Lecture 7
XML

<?xml?>

The Course So Far

- We've talked about the *mechanics* of Web data exchange
  - TCP, HTTP
  - Dynamic content, sessions, logins
  - Security and Cryptography
- We're moving to *semantics* of data exchange
  - How do we interpret data?
  - What is a page about?
  - What is a site/page/file trying to say?

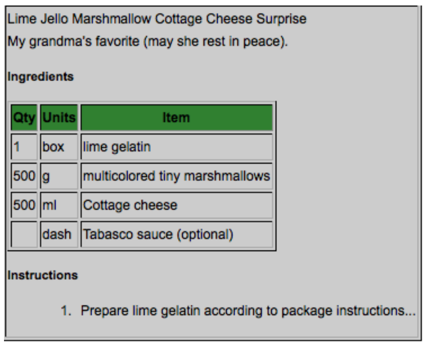
Data Exchange

- HTML is good for layout
  - (well, maybe not *terrible* for layout)
- But HTML entangles semantic content and layout

HTML Recipes

```
<!-- The original html recipe -->
<HTML> <HEAD>
<TITLE>Lime Jello Marshmallow…</TITLE> </HEAD>
<BODY> <H3>Lime Jello Marshmallow …</H3>
<H4>Ingredients</H4>
<TABLE BORDER="1">

<TR BGCOLOR="#308030">
<TH>Qty</TH><TH>Units</TH> <TH>Item</TH></TR>
  <TR><TD>1</TD><TD>box</TD><TD>lime
gelatin</TD></TR>
  …
</TABLE> <P> <H4>Instructions</H4>
<OL>
<LI>Prepare lime gelatin...</LI>
<!-- and so on -->
</BODY>
</HTML>
```

HTML Recipes



Lime Jello Marshmallow Cottage Cheese Surprise

My grandma's favorite (may she rest in peace).

Ingredients

Qty	Units	Item
1	box	lime gelatin
500	g	multicolored tiny marshmallows
500	ml	Cottage cheese
	dash	Tabasco sauce (optional)

Instructions

1. Prepare lime gelatin according to package instructions...

Data Exchange

- HTML is good for layout
  - (well, maybe not *terrible* for layout)
- But HTML entangles semantic content and layout
- One idea: markup for data, not display
  - Good for Info exchange, File serialization, Property lists
- Of course, many different kinds of data
  - <product>, <city>, <phonecall>, …
  - You can't include *all* of these items in one standard

XML Recipe Example

```
<?xml version="1.0"?>
<Recipe>
  <Name>Lime Jello Marshmallow Cottage
Cheese Surprise</Name>
  <Description> My grandma's favorite
(may she rest in peace).
  </Description>
  <Ingredients>
    <Ingredient>
      <Qty unit="box">1</Qty>
      <Item>lime gelatin</Item>
    </Ingredient>
  </Ingredients>
```

Data Exchange

- XML (eXtensible Markup Language) is a *metalanguage* for describing other formats
- Tries to handle general data exchange at several levels:
  - Character encoding (ASCII, Unicode, etc)
  - Syntax (tag names)
  - Structure (tag combination)
  - Semantics (what do tags mean?)

Sample XML

- `<article length="4055">`
  `Blah…`
  `</article>`

Start tag

Attribute

Item

End tag

- `<article length="0"></article>`
- `<article length="0"/>`

Empty tag

XML Formatting

- XML *well-formed* if syntax is right, incl:
  - Strictly hierarchical tag containment
  - No unclosed tags
  - Single root element
  - Lots of other syntactical things
- XML processors are generally strict
- So far, so good: we've got a nice standard for serializing data structures
  - BUT, your document can be well-formed XML and *still make no sense*

Class Exercise

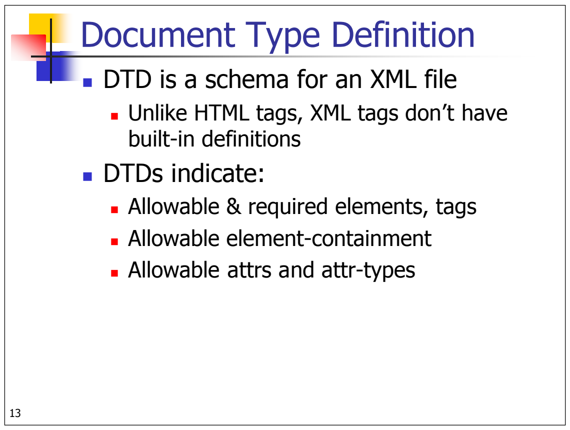
- Imagine you have just received the following XML file. How would you interpret it?

Class Exercise

- Imagine you have just received the following XML file. How would you interpret it?

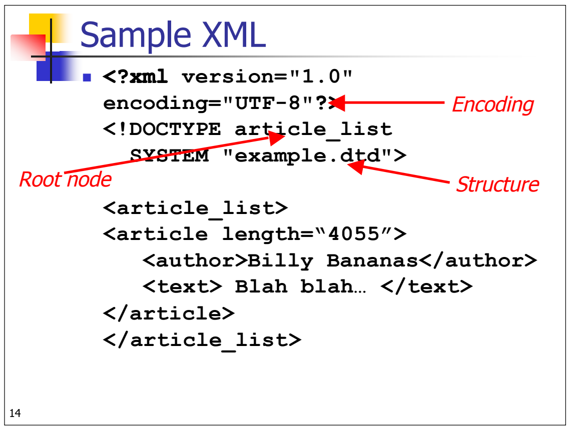
-
  ```
  <?xml version="1.0"?>
  <blimfark carmatomer="21">
     <glackett displacement="99">
       Bright Red
     </glackett>
     <transcrumble reggies="LR"/>
  </blimfark>
  ```
- Can't do it with the XML alone

–



Document Type Definition

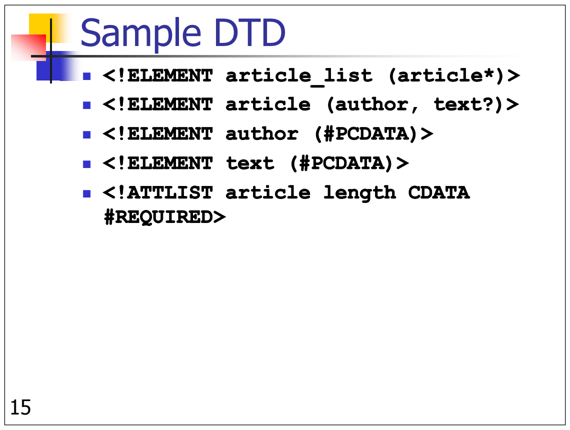
- DTD is a schema for an XML file
  - Unlike HTML tags, XML tags don't have built-in definitions
- DTDs indicate:
  - Allowable & required elements, tags
  - Allowable element-containment
  - Allowable attrs and attr-types



Sample XML

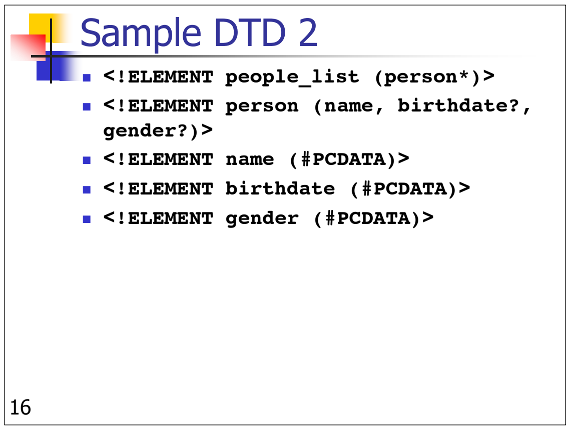
```
<?xml version="1.0"
encoding="UTF-8"?>
<!DOCTYPE article_list
  SYSTEM "example.dtd">

<article_list>
<article length="4055">
   <author>Billy Bananas</author>
   <text> Blah blah… </text>
</article>
</article_list>
```

Encoding
Root node
Structure



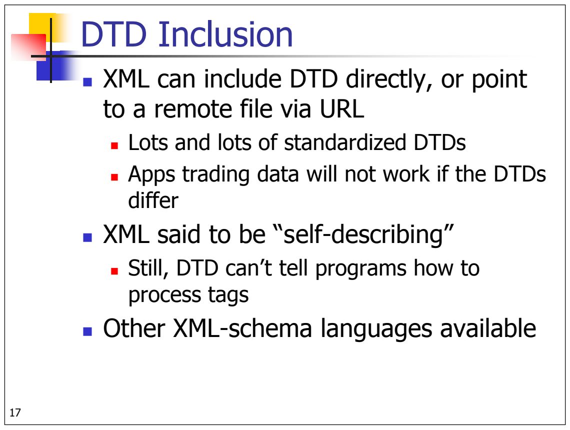
Sample DTD

- **<!ELEMENT article_list (article*)>**
- **<!ELEMENT article (author, text?)>**
- **<!ELEMENT author (#PCDATA)>**
- **<!ELEMENT text (#PCDATA)>**
- **<!ATTLIST article length CDATA #REQUIRED>**



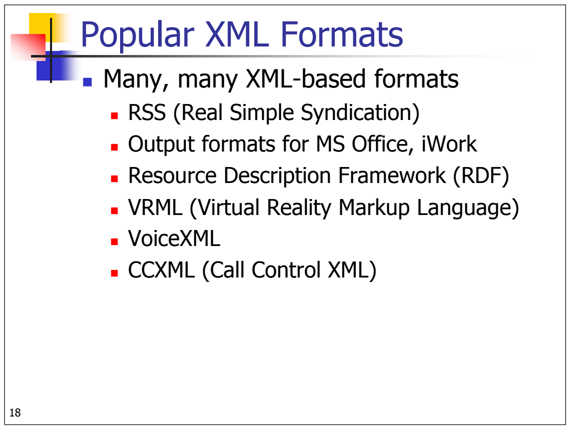
Sample DTD 2

- **<!ELEMENT people_list (person*)>**
- **<!ELEMENT person (name, birthdate?, gender?)>**
- **<!ELEMENT name (#PCDATA)>**
- **<!ELEMENT birthdate (#PCDATA)>**
- **<!ELEMENT gender (#PCDATA)>**



DTD Inclusion

- XML can include DTD directly, or point to a remote file via URL
  - Lots and lots of standardized DTDs
  - Apps trading data will not work if the DTDs differ
- XML said to be "self-describing"
  - Still, DTD can't tell programs how to process tags
- Other XML-schema languages available



Popular XML Formats

- Many, many XML-based formats
  - RSS (Real Simple Syndication)
  - Output formats for MS Office, iWork
  - Resource Description Framework (RDF)
  - VRML (Virtual Reality Markup Language)
  - VoiceXML
  - CCXML (Call Control XML)

XML Validation

- A *valid* XML document is well-formed and obeys DTD rules
  - The DTD constrains set of valid XML files
  - *Validating* the XML means testing whether it obeys the DTD rules
- XML parsers may or may not validate
  - HTML parsers (like browsers) are generally non-validating

19

Intermission: The Family Tree

- Generalized Markup Language (IBM ‘60s)
- Begat: Standard GML (ISO, 1986)
  - Used for technical publishing, document archives, military documents
  - E.g., all the docs for a 747
  - An instance of SGML: HTML 4 (earlier versions not quite compliant)
- Begat: XML (WWW, 1996)
  - Simplified SGML
- XHTML is HTML that is XML-compliant

20

Parsing XML

- Different motivations
  - Information exchange
  - Rendering information
  - Transforming information
- *Exchange* typically involves many distinct items
  - Shipping trade info to bank partner
- *Rendering* involves a few interrelated items
  - Drawing family tree to screen

21

XML Information Exchange

- Parsing should be fast & efficient
- Simple API for XML (SAX)
  - Single-pass, event-based parsing
  - Trigger callbacks as tags are encountered
  - No memory of past events/tags

22

XML Information Exchange

XML	*SAX Parser*
▪ <priceList>	▪ startElt()
▪ <coffee>	▪ startElt()
▪ <name> MochaJava </name>	▪ startElt() characters() endElt()
▪ <price> 11.95 </price>	▪ startElt() characters() endElt()
▪ </coffee>	▪ endElt()
▪ </pricelist>	▪ endElt()

23

XML Rendering Information

- Need to examine entire XML structure?
- Document Object Model (DOM)
  - Similar to JavaScript DOM access to page
  - Create in-memory tree structure that reflects XML file
  - Can be directly manipulated, edited

24

XML Parsing

- What's the best parser for…
  - Sensor updates
  - Medical record editor
  - Music library sync update

XML Transformation

- XSLT is **Xml Stylesheet Language for Transformations**
- Used to translate XML doc into a different format
  - XML into HTML or XHTML
  - Translation among XML schemas
  - Message filtering or editing
- XSLT xforms "src tree" into "result tree"

XSLT

- An XSLT program:
  - called a "stylesheet"
  - consists of "templates" that are matched (or not) by input XML elements
- If template matched, then xform rule fires; transformed elts sent to output
- A declarative prog language that is itself encoded in XML
  - XPath determines matches; more next time
  - Hard to read, if you ask me

XSLT: Input

```
<?xml version="1.0"
encoding="ISO-8859-1"?>
<catalog>
<cd>
  <title>EmpireBurlesque</title>
  <artist>Bob Dylan</artist>
  <country>USA</country>
  <company>Columbia</company>
  <price>10.90</price>
  <year>1985</year>
</cd>
</catalog>
```

XSLT: Stylesheet

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
  <body>
  …
  <xsl:for-each select="catalog/cd">
   <tr>
   <td><xsl:value-of select="title"/></td>
   <td><xsl:value-of select="artist"/></td></tr>
  </xsl:for-each>
  …
 </body></html>
</xsl:template>
</xsl:stylesheet>
```

Back to the Input

```
<?xml version="1.0"
encoding="ISO-8859-1"?>
<catalog>
<cd>
  <title>EmpireBurlesque</title>
  <artist>Bob Dylan</artist>
  <country>USA</country>
  <company>Columbia</company>
  <price>10.90</price>
  <year>1985</year>
</cd>
</catalog>
```

Back to the Input

```
<?xml version="1.0"
encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl"
href="cdcatalog.xsl"?>
<catalog>
<cd>
  <title>EmpireBurlesque</title>
  <artist>Bob Dylan</artist>
  <country>USA</country>
  <company>Columbia</company>
  <price>10.90</price>
  <year>1985</year>
</cd>
</catalog>
```

XML Data

- Used XML as transfer protocol so far
  - XML documents represent information
  - Most DBMSes follow **relational model**
- Could store data using **XML model** instead
  - No need to store for long periods
  - No need for queries or updates
  - No need to change structure over time
- For XML data model, we need it all
  - Extremely trendy, ~1996 - ~2003