Outline

• Why Structured Data?
• Types of Structured Data
  – XML and Friends
  – RDF and Semantic Web
Structured Data

• Overall Goal – add more “structure” to our otherwise freeform data

• Why?

• HTML/XHTML primarily designed to be rendered for __________ viewing. We want something that can be exploited by ____________.

XHTML → XML

```xml
<?xml version = "1.0"?>
<article>
  <title>Simple XML</title>
  <date>July 13, 2003</date>
  <author>
    <firstname>Carpenter</firstname>
    <lastname>Cal</lastname>
  </author>
  <summary>XML is pretty easy.</summary>
  <content>Once you have mastered XHTML, XML is easily learned. You must remember that XML is not for displaying information but for managing information.</content>
</article>
```
Another example

```xml
<xml version="1.0">
<!-- Fig. 20.4: article.xml -->
<!-- Article structured with XML -->
<article>
<!-- Simple XML files -->
<date>July 16, 2003</date>
<p>XML is pretty easy</p>
<p>Once you have mastered XHTML, XML is easily learned.
You must remember that XML is not for displaying information but
for managing information.</p>
</article>

<!-- Fig. 20.3: letter.xml -->
<!-- Business letter formatted with XML -->
<!DOCTYPE letter SYSTEM "letter.dtd">
<letter>
<name>John Doe</name>
<address1>123 Main St.</address1>
<address2></address2>
<city>Anytown</city>
<state>Anystate</state>
<zip>12345</zip>
<phone>555-1234</phone>
<flag gender="m"/>
</contact>

<contact type="to">
<name>Joe Schmo</name>
<address1>Box 12345</address1>
<address2>15 Any Ave.</address2>
<city>Othertown</city>
</contact>
```
Document Type Definitions

```xml
<!-- Fig. 20.6: letter.dtd -->
<!-- DTD document for letter.xml -->

4
5
element letter ( contacts, salutation, paragraphs, 
   closing, signature );

6
7
element contact ( name, address1, address2, city, state, 
   zip, phone, flag );

8
9
<!ELEMENT contact type CDATA #IMPLIED >
10
11
element name ( #PCDATA )
12
element address1 ( #PCDATA )
13
element address2 ( #PCDATA )
14
element city ( #PCDATA )
15
element state ( #PCDATA )
16
17
element zip ( #PCDATA )
18
20
21
element salutation ( #PCDATA )
22
23
element closing ( #PCDATA )
24
element paragraph ( #PCDATA )
25
element signature ( #PCDATA )
```
Additional XML Technology

- XML Schema – like DTD, but in XML and more sophisticated
- XPath – way of identifying a particular element inside a document
- XSL – render XML for viewing in browser
- XSLT – transform one XML doc to another

What’s Missing With XML?

- XML is centered around documents
  - DTDs/Schemas say what is legal
  - Software can use documents with DTDs it “knows about”
- RDF is centered around statements
  - Statement is (subject, predicate, object)
  - Example: ( <it350> , location, “MI 223” )
  - Example: ( <it350> , type, course )
  - XML-based syntax:
    <course about= http://www.cs.usna.edu/~it350/F06 >
    <location>MI 223</location> 
    </course>
  - RDF =
More Complete RDF Example

```xml
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns="http://www.cs.washington.edu/semweb/rdf-equiv-ns#">
  <rdf:Description>
    <course>
      <rdf:Description>
        <name>Networking Seminar</name>
        <time>1 p.m.</time>
        <location>Sieg 134</location>
        <instructor>
          <rdf:Description>
            <value>Prof. John Fitz</value>
            <office hours>Tue 3-4 p.m.</office hours>
          </rdf:Description>
        </instructor>
        <instructor>
          <rdf:Description>
            <value>Prof. Helen Randolph</value>
            <office hours>Fri 9-10 a.m.</office hours>
          </rdf:Description>
        </instructor>
        ...  
      </course>
  </rdf:Description>
</rdf:RDF>
```

**RDF → Semantic Web?**

- **XML forces nesting of tags in certain order**
  - Vocabulary and structure fixed by the DTD
- **With RDF, can place statements anywhere, in any order**
  - And can mix vocabulary (types, predicates) from many different documents
- **RDF Schema lets us define different schemas for different domains**
  - University domain, sailing domain, animal domain, …
  - Software that understands `<sailing:knots>` can utilize this info in any document that it finds it
  - Crawlers can collect knowledge → RDF databases
- **Semantic == meaning**
  - “Semantic Web” – collection of data for which software can “know” meaning of many of the terms – and exploit this
  - Aggregate data from many different sites (prices, ratings)
  - Search for info where answer not found on any one page
  - Personal agents to reason about such info to buy/schedule/search for you
Example Semantic Services – Calendar


Example Semantic Services – “Who’s Who”

(see http://www.cs.washington.edu/research/semweb/index.html)
Example Semantic Services – “Flink”

(see http://flink.semanticweb.org/index.jsp)

Conclusion

- HTML/XHTML has humans in mind
- Both XML and RDF add _____________ to data, making more amenable to _____________ interpretation
- Any one application could always just use XML with a fixed Schema
  - But RDF allows a much more flexible approach
  - Hopefully enabling data to be re-used in unexpected ways
  - “Semantic Web” is all about making smart apps to exploit this data
- Key Challenge – who will make the RDF?
  - Idea 1 – make it rewarding for normal people
  - Idea 2 – smart software to “scrape” it off the web