Introduction to XML

Presentation Outline

- **Chapter (1):** Introduction To XML

  What is XML? and, what XML can do?

  Creating XML

  To learn XML syntax rules (writing XML document)

- **Chapter (2):** DTD

  To validate the XML document.

- **Chapter (3):** XML Schema

  Another way to validate the of XML document.

Chapter 1

Creating Markup with XML

XML Introduction ?

- Stands for e**X**tensible **M**arkup **L**anguage.
- W3C Initiative.
- Created to provide a standard format for computer documents that is used as :
  - Websites documents
  - Electronic data interchange documents
    - (Example: Configuration file)

What is XML?

- XML is a text file organized using tags to be:
  - **Standard based**
    - To be language independent and platform independent.
  - **Simple**
    - So it is ‘human and machine understandable’.
  - **Self describing** for the data it contains.
    - Because, it separates the content from its presentation
  - **Validated against strict rules** to
    - Make XML document optimized for computer manipulation
    - Be easy for data exchange

What XML is? (Cont'd)

- XML, unlike HTML,

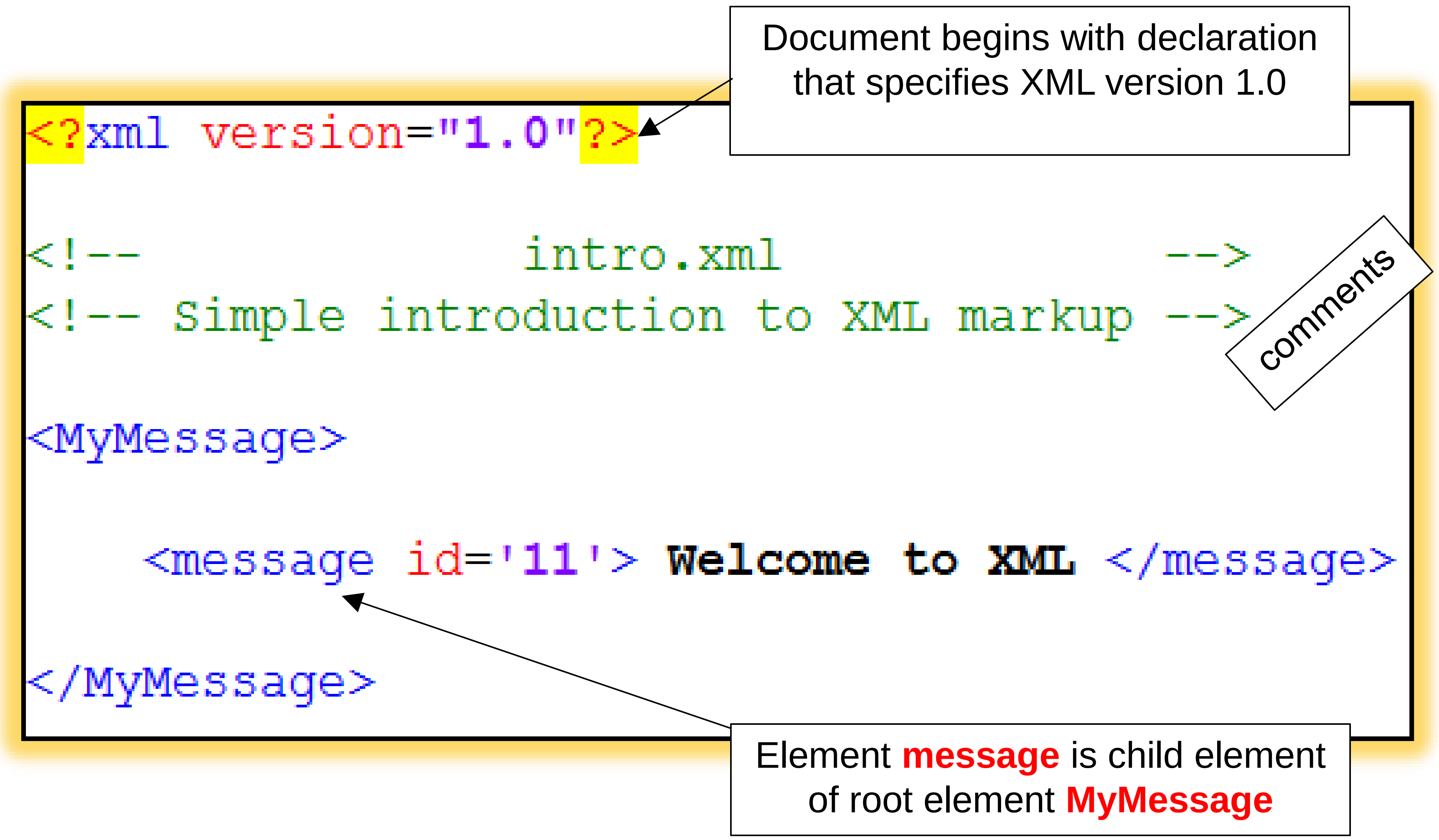
  Doesn't have a fixed or predefined set of tags,

  The designer of the XML document invent his own set of tags.

Introduction

- **XML:**
  - Technology for creating markup languages
  - Enables authors to describe data of any type
  - Allows creating new tags
    - HTML limits authors to fixed tag set
  - Commonly stored in text files
    - Extension **`.xml`**
  - Example: **`intro.xml`**

XML Example



```
<?xml version="1.0"?>

<!--              intro.xml              -->
<!-- Simple introduction to XML markup -->

<MyMessage>

    <message id='11'> Welcome to XML </message>

</MyMessage>
```

Document begins with declaration that specifies XML version 1.0

comments

Element **message** is child element of root element **MyMessage**

Well-formed XML

- XML document Considered ***well formed*** if it has:
  1. Single root element.
  2. Each element has start tag and end tag.
     - Empty element is defined as: `<element/>`
  3. Tags well nested.
     - Incorrect: `<x><y>hello</x></y>`
     - Correct: `<x><y>hello</y></x>`
  4. Attribute values in quotes.
  5. Tag & Attributes names written as variable names:
     - Start with character,
     - One word “must not contain spaces”,
     - Case sensitive.
  6. An element may not have two attributes with the same name.

Parsers

- **XML parser:**
  - Processes XML document:
    - Reads XML document.
    - Checks syntax.
    - Reports errors (if any).
  - Example:
    - Internet browser
    - XML Editors.
    - Built in component Java JDK.

Characters

- **Characters:**
  - ASCII characters:
    - Letters of English alphabet
    - Digits (**0–9**)
    - Punctuation characters, such as **!** , **-** and **?**
    - Carriage returns “\r” .
    - Line feeds “\n”.
  - Unicode characters:
    - Enables computers to process characters for several languages.

Entity References & Built-in Entities

- XML **Reserved** characters:
  - Ampersand (`&`)
  - Left-angle bracket (**<**)
  - Right-angle bracket (**>**)
  - Apostrophe (′)
  - Double quote (″)

- But How to make this sequence
  - “ 5 < `x && y` > 6” in element **`message`** ?!!!

Entity References and Built-in Entities (Cont'd)

- **Entity references** (**Example**)
  - Allow to use XML-reserved characters
    - Begin with ampersand (`&`) and end with semicolon (`;`)

- **Built-in entities:**

(&) → (&amp;)	(<) → (&lt;)
(>) → (&gt;)	(′) → (&apos;)
(″) → (&quot;)	

- So

  “ 5 < x && y > 6” → written as

```
5 &lt; x &amp;&amp; y &gt; 6
```

Attributes

- **Attributes:**
  - Elements may have associated attributes:
  - Placed within element's start tag:
  - Values are enclosed in quotes (single or double):
    - Element **`Father`** contains
      - attribute ***name***, which has value ***"Ahmad"***

```
<Father name = "Ahmad">
    <Child> Mohamad </Child>
</Father>
```

Processing instruction

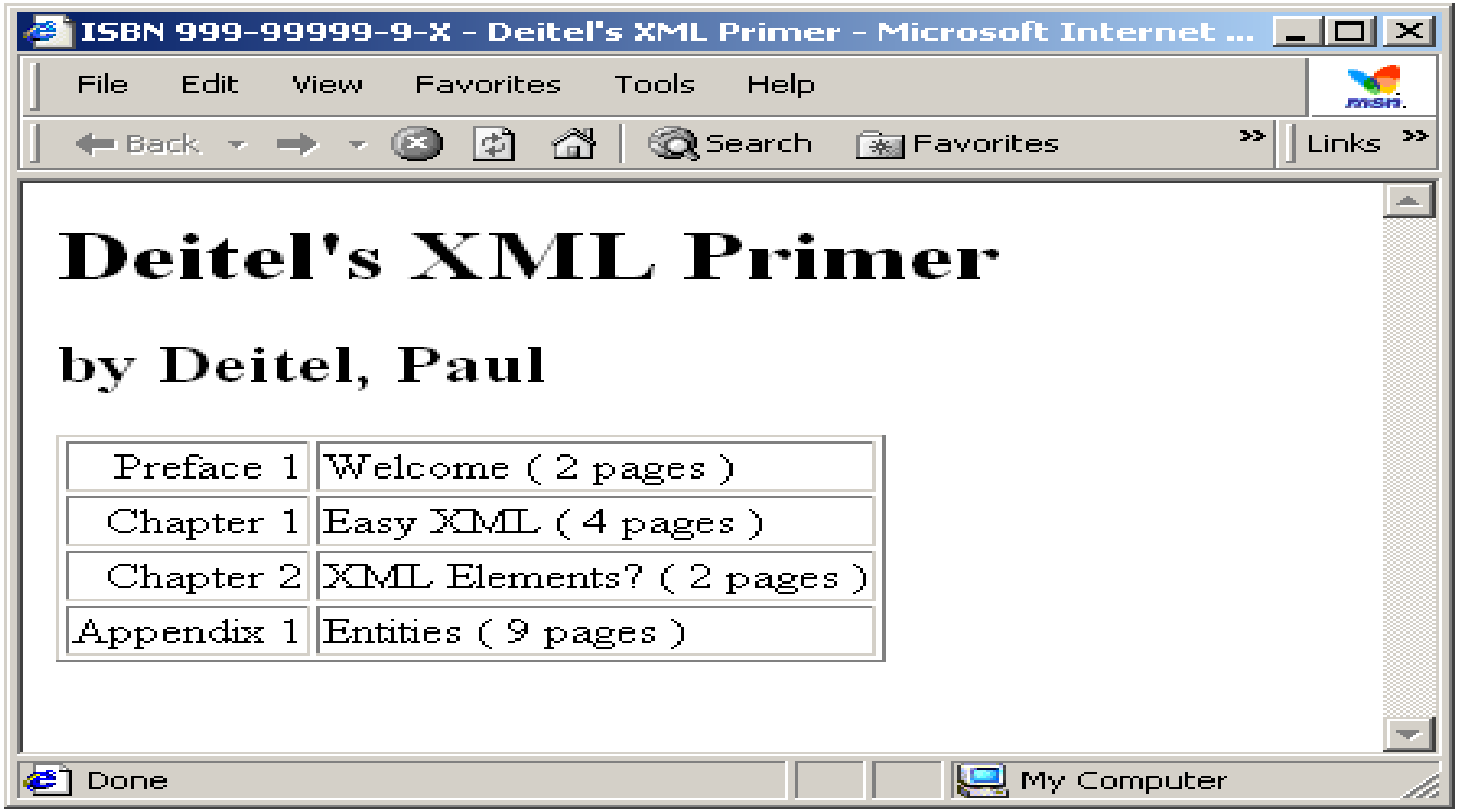
- **Processing instruction (PI)**
  - Passed to application using XML document.
  - Provides application-specific document information.
  - Delimited by **`<?`** and **`?>`**

PI Example

```
<?xml version = "1.0"?>

<!-- Fig. 5.5 : usage.xml                 -->
<!-- Usage of elements and attributes -->

<?xml:stylesheet type = "text/xsl" href = "usage.xsl"?>

<book isbn = "999-99999-9-X">
   <title>Deitel&amp;s XML Primer</title>

   <author>
      <firstName>Paul</firstName>
      <lastName>Deitel</lastName>
   </author>

   <chapters>
      <preface num = "1" pages = "2">Welcome</preface>
      <chapter num = "1" pages = "4">Easy XML</chapter>
      <chapter num = "2" pages = "2">XML Elements?</chapter>
      <appendix num = "1" pages = "9">Entities</appendix>
   </chapters>

   <media type = "CD"/>
</book>
```

Stylesheet (discussed in Chapter 7)

Present XML as HTML page Using XSL


ISBN 999-99999-9-X - Deitel's XML Primer - Microsoft Internet ...
File Edit View Favorites Tools Help
Back Search Favorites Links
Done
My Computer

Deitel's XML Primer

by Deitel, Paul

Preface 1	Welcome (2 pages)
Chapter 1	Easy XML (4 pages)
Chapter 2	XML Elements? (2 pages)
Appendix 1	Entities (9 pages)

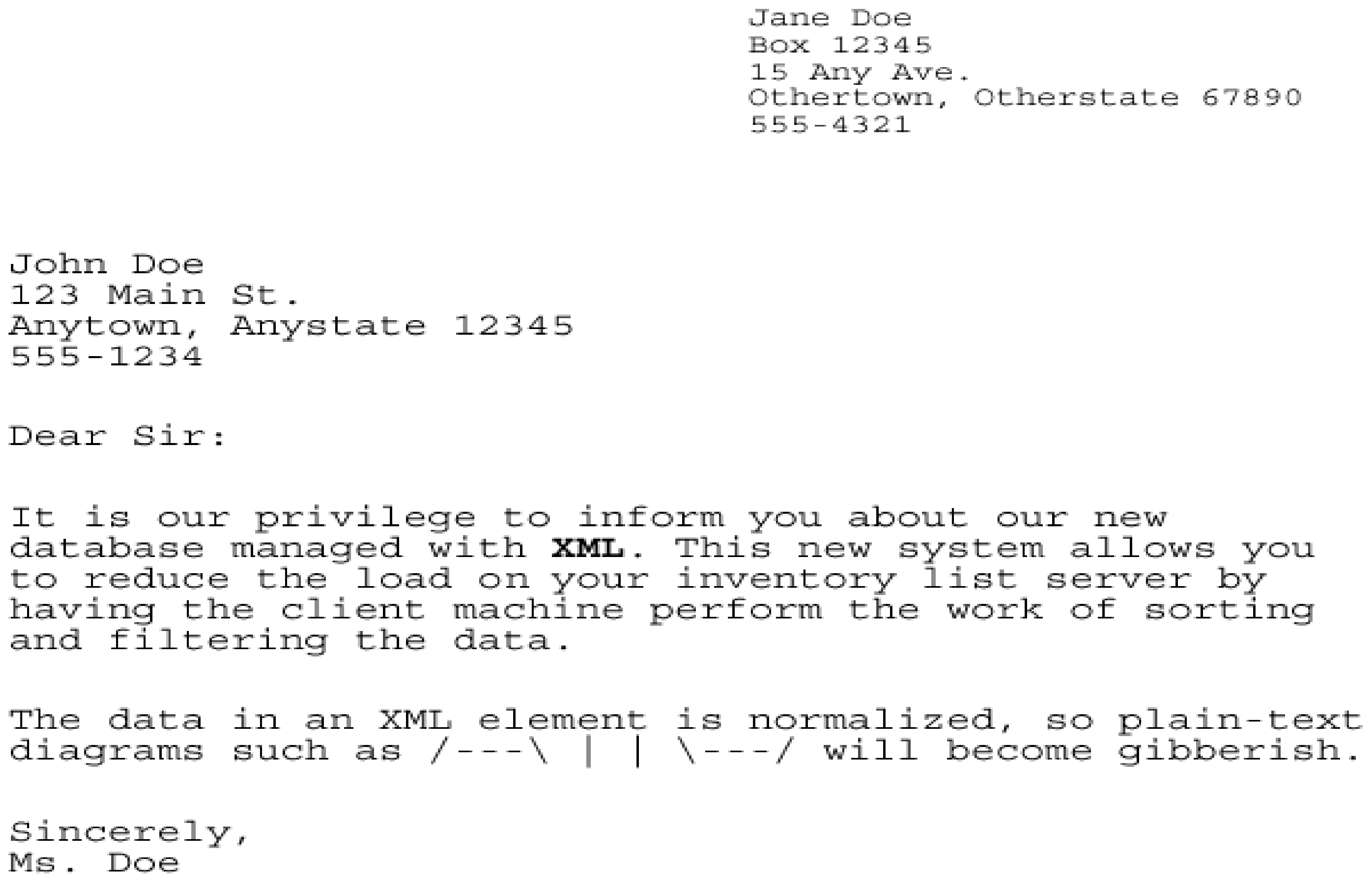
PI Example 2

```xml
<?xml version = "1.0"?>

<!-- Fig. 5.6: letter.xml                   -->
<!-- Business letter formatted with XML -->

<letter>

   <contact type = "from">
      <name>Jane Doe</name>
      <address1>Box 12345</address1>
      <address2>15 Any Ave.</address2>
      <city>Othertown</city>
      <state>Otherstate</state>
      <zip>67890</zip>
      <phone>555-4321</phone>
      <flag gender = "F"/>
   </contact>

   <contact type = "to">
      <name>Jane 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>
```

PI Example 2 (cont’d)

```
   <salutation>Dear Sir:</salutation>

   <paragraph>It is our privilege to inform you about our new
      database managed with <bold>XML</bold>. This new system
      allows you to reduce the load on your inventory list
      server by having the client machine perform the work of
      sorting and filtering the data.</paragraph>

   <paragraph>The data in an XML element is normalized, so
      plain-text diagrams such as
         /---\
         |   |
         \---/
      will become gibberish.</paragraph>

   <closing>Sincerely</closing>
   <signature>Ms. Doe</signature>

</letter>
```

Present XML As PDF Document Using XSL-FO.



Jane Doe
Box 12345
15 Any Ave.
Othertown, Otherstate 67890
555-4321

John Doe
123 Main St.
Anytown, Anystate 12345
555-1234

Dear Sir:

It is our privilege to inform you about our new
database managed with **XML**. This new system allows you
to reduce the load on your inventory list server by
having the client machine perform the work of sorting
and filtering the data.

The data in an XML element is normalized, so plain-text
diagrams such as /---\ | | \---/ will become gibberish.

Sincerely,
Ms. Doe

CDATA Sections

- `CDATA` sections:
  - May contain text, reserved characters and white space
    - Reserved characters need not be replaced by entity references
  - Not processed by XML parser
  - Commonly used for scripting code (e.g., JavaScript)
  - Begin with `<![CDATA[`
  - Terminate with `]]>`

CDATA Example

```
<?xml version = "1.0"?>

<!-- Fig. 5.7 : cdata.xml                 -->
<!-- CDATA section containing C++ code    -->

<book title = "C++ How to Program" edition = "3">

   <sample>
         // C++ comment
         if ( this-&gt;getX() &lt; 5 &amp;&amp; value[ 0 ] != 3 )
            cerr &lt;&lt; this-&gt;displayError();
   </sample>

   <sample>
      <![CDATA[

         // C++ comment
         if ( this->getX() < 5 && value[ 0 ] != 3 )
            cerr << this->displayError();
      ]]>
   </sample>

   C++ How to Program by Deitel &amp; Deitel
</book>
```

Entity references required if not in **CDATA** section

XML does not process **CDATA** section

Note the simplicity offered by **CDATA** section

Lab Exercise

Assignment: A Configuration File

- Design a configuration file for a library.
  - Info. of library consists of a location, a description of the library, a librarian and a lot of books.
  - Each book has title, ISBN, and Author. The book contains also a preface and many of parts.
  - Each part has title and contains many of chapters.
  - Each chapter has title and contains a summary and many of sections.
  - Sections contain the content of the book as paragraphs.

XML must have elements (usual and empty), attributes, and CDATA section