XML
(eXtensible Markup Language)

CS 4640
Programming Languages
for Web Applications

[Robert W. Sebesta, "Programming the World Wide Web]
[http://www.w3.org/XML/]

Overview

1. What is XML?
2. Why XML?
3. How does XML work?
4. Syntax of XML documents

What is XML?

- e**X**tensible **M**arkup **L**anguage
- Markup languages insert "tags" into text files to describe presentation or other information
  - Human- and machine-readable
- SGML: Standard Generalized Markup Language
  - HTML: visual presentation
  - Latex: document formatting
  - XML: data description
- Structure, store, and transport data over the Internet
- W3C standard: http://www.w3.org/XML/

Why XML?

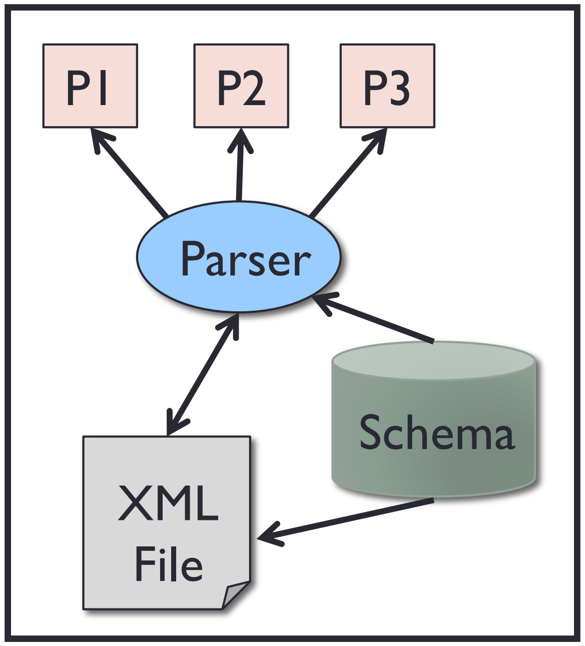
- Parsing data from one software component to another has always been difficult
- The two components must agree on format, types, and organization
- Web apps have unique requirements for data passing
  - Very loose coupling
  - Dynamic integration
- XML provides a way to separate data from the format

Why XML? – 5 Basic Reasons

- Simplicity
  - User-defined tags, easy to understand
- Organization
  - Organize data in one resource and formatting rules in another resource
- Accessibility
  - Save time and easy to change data (because of the separation)
- Standardization
  - XML is an international standard – easy to distribute data over the Internet
- Multiple applications
  - XML data resource can easily be reused to generate different views (promoting MVC)

Passing Data with XML

- Data are passed directly between components
- XML allows for self-documenting data



- P1, P2 and P3 can see the format, contents, and structure of the data
- Free parsers are available to put XML messages into a standard format
- Information about type and format is readily available

```
<customer>
  <number>12345</number>
  <name>Mary Kay</name>
  <address>…..</address>
    …
</customer>
```

```
<cust>
  <custname>Duh Huh</custname>
  <custID>12345</custID>
  <addr>…..</addr>
    …
</customer>
```

How does XML work?

- Programmers can create their own tags
- Tags have been designed for mathematics, formal specifications, resumes, recipes, addresses, …
- Pizza Markup Language (PML):

```
<pizza>
   <topping extracheese="yes">Pepperoni</topping>
   <price> 13.00 </price>
   <size> large </size>
</pizza>
```

- Event Markup Language (EML) ?
- Invitation Markup Language (IML) ?
- Pirate Markup Language (PiML) ?

Markup Languages – Type setting

- Documents were marked-up to represent how they would be printed
- For example, words can be **Bold**, *italicized*, or <u>underlined</u>
- Typesetting only effects the printing of specific phrases or words, and not categories of phrases or words

Markup Languages – Semantic Tags

- Markup languages can be used to logically organize the contents of a document
- For example, a document representing a book can contain the following organizational tags:
  - Title
  - Chapter headings
  - Section headings

Markup Languages – Semantic Tags

- A markup language can also provide semantic information (*meta-data*) about the text in a document
  - Examples : *First name, Last name, Phone number*

- Semantic tags can improve the accuracy of document queries
  - Documents can be searched using their tag assignments rather than the plain-text contents

Markup Languages – Semantic Tags

- Use semantic tags to define the hierarchical structure of the document
  - Author
    - First name
    - Last name
  - Publisher
    - Name
    - Address

Markup Languages – Examples

- Typesetting tags

  <bold> **Chapter 1** *</bold>*

  <italic> Background </italic>

  <underline> Important text *</underline>*

- Semantic tags

  <first name> Upsorn </first name>

  <last name> Praphamontripong </last name>

  <phone number> 434-123-1234 </phone number>

SGML (Standard Generalized Markup Language)

- Set up by the ISO in 1986
- Super set of all markup languages
  - Includes all the features of every markup language derived from it
- Allows a document to be annotated with text that describes the semantic meanings of portions of the document
- Separates the structure of the document from the content
  - The structure denotes the purpose of the document's data
- Use grammars (schemas and DTDs) to define the syntax of the annotations used in a document
- Captures meta-data for a document by marking up the content

Characteristics of XML

1. XML is extensible
   - Tags have been designed for mathematics, format specification, resumes, recipes, addresses, pizza, …
2. XML has a strict structure
3. XML is validating
   - Grammars (schemas and DTDs) define XML languages
   - Documents can be checked against the grammar
   - Allows programs to assume the data is formatted correctly, reducing the amount of checking the program must do

XML Provides Data Independence

- Allows data to be used by any application
- Requires every document to be in a clear and specific format
- Fosters information sharing better than other markup languages

XML Simplifies Data Sharing

- Plain text
  - Create and edit files with any editor
  - Easy to debug
  - Scalability : suitable for both small and large scaled data
- Data identification
  - Once different parts of the information have been identified, they can be used in different ways by different applications
- Data transference
  - Very easy to move between XML and form parameters
  - Very easy to move between XML and databases

XML Example: Message

```
<message>
    <to> you@yourAddress.com </to>
    <from> me@myAddress.com </from>
    <subject> XML Is Really Cool </subject>
    <text>
        How many ways is XML cool? Let me count the ways ...
    </text>
 </message>
```

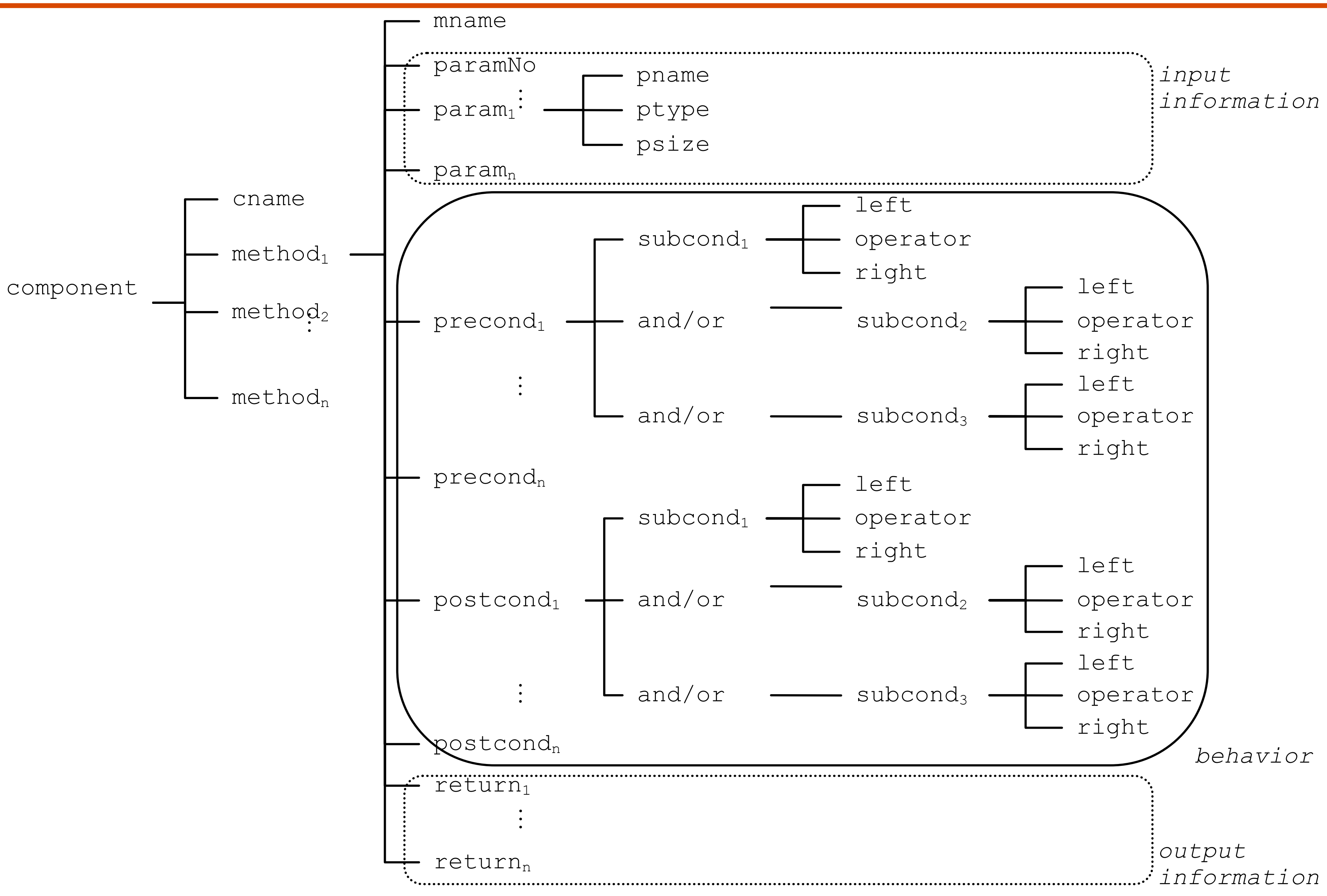
Another Example: Software Library

```
<library>
  <component>
    <cname> simple_list </cname>
    <method>
      <mname> create </mname>
      <paramNo> 1 </paramNo>
      <param>
        <pname> L </pname>
        <ptype> list </ptype>
      </param>
      <postcond>
        <operator> exist </operator>
        <right> L </right>
        <and>
          <subcond>
            <left> L </left>
            <operator> is </operator>
            <right> empty </right>
          </subcond>
        </and>
      </postcond>
      <return> none </return>
    </method>
```

```
<method>
      <mname> clear </mname>
      <paramNo> 1 </paramNo>
      <param>
        <pname> L </pname>
        <ptype> list </ptype>
      </param>
      <precond>
        <operator> exist </operator>
        <right> L </right>
      </precond>
      <postcond>
        <left> L </left>
        <operator> is </operator>
        <right> empty </right>
      </postcond>
      <return> none </return>
    </method>

…
</library>
```

Another Example: Component Spec



[XML-based software component retrieval, U. Praphamontripong and H. Gongzhu]

Another Example: Component Spec

```
<component>
   <cname> component_name </cname>
   <method>
        <mname> method_name₁ </mname>
        <paramNo> no_of_parameter </paramNo>
        <param>
           <pname> parameter_name₁ </pname>
           <ptype> parameter_type₁ </ptype>
           <psize> parameter_size₁ </psize>
        </param>
        …
        <precond>
           <weight> weight₁ </weight>
           <left> left_operand₁ </left>
           <operator> operator₁ </operator>
           <right> right_operand₁ </right>
        </precond>
        <precond>
           <weight> weight₂ </weight>
           <left> left_operand₂ </left>
           <operator> operator₂ </operator>
           <right> right_operand₂ </right>
        </precond>
        …
        <postcond>
           <weight> weight₁ </weight>
           <left> left_operand₁ </left>
           <operator> operator₁ </operator>
           <right> right_operand₁ </right>
        </postcond>
        <postcond>
           <weight> weight₂ </weight>
           <left> left_operand₂ </left>
           <operator> operator₂ </operator>
           <right> right_operand₂ </right>
        </postcond>
        …
        <return> return_type </return>
   </method>
        …
</component>
```

[XML-based software component retrieval, U. Praphamontripong and H. Gongzhu]

XML Structure

- Containment: Tags can be contained in other tags
- Tag names should be meaningful
- All tags must have an end tag
  - Note that HTML does not (i.e., HTML is not fully SGML-compliant)

XML Can Easily Be Validated

- XML messages are described in grammars
- Two ways to describe an XML language
  - Schemas : Grammar plus types and facets
  - Document Type Definitions (DTD) : Older, easier to read and understand, but somewhat limited
- Documents can be checked against the grammar
- Grammar can specify that certain fields are required
- Allows programs to assume the data is formatted correctly, reducing the amount of checking the program must do

Syntax of XML

- XML syntax is defined at two levels
  - General syntax : defines syntax on all XML documents
    - Correct documents said to be "well formed"
  - Specific syntax : defines syntax on a specific group of documents
    - Correct documents said to be "valid"
- Statements in an XML document
  - XML declaration – which version of XML
  - Data elements – the primary contents of the document
  - Markup declarations – instructions to XML parser
  - Processing instructions – instructions to the program

Well formed
→ adheres to the XML standard (syntax)

Valid
→ adhere to a DTD or schema (semantics)

XML Declaration

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

required

optional

- *version*
  - Identifies the version of the XML markup language used in the data
  - This attribute is required
- *encoding*
  - Identifies the character set used to encode the data
  - "ISO-8859-1" is "Latin-1" the Western European and English language character set
  - Default is compressed Unicode: UTF-8
- *standalone*
  - Tells whether or not this document references an external entity or an external data type specification
  - If there are no external references, use "yes"

XML Data Element (or Tag) Names

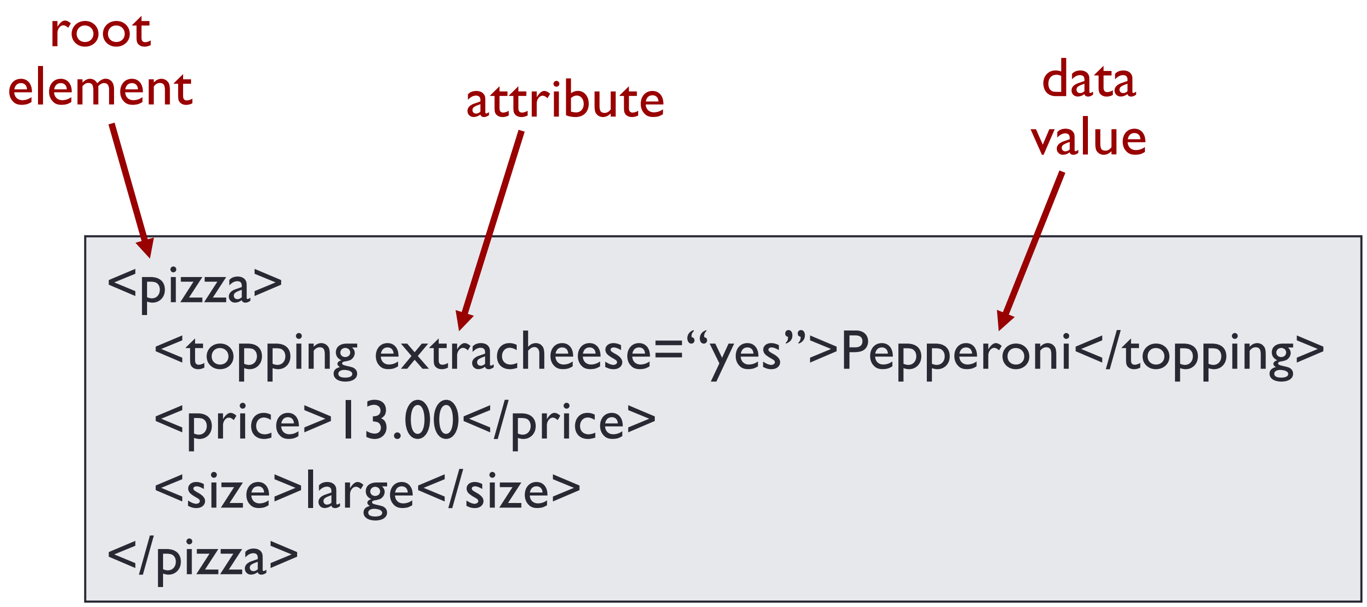
- Must start with a letter or underscore, and can include digits, hyphens, and periods
- XML names are case sensitive
  - lastName, lastname, LASTNAME are all different

XML General Syntax Rules “Well-formed”

- Every XML document has a single root element
  - Opening tag must be first line of XML
  - All other elements are nested inside the root element
- XML tags are surrounded by pointy brackets “< >”
- Every XML tag must have a closing tag
  - If no content: <empty/>
- XML elements must be properly nested
  - <B><I> … </B></I> is **not well formed** XML
- All attribute values must be enclosed in quotes

XML Example

Pizza Markup Language (PML)



```
<pizza>
   <topping extracheese="yes">Pepperoni</topping>
   <price>13.00</price>
   <size>large</size>
</pizza>
```

Attributes vs. Nested Tags

- In PML, “extraCheese” could have been defined as attribute or a nested tag
- Images can only be attributes
- It is easier to add new tags than attributes
- Attributes cannot define structure

Attribute

```
<… name=“Yao Ming”>
```

Nested Tags

```
<name>
      <familyName>Ming</familyName>
      <givenName>Yao</givenName>
</name>
```

Attributes vs. Nested Tags (2)

- Attributes are necessary when:
  - Identifying numbers or names of elements
  - Values are selected from a finite set
- Attributes should be used when:
  - No substructure
  - Attribute describes information about the element

XML Entity References (Variables)

- Entities are usually used to embed special characters into XML messages
- *Document Entity* : The file that represents the document
- Other entities have names
- Entity names start with letters, dash, colon
  - Can also contain digits, periods, underscores
- References to entities surround name with &;
  - &entityName;
- Some built-in XML entities: &lt; &gt; &amp; &quot; &apos;
- Use entities to avoid malformed XML

  <pred> X < Y </pred> … <pred> X &lt; Y </pred>

XML vs. HTML

- Unlike HTML, XML tags tell you what the data means, rather than how to display it
- XML elements must be strictly nested, XML can represent data in any level of complexity
- Both XML and HTML allow empty tags; in XML an empty tag must be followed by a forward slash: <emptyTag />
- XML attribute values must be surrounded by single or double quotes but HTML does not require quotes for single values
- XML tags are case sensitive but HTML tags are not

Summary

- XML gives software engineers an incredibly flexible, simple, and powerful way to represent data
  - Works with all sorts of data
  - Maps naturally to tables, spreadsheets and databases
- Grammatical rules can be defined
- Well formed XML may not be valid
- Valid XML is well formed XML
- Human readable
- Performance costs
  - Plain text files use more space on disk
  - Takes time to read, write, and reformat XML to and from internal representations
  - This cost is seldom important and almost never within web applications