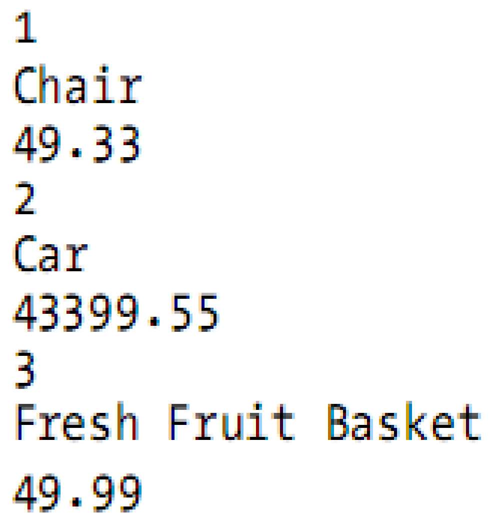
XML

What is XML

- *XML* stands for *Extensible Markup Language*
- Used to describe documents and data in a standardized, text-based format
- Based on Standard Generalized Markup Language (SGML)
- SGML was created in 1974, became International Organization for Standardization (ISO) standard in 1986
- HTML - a common language for sharing technical documents
    - Standardized way of describing document layout and display
    - But , there was no standardized way to describe and share data that was stored in the document.
        - Ex: closing prices of a share of every company
- In 1998 the World Wide Web Consortium (W3C) recommended first Extensible Markup Language (XML)

XML Explained

- Simple file format without XML
- Product information (ID, name, and price)



```
1
Chair
49.33
2
Car
43399.55
3
Fresh Fruit Basket
49.99
```

XML Explained

- You might need to resort to adding a header that indicates the version of the file

```
SuperProProductList
Version 2.0
1
 Chair
 49.33
 True
 2
 Car
 43399.55
 True
 3
 Fresh Fruit Basket
 49.99
 False
```

XML Explained

- Add a special sequence of characters (##Start##) to show where each new record begins

```
SuperProProductList
Version 3.0
##Start##
1
Chair
49.33
True
##Start##
2
Car
43399.55
True
##Start##
3
Fresh Fruit Basket
49.99
False
```

Improving the List with XML

```xml
<?xml version="1.0"?>
<SuperProProductList>
    <Product>
        <ID>1</ID>
        <Name>Chair</Name>
        <Price>49.33</Price>
        <Available>True</Available>
        <Status>3</Status>
    </Product>
    <Product>
        <ID>2</ID>
        <Name>Car</Name>
        <Price>43399.55</Price>
        <Available>True</Available>
        <Status>3</Status>
    </Product>
    <Product>
        <ID>3</ID>
        <Name>Fresh Fruit Basket</Name>
        <Price>49.99</Price>
        <Available>False</Available>
        <Status>4</Status>
    </Product>
</SuperProProductList>
```

XML Elements

- Element names can contain letters, numbers, hyphens, underscores, periods, and colons

  <element></element>

- Element names cannot contain spaces
- Element names can start with a letter, underscore, or colon, but cannot start with other non-alphabetic characters or a number, or the letters *xml*
- hyphens or periods – not suggested
- Elements with no attributes or text can also be represented in an XML document like this:

  <element/>

XML Attributes

- Attributes contain values that are associated with an element

  <element attribute=”value”></element>

- Attribute name must follow an element name, then an equals sign (=), then the attribute value, in single or double quotes

Text

- Text is located between the opening and closing tags of an element, and usually represents the actual data associated with the elements and attributes that surround the text:

  <element attribute="value">text</element>

XML Document Structure

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rootelement>
  <firstelement position="1">
    <level1 children="0">
      This is level 1 of the nested
      elements
    </level1>
  </firstelement>
  <secondelement position="2">
    <level1 children="1">
      <level2>
        This is level 2 of the nested
        elements
      </level2>
    </level1>
  </secondelement>
</rootelement>
```

XML Basics

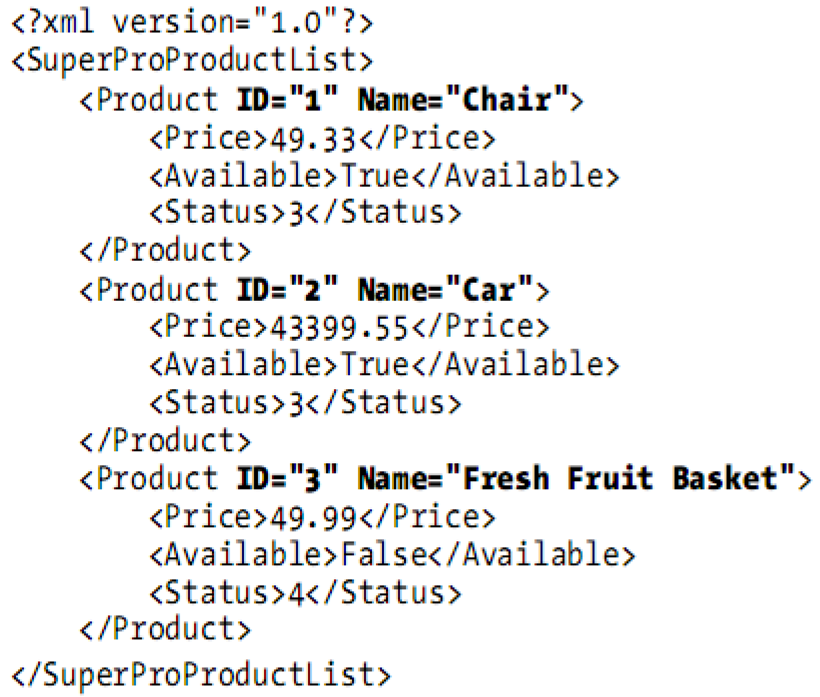
- XML elements are composed of a start tag (like <Name>) and an end tag (like </Name>)
- Whitespace between elements is ignored.
- XML elements are case sensitive
- All elements must be nested in a root element.
  - Ex: the root element is <SuperProProductList>
- Every element must be fully enclosed.

  <Product><ID></ID></Product> is valid, but

  <Product><ID></Product></ID> isn’t
- XML documents must start with an XML declaration like <?xml version="1.0"?>

XML Basics (Attributes)

- Attributes add extra information to an element.
- Use subelements or attributes



```
<?xml version="1.0"?>
<SuperProProductList>
    <Product ID="1" Name="Chair">
        <Price>49.33</Price>
        <Available>True</Available>
        <Status>3</Status>
    </Product>
    <Product ID="2" Name="Car">
        <Price>43399.55</Price>
        <Available>True</Available>
        <Status>3</Status>
    </Product>
    <Product ID="3" Name="Fresh Fruit Basket">
        <Price>49.99</Price>
        <Available>False</Available>
        <Status>4</Status>
    </Product>
</SuperProProductList>
```

You can use single or double quotes around any attribute value.

XML Basics (Comments)

- Comments are bracketed by the <!-- and --> character sequences.

```
<?xml version="1.0"?>
<SuperProProductList>
    <!-- This is a test file. -->
    <Product ID="1" Name="Chair">
        <Price>49.33<!-- Why so expensive? --></Price>
        <Available>True</Available>
        <Status>3</Status>
    </Product>
    <!-- Other products omitted for clarity. -->
</SuperProProductList>
```

Data Source Encoding

- Unicode was chosen as the standard text format to accommodate the world’s languages, instead of just English.
- *UTF* stands for *Universal Character Set Transformation Format* (UTF-8, UTF-16 or UTF-32)
- Each 8-, 16- or 32-bit container represents the value of the character in bits as well as the identity of each character and its numeric value
- **UTF-8 is an encoding - Unicode is a character set**
    - A character set is a list of characters with unique numbers (these numbers are sometimes referred to as "code points"). For example, in the Unicode character set, the number for *A* is 41.
    - An encoding on the other hand, is an algorithm that translates a list of numbers to binary so it can be stored on disk. For example UTF-8 would translate the number sequence 1, 2, 3, 4 like this:

      00000001 00000010 00000011 00000100

What is XML

The three pillars of XML are

- **Extensibility:** XML does a great job of describing structured data as text, and the format is open to extension. This means that any data that can be described as text and that can be nested in XML tags will be generally accepted as XML. The only limits are imposed on the data by the data itself, via syntax rules and self-imposed format directives via data validation
- **Structure:** The structure of XML is usually complex and hard for human eyes to follow, but it's important to remember that it's not designed for us to read. XML parsers and other types of tools that are designed to work with XML easily digest XML, even in its most complex forms.
- **Validity:** Aside from the mandatory syntax requirements that make up an XML document, data represented by XML can optionally be validated for structure and content, based on two separate data validation standards. The original XML data validation standard is called Data Type Definition (DTD), and the more recent evolution of XML data validation is the XML Schema standard.

What Is XML Not?

- While XML facilitates data integration by providing a transport with which to send and receive data in a common format, XML is not data integration. It's simply the glue that holds data integration solutions together with a multi-platform "lowest common denominator" for data transportation.
- XML cannot make queries against a data source or read data into a repository by itself.
- Similarly, data cannot be formatted as XML without additional tools or programming languages that specifically generate XML data from other types of data.
- Also, data cannot be parsed into destination data formats without a parser or other type of application that converts data from XML to a compatible destination format.
- It's also important to point out that XML is not HTML. XML may look like HTML, based on the similarities of the tags and the general format of the data, but that's where the similarity ends. While HTML is designed to describe display characteristics of data on a Web page to browsers, XML is designed to represent data structures.

XML Namespaces

- Namespaces are a method for separating and identifying duplicate XML element names in an XML document.
  - In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.
- Namespaces can also be used as identifiers to describe data types and other information.
- The namespace can be defined by an **xmlns** attribute in the start tag of an element.
- The namespace declaration has the following syntax.

  xmlns:*prefix*="*URI*".
- Define the prefix in the xmlns attribute by inserting a colon (:) followed by the characters you want to use for the prefix
- **Note:** The namespace URI is not used by the parser to look up information. The purpose of using an URI is to give the namespace a unique name. However, companies often use the namespace as a pointer to a web page containing namespace information

XML Namespaces

```
<root>
  <h:table xmlns:h="http://www.w3.org/TR/html4/">
    <h:tr>
      <h:td>Apples</h:td>
      <h:td>Bananas</h:td>
    </h:tr>
  </h:table>

  <f:table xmlns:f="http://www.w3schools.com/furniture">
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>
</root>
```

This XML carries information about an HTML table, and a piece of furniture. If these XML fragments were added together, there would be a name conflict. Both contain a <table> element, but the elements have different content and meaning. A user or an XML application will not know how to handle these differences.

XML Namespaces

- To assign a namespace to all the elements, you can use an xmlns attribute to assign a URI to the default namespace:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rootelement xmlns="www.mit.edu">
  <firstelement position="1">
    <level1>level1 elements</level1>
  </firstelement>
  <secondelement position="2">
    <level1 children="1">
      <level2>level2 elements</level2>
    </level1>
  </secondelement>
</rootelement>
```

Here `firstelement` and `secondelement` belong to the same namespace as `rootelement`