The eXtensible Markup Language (XML)

Download Notepad++

- https://notepad-plus-plus.org/download/v7.6.6.html

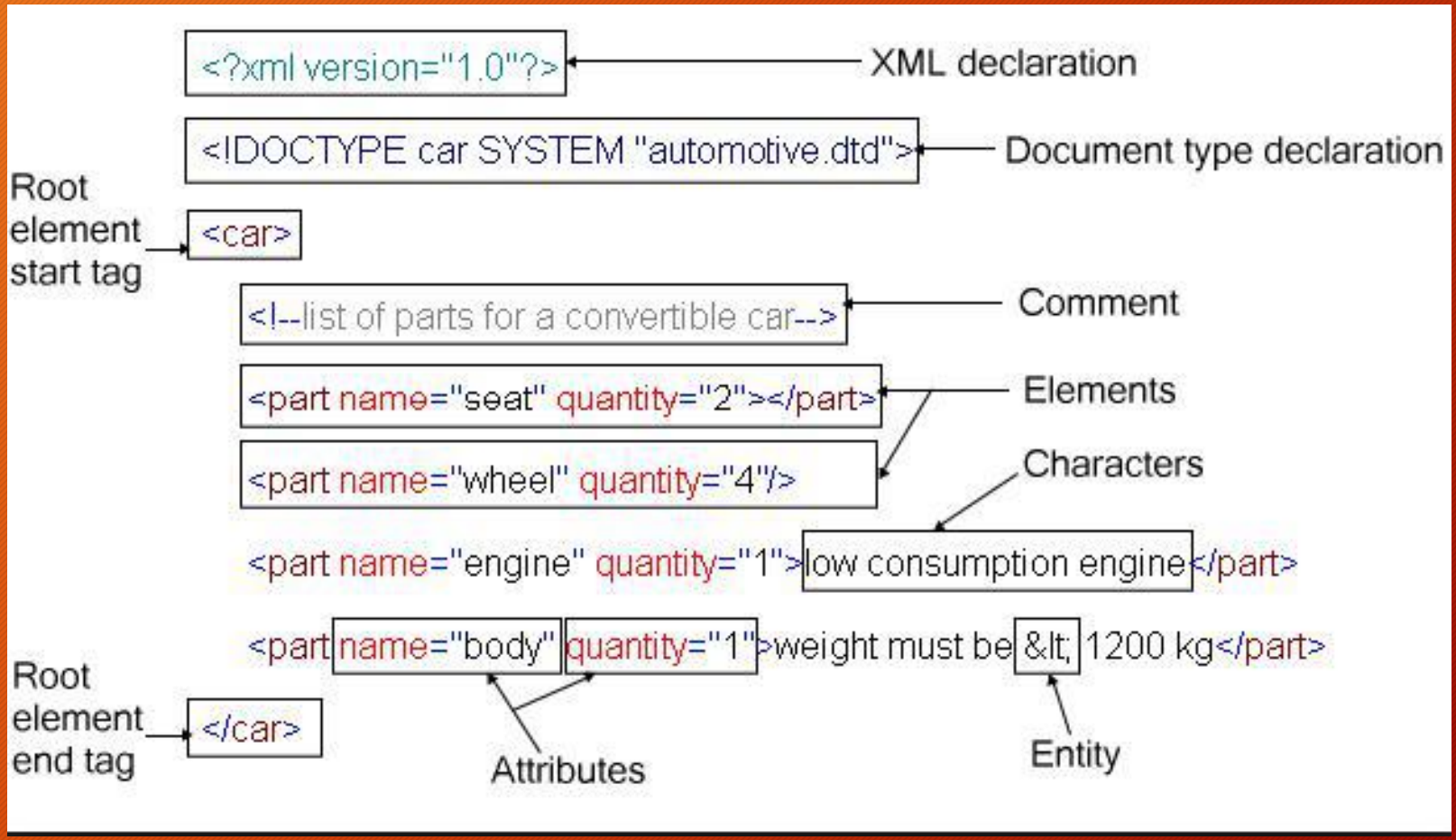
Presentation Outline

- Part 1: The basics of creating an XML document
- Part 2: Developing constraints for a well formed XML document
- Part 3: XML and supplementary technologies

Part 1: Background for XML

- An Extensible Markup Language (XML) document describes the *structure of data*
- XML and HTML have a similar syntax … both derived from SGML
- XML has no mechanism to specify the format for presenting data to the user
- An XML document resides in its own file with an '.xml' extension

Main Components of an XML Document



An Example XML Document

- An example of an well-commented XML document
- In Notepad ++ Use launch -> Run in Chrome

```
▼<breakfast_menu>
  ▼<food>
      <name>Belgian Waffles</name>
      <price>$5.95</price>
    ▼<description>
        Two of our famous Belgian Waffles with plenty of real maple syrup
      </description>
      <calories>650</calories>
    </food>
  ▼<food>
      <name>Strawberry Belgian Waffles</name>
      <price>$7.95</price>
    ▼<description>
        Light Belgian waffles covered with strawberries and whipped cream
      </description>
      <calories>900</calories>
    </food>
  ▼<food>
      <name>Berry-Berry Belgian Waffles</name>
      <price>$8.95</price>
    ▼<description>
        Belgian waffles covered with assorted fresh berries and whipped cream
      </description>
      <calories>900</calories>
    </food>
  ▼<food>
      <name>French Toast</name>
      <price>$4.50</price>
    ▼<description>
        Thick slices made from our homemade sourdough bread
      </description>
      <calories>600</calories>
```

The Basic Rules

- XML is case sensitive
- All start tags must have end tags
- Elements must be properly nested
- XML declaration is the first statement
- Every document must contain a root element
- Attribute values must have quotation marks
- Certain characters are reserved for parsing

Common Errors for Element Naming

- Do not use white space when creating names for elements
- Element names cannot begin with a digit, although names can contain digits
- Only certain punctuation allowed – periods, colons, and hyphens

Walking through an Example

- Modify the new2.xml document
- Add a new root “Home Delivery”
- Add a new element “Available”
- Add values as “Yes” or “No”
- Add a comment once before new root “Home delivery” & comment “Edited on <date> by<your Name>

Part 2: Legal Building Blocks of XML

- A Document Type Definition (**DTD**) allows the developer to create a set of rules to specify legal content and place restrictions on an XML file
- If the XML document does not follow the rules contained within the DTD, a parser generates an error
- An XML document that conforms to the rules within a DTD is said to be **valid**

Why Use a DTD?

- A single DTD ensures a common format for each XML document that references it
- An application can use a standard DTD to verify that data that it receives from the outside world is valid
- A description of legal, valid data further contributes to the interoperability and efficiency of using XML

Some Example DTD Declarations

- Example 1: The Empty Element

```
<!ELEMENT Bool (EMPTY)> <!--DTD declaration of empty element-->
<Bool Value="True"></Bool> <!--Usage with attribute in XML file--
>
```

- Example 2: Elements with Data

```
<!ELEMENT Month (#PCDATA)> <!--DTD declaration of an element->
<Month>April</Month> <!—Valid usage within XML file-->

<Month>This is a month</Month> <!—Valid usage within XML file-->

<Month> <!—Invalid usage within XML file, can't have children!-->
<January>Jan</January>
<March>March</March>
</Month>
```

Some Example DTD Declarations

- Example 3: Elements with Children

  To specify that an element must have a single child element, include the element name within the parenthesis.

```
<!ELEMENT House (Address)> <!—A house has a single address-->
<House> <!—Valid usage within XML file-->
<Address>1345 Preston Ave Charlottesville Va 22903</Address>
</House>
```

  An element can have multiple children. A DTD describes multiple children using a *sequence*, or a list of elements separated by commas. The XML file must contain one of each element in the specified order.

```
<!--DTD declaration of an element-->
  <!ELEMENT address (person,street,city, zip)>
  <!ELEMENT person  (#PCDATA)>
  <!ELEMENT street  (#PCDATA)>
  <!ELEMENT city    (#PCDATA)>
  <!ELEMENT zip     (#PCDATA)>
  <!—Valid usage within XML file-->
  <address>
  <person>John Doe</person>
  <street>1234 Preston Ave.</street>
  <city>Charlottesville, Va</city>
  <zip>22903</zip>
  </address>
```

Cautions concerning DTDs

- All element declarations begin with <!ELEMENT and end with >
- The ELEMENT declaration is *case sensitive*
- The programmer must declare all elements within an XML file
- Elements declared with the #PCDATA content model can not have children
- When describing sequences, the XML document must contain exactly those elements in exactly that order.

Walking Through an Example

- Run the internal DTD and xml file in new 3.xml
- Note save the file before running
- Read this https://www.w3schools.com/xml/xml_dtd_intro.asp

Part 3: XML and Supplementary Technologies

- The W3C Document Object Model (DOM)
  - an API that allows developers to programmatically manage and access XML nodes
  - allows programmers to update and change XML documents within an application
  - reads the whole XML file and then stores a hierarchical tree structure containing all elements within the document
  - This tree has a single root node, which is the root element, and may contain many children, each of which represents an XML element

W3C DOM with JavaScript

- Example 1: Loading the XML document: DOMDocument
  - The programmer can use a Microsoft Active X object to parse an XML file

```
//Instantiate DOMDocument object
var XMLfile = new
  ActiveXObject("Msxml2.DOMDocument");
XMLfile.load("newspaper.xml");
var rootElement = XMLfile.documentElement;
document.write("The root node of the XML file is:
  ");
document.writeln("<b>" + rootElement.nodeName
  +"</b>");
```

W3C DOM with JavaScript

- Example 2: Accessing the Children Elements
  - The *childNodes* member of any element node gives the programmer access to all of the sibling nodes of that element

```
//traverse through each child of the root element
//and print out its name
for (i=0; i<rootElement.childNodes.length; i++) {
    var node = rootElement.childNodes.item(i);
    document.write("The name of the node is ");
    document.write("<b>" + node.nodeName + "</b>");
}
```

W3C DOM with JavaScript

- Example 3: Getting Element Attributes

```
//traverse through each child of the root element
//and print out its name
   for (i=0; i<rootElement.childNodes.length; i++) {
          //get the current element
          var elementNode = rootElement.childNodes.item(i);
          document.writeln("Processing Node: " +
                    elementNode.nodeName + "<BR>");

          var attributeValue;
          //get an attribute value by specific name
          attributeValue = elementNode.getAttribute("articleID");
          //print it out
          document.writeln("Attribute value: <b>" + attributeValue +
                    " </b><br>");
   }
```

Cautions with DOM

- Make sure that the XML file resides in the same directory as the html file with the JavaScript code
- The Attribute node does not appear as the child node of any other node type; it is not considered a child node of an element
- Use caution when outputting raw XML to Internet Explorer.If the programmer uses the *document.writeln* method, IE attempts to interpret the XML tags and jumbles the text. Instead, use an alert box when debugging.

Walking through an Example

1. Check the new 4.xml file and read the code
2. Run the code at w3schools at https://www.w3schools.com/xml/dom_intro.asp
3. Try to write a DOM parser in Java now using Eclipse (Homework)

Questions

- ROS uses xml and acts as a middleware at times
- To use ROS you need basics of XML