Chapter 2: Well-Formed XML

Chapter 2 Objectives

- Understand components of an XML document
- How to create XML elements using start-tags and end-tags
- How to further describe elements with attributes
- How to declare your document as being XML
- How to send instructions to applications that are processing the XML document
- Which characters aren't allowed in XML – and how to use them in your documents anyway!
- What is well-formedness anyway?

Components

- Elements
- Attributes
- Declaration
- Instructions
- Comments
- CDATA

Tags and Elements

```
<name>
<first>John</first>
<middle>Fitzgerald Johanson</middle>
<last>Doe</last>
</name>
```

- `<first>` is a start-tag
- `</first>` is an end-tag
- `<first>John</first>` is an element
- `John` is the content of the element

- Analogy: Sandwich

Try It Out

Creating a Distribution Process

Rules for Elements

Every Start Tag Must Have An End Tag

Bad Example

```
<P>Here is some text in an HTML paragraph.
<BR>
Here is some more text in the same paragraph.
<P>And here is some text in another HTML
  paragraph.</P>
```

Good Example

```
<P>Here is some text in an HTML paragraph.</P>
<P>Here is some more text in the same paragraph.</P>
<P>And here is some text in another HTML
  paragraph.</P>
```

Rules for Elements

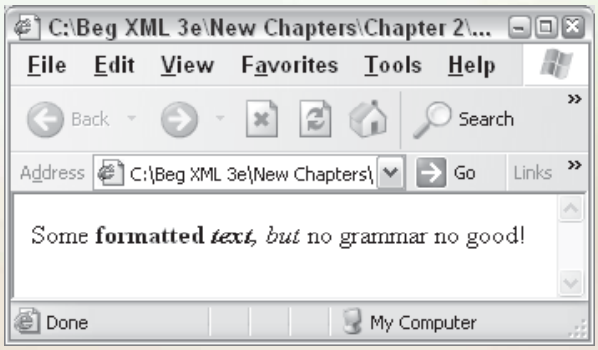
Elements Must Be Properly Nested

Bad Example

```
<P>Some <STRONG> formatted <EM>text</STRONG>, but</EM>
   no grammar no good!</P>
```

Good Example

```
<P>Some
  <STRONG>
     Formatted
        <EM>
          Text
        </EM>
  </STRONG>
        <EM>
         , but
        </EM>
  No grammar no good!
</P>
```



Rules for Elements

Can Have Only One Root Element

Bad Example

```
<name>John</name>
<name>Jane</name>
```

Good Example

```
<names>
<name>John</name>
<name>Jane</name>
</names>
```

Rules for Elements

Elements Must Obey XML Naming Conventions

- Names can start with letters, no numbers
- After first character, numbers, hyphens and periods are allowed
- Names can't contain spaces
- There are *reserved* characters like ":"
- Names can't start with the letters "xml", "XML", or "Xml", or any other combination
- No spaces after the "<", but before the ">" if desired

Rules for Elements

Case Sensitivity

These are two different elements

```
<name>John</name>
<NAME>John</NAME>
```

Rules for Elements

Whitespace in PCDATA

Whitespace stripping takes place in HTML…

```
<P>This is a paragraph.
      It has a whole
  bunch<BR>  of space.</P>
```

…but not in XML

Example

```
<tag>This is a paragraph.  It has a whole
Bunch of space.</tag>

This is a paragraph.  It has a whole
Bunch of space.
```


Rules for Elements

End-of-Line Whitespace

- Windows uses both the line feed and the carriage return
- UNIX uses only line feed
- XML parsers will convert all Windows "line feed and carriage returns" to just line feed characters to standardize end-of-line logic

Rules for Elements

Readability Whitespace

<Tag>
  <AnotherTag>This is some XML</AnotherTag>
</Tag>

This is known as extraneous whitespace in the markup.

Attributes

- Simple name/value pairs associated with an element
- name="value" or name='value'
- Can have more than one attributes for an element
- Order is not important
- Elements' order is important!

```
<name nickname="Shiny John">
  <first>John</first>
  <middle>Fitzgerald Johansen</middle>
  <last>Doe</last>
</name>
```

(Adding attributes to orders …)

Elements or Attributes?

- What is the difference between

```
<name nickname="Shiny John">
</name>
```

and

```
<name>
 <nickname>Shiny John</nickname>
</name>
```

- Element can contain elements
  - Not the case for attribute
- Order
- Part of data or property of data
- Personal preference / style

Comments

```
<name nickname='Shiny John'>
  <first>John</first>
<!--John lost his middle name in a fire-->
  <middle></middle>
  <last>Doe</last>
</name>
```

(Adding comments to orders …)

Empty Elements

```
<name nickname='Shiny John'>
  <first>John</first>
  <!--John lost his middle name in a fire-->
  <middle></middle>
  <last>Doe</last>
</name>
```

can also use self-closing tag `<middle/>`

ok to have attributes

```
<name nickname="Shiny John"/>
```


XML Declaration

Define the Encoding and Standalone Attributes

```
<?xml version='1.0' encoding='UTF-16' standalone='yes'?>
<name nickname='Shiny John'>
  <first>John</first>
  <!--John lost his middle name in a fire-->
  <middle/>
  <last>Doe</last>
</name>
```

(Adding declaration to orders …)

Processing Instructions

```
<?xml version='1.0'?>
<name nickname='Shiny John'>
  <first>John</first>
  <!--John lost his middle name in a
  fire-->
  <middle/>
  <?nameprocessor SELECT * FROM blah?>
  <last>Doe</last>
</name>
```


Processing Instructions

```
<?xml version='1.0'?>
<name nickname='Shiny John'>
  <first>John</first>
  <!--John lost his middle name in a fire-->
  <middle/>
  <?nameprocessor SELECT * FROM blah?>
  <last>Doe</last>
</name>
```

PITarget *instruction*

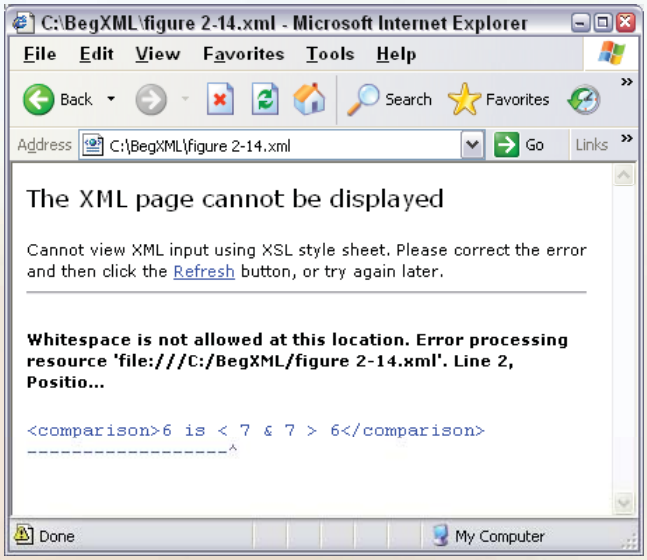
Is the XML declaration a PI? **NO!**

(Adding PI to orders …)

Illegal PCDATA Characters

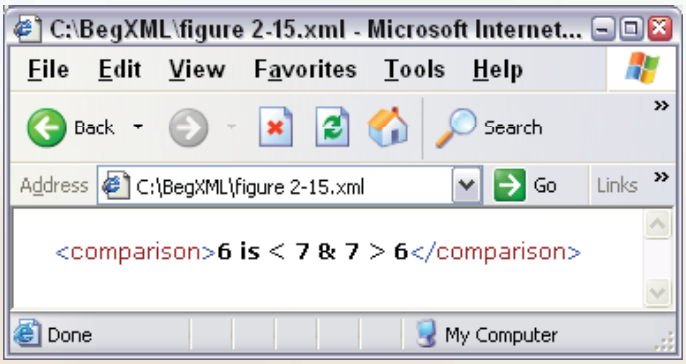
- **PCDATA: Parsed character data**

```
<!--This is not well-formed XML!-->
<comparison>6 is < 7 & 7 > 6</comparison>
```



Escaping Characters

```
<comparison>
   6 is &lt; 7 &amp; 7 &gt; 6
</comparison>
```



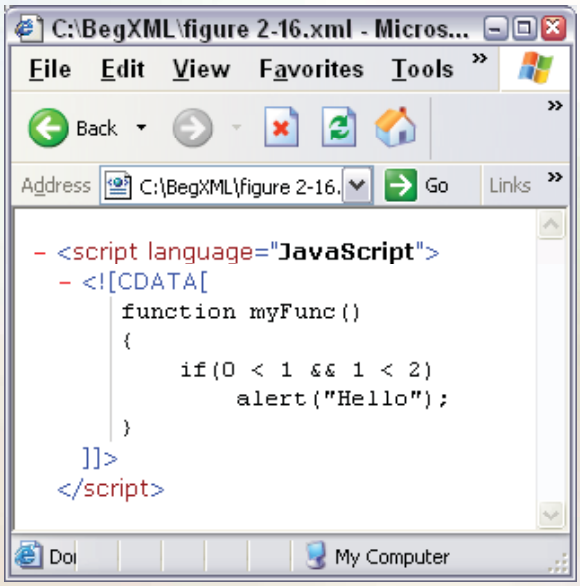
Escaping Characters

- Entity reference
  - & &
  - < <
  - > >
  - ' ‘
  - " “
  - &#*n* or &#x*n*
    - &#169 or &#xA9 ©

CDATA Sections

•CDATA: Character data

```
<script language='JavaScript'>
<![CDATA[
      function myFunc()
{
    if(0 < 1 && 1 < 2)
        alert("Hello");
}
]]>
</script>
```



•Example: HTML in XML

Errors in XML

There Are Errors, and Then There Are Fatal Errors

- Errors
  - Violations rules in recommendation
  - May recover
  - Continue processing
- Fatal errors
  - Draconian error handling
  - Not allowed to continue
  - XML parser should not try to “recover” errors

Well-Formedness

- XML documents must have exactly one root element
- A start tag must have a matching end tag
- XML tags are case sensitive
- XML elements must be properly nested
- XML attribute values must be quoted
- Reserved characters must be handled properly