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"?>
<doc>
<!-- Fig. 20.1: article.xml -->
<!-- Article structured with XML -->
<article>
  <title>Simple XML</title>
  <date>July 13, 2001</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. 20.1: article.xml -->
  <! Article structured with XML -->
  <article>
    <div>
      <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>
    </div>
  </article>
</xml>

```

```xml
<xml version="1.0">
  <! Fig. 20.3: letter.xml -->
  <! Business letter formatted with XML -->
  <!DOCTYPE letter SYSTEM "letter.dtd">
  <letter>
    <contact type = "From">
      <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 Schmoe</name>
      <address1>Box 12345</address1>
      <address2>15 Any Ave.</address2>
      <city>Other town</city>
```

```xml
```
Document Type Definitions

```xml
1  <--- Fig. 20.6: letter.dtd  --->
2  <--- DTD document for letter.xml --->
3
4  <!ELEMENT letter (contacts, salutation, paragraphs, closing, signature)>
5
6  <!ELEMENT contact (name, address1, address2, city, state, zip, phone, flag)>
7  <!ATTLIST contact type CDATA #IMPLIED>
8
9  <!ELEMENT name (PCDATA)>
10  <!ELEMENT address1 (PCDATA)>
11  <!ELEMENT address2 (PCDATA)>
12  <!ELEMENT city (PCDATA)>
13  <!ELEMENT state (PCDATA)>
14  <!ELEMENT zip (PCDATA)>
15  <!ELEMENT phone (PCDATA)>
16  <!ELEMENT flag EMPTY>
17  <!ATTLIST flag gender (m | f) "m">
18
19  <!ELEMENT salutation (PCDATA)>
20  <!ELEMENT closing (PCDATA)>
21  <!ELEMENT paragraph (PCDATA)>
22  <!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, “MI200”)  
  - Example: ( <it350>, type, course )
  - XML-based syntax:
    ```xml
    <course about= http://www.cs.usna.edu/~it350/F05 >
    <location>Sieg 134</location>
    </course>
    ```
  - RDF =
More Complete RDF Example

```xml
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-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>
      </rdf:Description>
    </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)
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