 

```
&nbsp;
```
- Comments
 

```
<!--This is a comment -->
```
- Doctype declaration
  - Identifies specific document class
 

```
<!DOCTYPE Catalog SYSTEM "cd.dtd">
```

## XML Syntax

- First line is XML declaration
 

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

 Nothing can precede this declaration, not even spaces
- All elements must have closing tags
 

```
<p>This is a paragraph</p> <br /> Empty element tag
```
- Tags are case sensitive
 

```
<a>Illegal</A> <a>Correct</a>
```
- Elements must be properly nested
 

```
<a><b>Illegal</a></b> <a><b>Correct</b></a>
```
- Must have one root element
  - Attribute values must always be quoted
 

```
<a sample=illegal> <a sample="correct">
```
- Example:
 

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<foo>
  <bar foobar="abc"> Some Content </bar>
</foo>
```

## Definition

- A XML document is “*well formed*” if it conforms to XML syntax rules.
- XML parser will recognize document as an XML instance.
- i.e., parser will not crash.
- An XML document is “*valid*” if it conforms to its declared schema.

## DTD (Document Type Declaration)

- Formal description of a particular document type
- Defines the legal building blocks of an XML document: **the document structure**
  - Names of elements, attributes, and entities
  - Where elements and attributes may occur
    - How elements fit together

## Definition

- A XML document is “*valid*” if it is well formed and conforms to its DTD
- No extra or missing tags
- All tags in order

## Advantages of XML Validation

## Advantages of XML Validation

- Can use a single parser for all XML documents
- Simplifies application writing as no need to handle syntactic or structural errors
- Independent programs can check document structure
- Allow standardized data exchange

## DTD Grammar

- !ELEMENT
- !ATTLIST
- !ENTITY

## !ELEMENT

- Declares a new element tag.
  - `<!ELEMENT tag-name content>`  
tag-name: has to start with letter or “\_” or “:”  
has no white spaces
  - content: EMPTY | #PCDATA | children | mixed | ANY
- #PCDATA is text that will be parsed. Tags inside the text will be treated as markup and entities will be expanded.
- children: [element[choice|sequence][?]\*+]  
choice: element [ element ]+  
sequence: element [, element ]+

## Examples

```
<!ELEMENT a EMPTY>
<a>Illegal</a>
<a></a>
<a />

<!ELEMENT name #PCDATA>
<name>John</name>

<!ELEMENT a (b,(c|d))+>
Assume b,c,d are all empty tags
illegal <a><b/><b/></a>
<a></a>
legal <a> <b/><c/><d/> <b/><d/> </a>

<!ELEMENT a (#PCDATA)a+>
Mixed example
<a> text <a>more</a> </a>
```

## !ATTLIST

- Associates name-value pairs with elements
- Attribute specifications may appear only within start-tags and empty-element tags

• <!ATTLIST tag-name att-def\*>  
 att-def: att-name att-type att-default  
 att-type: #CDATA | Character data *Tags inside the text will NOT be treated as markup, and entities will not be expanded.*  
           ID | Unique identified *Cannot start with digit "1"*  
           IDREF | ID of another element  
           NMTOKEN | Valid XML name (no spaces)  
           NMTOKENS | Space-separated list of names "blue red"  
           (en1 | en2| ..) Must be one from enumerated list  
  
 att-default: value | Default value  
               #REQUIRED |  
               #IMPLIED | Optional  
               #FIXED | value

## Examples

```

<!ELEMENT student EMPTY>
<!ATTLIST student
  id ID #REQUIRED
  name CDATA #REQUIRED
  department (ECE, CS) #IMPLIED CS>

<student id="s01" name="John" department="ECE" />
  
```

## !ENTITY

- <!ENTITY [%] name entity-value>

```

<!ENTITY threeOverFour "&#190;">
<!ENTITY % versionnumber "4.3.2.1">
<!ENTITY version "Version %versionnumber;">

<a> &version; and &threeOverFour; </a>
<a>Version 4.3.2.1 and ¾</a>
  
```

## !DOCTYPE

- Associates an XML document with a specific document class
- <!DOCTYPE rootelement PUBLIC[SYSTEM [name] URL]>

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">

rootelement = HTML
name = -//W3C//DTD HTML 4.01//EN
url= http://www.w3.org/TR/html4/strict.dtd

<!DOCTYPE Catalog SYSTEM "cd.dtd">

root element = Catalog
url= cd.dtd
  
```

## Example: XML email application

Write an XML application for representing an email. Include the following information: date, sender, destination, subject, and email body. No attachments.

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE email SYSTEM "email.dtd" >
<email date="September 20, 2004">
  <from>Jani</from>
  <to>Tove</to>
  <subject>Reminder</subject>
  <body>Don't forget me this weekend!</body>
</email>

<!ELEMENT email (from,to,subject,body)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT body (#PCDATA)>
<!ATTLIST email date #CDATA #REQUIRED>
  
```

## Example: XML library application

- Write an XML application for storing information about books in a library.
- Include the following information:
  - title
  - author
  - publisher
  - description
  - type (one of fiction,non-fiction,cookbook,technical)
  - ISBN
  - book id for internal purposes.
  - set of related books
  - flag the best of the related books

## library.dtd

```
<!ELEMENT library (book+)>
<!ELEMENT book (title, author+,publisher,description,related*)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author ((firstname, lastname)|(lastname, firstname))>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT related EMPTY>
<!ATTLIST book
  bookid ID #REQUIRED
  type (fiction|non-fiction|cookbooks|technical) #REQUIRED
  isbn CDATA #REQUIRED>
<!ATTLIST related bookref IDREF #REQUIRED>
<!ATTLIST related class CDATA #IMPLIED>
```

## Rules of Thumb

- Element
  - Data can be considered an independent object.
  - Data is related via a parent/child relationship to another piece of information.
  - Item needs to occur multiple times
  - Ordering is important
- Attribute
  - Information that describes other information, such as a status or id.
  - Limit values to a predefined list
  - Minimize the file size of target documents.
- More detail at <http://xml.coverpages.org/draft-stuhec-elemsattrib-03-20020316.pdf>

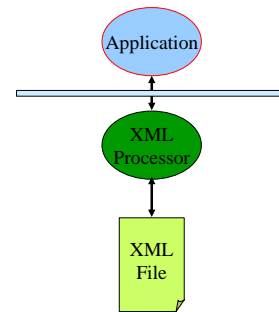
## XML Validation

Xerces validator

```
java -cp /u/csc309h/lib/xerces.jar:/u/csc309h/lib Validator -v email.xml
```

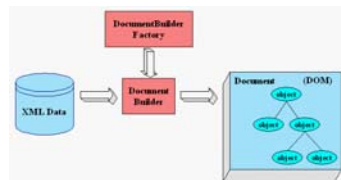
*-v is very important; otherwise it does not print any errors*

## XML Reference Architecture



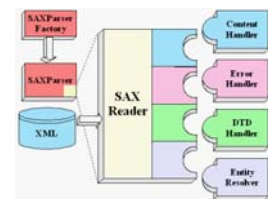
## DOM Sample

- DOM (Document Object Model)
  - Generates tree-like structure of XML document in memory
  - Programs transverse tree



## SAX

- SAX (Simple API for XML)
  - [www.saxproject.org](http://www.saxproject.org)
  - Event based
  - Trigger events as XML is parsed
  - Programs register for events



## Elements vs. Attributes

<b>Elements</b>	<b>Attributes</b>
Can have child Elements nested within them	Can contain only strings, or lists of strings
Structured and simple data	Used for "atomic" data items
Must appear in the order specified in the schema, but may appear several times.	Can only appear once in an element
Natural, core content, which would generally appear in every printout/display	Data of secondary importance; often metadata.
(Sub-)Elements represent parts of an Element	Properties of an Element