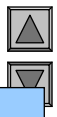
Chapter 20 - Extensible Markup Language (XML)

Outline



Outline

article.xml

```
1   <?xml version = "1.0"?>
2
3   <!-- Fig. 20.1: article.xml
4   <!-- Article structured with XML
5
6   <article>
7
8      <title>Simple XML</title>
9
10     <date>September 19, 2001</date>
11
12     <author>
13        <firstName>Tem</firstName>
14        <lastName>Nieto</lastName>
15     </author>
16
17     <summary>XML is pretty easy.</summary>
18
19     <content>Once you have mastered XHTML, XML is easily
20        learned. You must remember that XML is not for
21        displaying information but for managing information.
22     </content>
23
24  </article>
```

Element **article** is the root element.

Optional XML declaration.

Elements **title**, **date**, **author**, **summary** and **content** are child elements of **article**.

20.2 Structuring Data

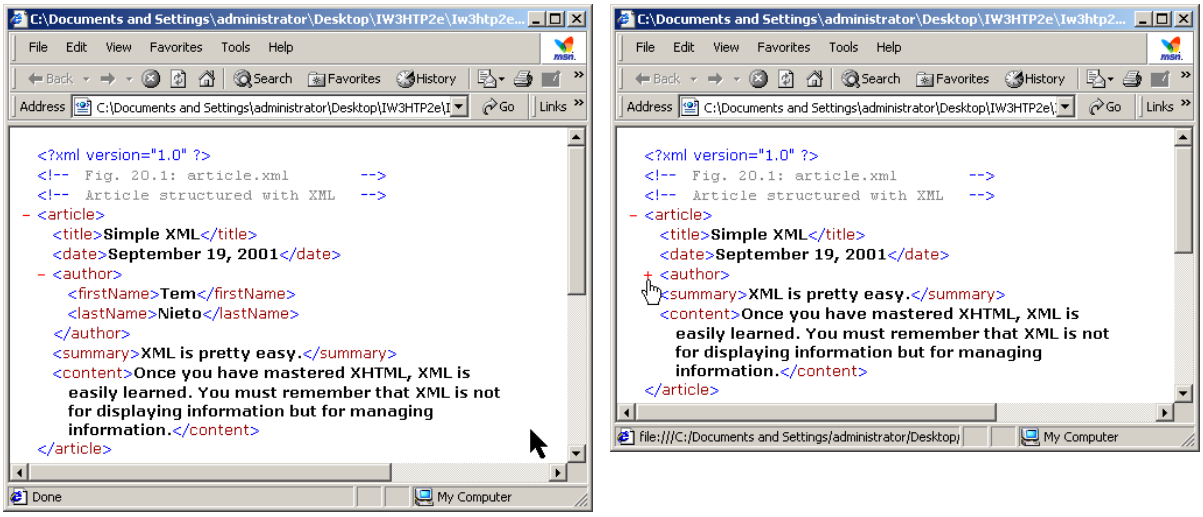
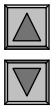


Fig. 20.2 IE5.5 displaying article.xml.



Outline

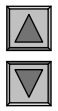
Letter.xml

```
1   <?xml version = "1.0"?>
2
3   <!-- Fig. 20.3: letter.xml                    -->
4   <!-- Business letter formatte
5
6   <!DOCTYPE letter SYSTEM "lett
7
8   <letter>
9
10     <contact type = "from">
11        <name>John Doe</name>
12        <address1>123 Main St.</address1>
13        <address2></address2>
14        <city>Anytown</city>
15        <state>Anystate</state>
16        <zip>12345</zip>
17        <phone>555-1234</phone>
18        <flag gender = "M"/>
19     </contact>
20
21     <contact type = "to">
22        <name>Joe Schmoe</name>
23        <address1>Box 12345</address1>
24        <address2>15 Any Ave.</address2>
25        <city>Othertown</city>
26        <state>Otherstate</state>
27        <zip>67890</zip>
28        <phone>555-4321</phone>
29        <flag gender = "M"/>
30     </contact>
31
```

Element **flag** is an empty element and does not contain any text.

Information for the person writing the letter.

Information for the person receiving the letter.



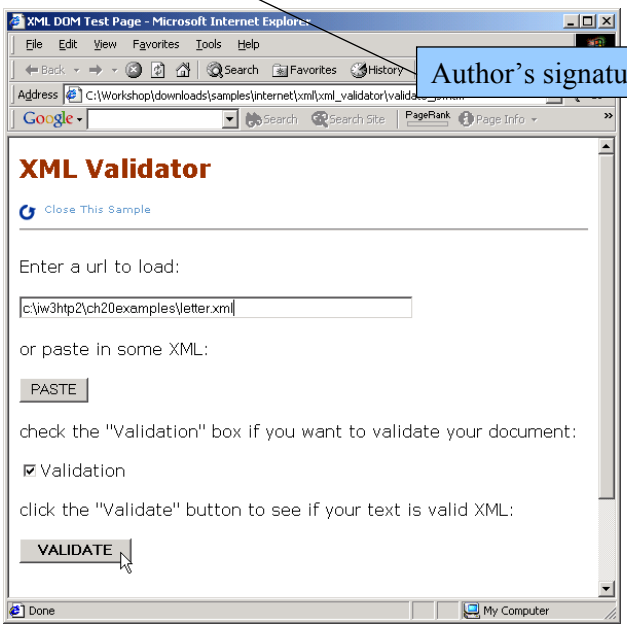
Outline

Letter.xml

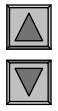
```
32      <salutation>Dear Sir:</salutation>
33
34      <paragraph>It is our privilege to inform you about our new
35         database managed with XML. This new system allows
36         you to reduce the load of your inventory list server by
37         having the client machine perform the work of sorting
38         and filtering the data.</paragraph>
39      <closing>Sincerely</closing>
40      <signature>Mr. Doe</signature>
41
42   </letter>
```

Body of letter.

Author's signature.

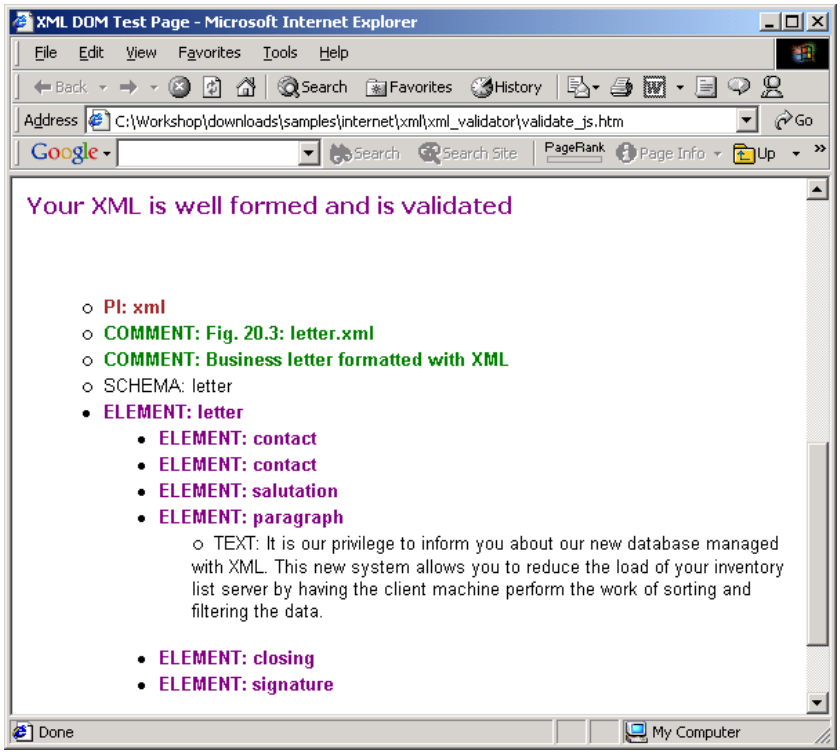


m Output



Outline

Program Output



Outline

```
1   <?xml version = "1.0"?>
2
3   <!-- Fig. 20.4 : namespace.xml
4   <!-- Demonstrating Namespaces
5
6   <text:directory xmlns:text = "urn:deitel:textInfo"
7      xmlns:image = "urn:deitel:imageInfo">
8
9      <text:file filename = "book.xml">
10        <text:description>A book list</text:description>
11     </text:file>
12
13     <image:file filename = "funny.jpg">
14        <image:description>A funny picture</image:description>
15        <image:size width = "200" height = "100"/>
16     </image:file>
17
18  </text:directory>
```

Keyword **xmlns** creates two namespace prefixes, **text** and **image**.

URIs ensure that a namespace is unique.

Namespace.xml

Outline

```
1   <?xml version = "1.0"?>
2
3   <!-- Fig. 20.5 : defaultnamespace.xml -->
4   <!-- Using Default Namespaces         -->
5
6   <directory xmlns = "urn:deitel:textInfo"
7      xmlns:image = "urn:deitel:imageInfo">
8
9      <file filename = "book.xml">
10        <description>A book list</description>
11     </file>
12
13     <image:file filename = "funny.jpg">
14        <image:description>A funny picture</image:description>
15        <image:size width = "200" height = "100"/>
16     </image:file>
17
18  </directory>
```

Default namespace.

Element **file** uses the default namespace.

Element **file** uses the namespace prefix **image**.

Defaultnamespace

```
1    <!-- Fig. 20.4: let
2    <!-- DTD document
3
4    <!ELEMENT letter (
5       closing, signat
6
7    <!ELEMENT contact ( name, address1, address2, city, state,
8       zip, phone, flag )>
9    <!ATTLIST contact type CDATA #IMPLIED>
10
11   <!ELEMENT name ( #PCDATA )>
12   <!ELEMENT address1 ( #PCDATA )>
13   <!ELEMENT address2 ( #PCDATA )>
14   <!ELEMENT city ( #PCDATA )>
15   <!ELEMENT state ( #PCDATA )>
16   <!ELEMENT zip ( #PCDATA )>
17   <!ELEMENT phone ( #PCDATA )>
18   <!ELEMENT flag EMPTY>
19   <!ATTLIST flag gender (M | F) "M">
20
21   <!ELEMENT salutation ( #PCDATA )>
22   <!ELEMENT closing ( #PCDATA )>
23   <!ELEMENT paragraph ( #PCDATA )>
24   <!ELEMENT signature ( #PCDATA )>
```

The **contact** element definition specifies that element **contact** ... **city**, ...

The ***ATTLIST*** element type declaration defines an attribute (i.e., **type**) for the **contact** element.

Keyword **#IMPLIED** specifies that if the parser finds a **contact** element without a **type** attribute, the parser can choose an arbitrary value for the attribute or ignore the attribute and the document will be valid.

Flag **#PCDATA** specifies that the element can contain parsed character data (i.e., text).

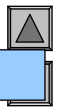
Outline

Book.xml

```
1    <?xml version = "1.0"?>
2    <!-- Fig. 20.7 : book.xml       -->
3    <!-- Book list marked up as XML -->
4
5    <deitel:books xmlns:deitel = "http://www.deitel.com/booklist">
6       <book>
7          <title>XML How to Program</title>
8       </book>
9       <book>
10         <title>C How to Program</title>
11      </book>
12      <book>
13         <title>Java How to Program</title>
14      </book>
15      <book>
16         <title>C++ How to Program</title>
17      </book>
18      <book>
19         <title>Perl How to Program</title>
20      </book>
21   </deitel:books>
```

Program Output

```
java -classpath .;..\lib\xmlparserv2.jar;..\lib\xschema.jar
XSDSetSchema book.xsd book.xml
The input file <book.xml> parsed without errors
```



Outline

Book.xsd

```
1   <?xml version = "1.0"?>
2
3   <!--
4   <!--
5
6
7
8
9
10
11
12     <xsd:complexType name = "Boo
13        <xsd:element name = "book" type = "deitel:BookType"
14           minOccurs = "1" maxOccurs = "unbounded"/>
15     </xsd:complexType>
16
17     <xsd:complexType name = "BookType">
18        <xsd:element name = "title" type = "xsd:string"/>
19     </xsd:complexType>
20
21  </xsd:schema>
```

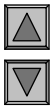
Attributes ***name*** and ***type*** specify the **element**'s name and data type, respectively.

element to be

an element type

Attribute **maxOccurs**, with value **unbounded** specifies that **books** may have any number of **book** child elements.

s specifies that **books** must contain a minimum of one **book** element.

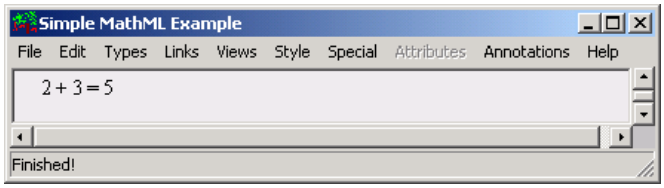


Outline

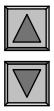
Mathml1.html

```
1   <?xml version="1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4
5   <!-- Fig. 20.9: mathml1.html -->
6   <!-- Simple MathML           -->
7
8   <html xmlns="http://www.w3.org/1999/xhtml">
9
10     <head><title>Simple MathML Example</title></head>
11
12     <body>
13
14        <math xmlns = "http
15
16           <mrow>
17              <mn>2</mn>
18              <mo>+</mo>
19              <mn>3</mn>
20              <mo>=</mo>
21              <mn>5</mn>
22           </mrow>
23
24        </math>
25
26     </body>
27  </html>
```

Element **mrow** behaves like parentheses, which allows the document author to group related elements properly.



Program Output



Outline

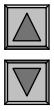
Mathml2.html

```
1   <?xml version="1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4   <!-- Fig. 20.10: mathml2.html -->
5   <!-- Simple MathML            -->
6
7   <html xmlns="http://www.w3.org/1999/xhtml">
8
9      <head><title>Algebraic Mat
10
11     <body>
12
13        <math xmlns = "http://www.w3.org/1998/Math/MathML">
14           <mrow>
15
16              <mrow>
17                 <mn>3</mn>
18                 <mo>&InvisibleTimes;</mo>
19
20                 <msup>
21                    <mi>x</mi>
22                    <mn>2</mn>
23                 </msup>
24
25              </mrow>
26
27              <mo>+</mo>
28              <mi>x</mi>
29              <mo>-</mo>
30
31              <mfrac>
32                 <mn>2</mn>
33                 <mi>x</mi>
34              </mfrac>
35
```

entity reference **⁢** to indicate a multiplication operation without a symbolic representation.

The **msup** element represents a superscript.

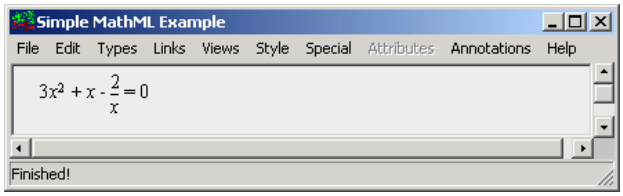
Element **mfrac** display a fraction.



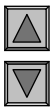
Outline

Mathml2.html

```
36              <mo>=</mo>
37              <mn>0</mn>
38
39           </mrow>
40        </math>
41
42     </body>
43  </html>
```



Program Output



Outline

Mathml3.html

```
1   <?xml version="1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4   <!-- Fig. 20.11 mathml3.html       -->
5   <!-- Calculus example using MathML -->
6
7   <html xmlns="http://www.w3.org/1999/xhtml">
8
9      <head><title>Calculus MathML
10
11     <body>
12
13        <math xmlns = "http://www.w3.org/1998/Math/MathML">
14           <mrow>
15              <msubsup>
16
17                 <mo>&Integral;</mo
18                 <mn>0</mn>
19
20                 <mrow>
21                    <mn>1</mn>
22                    <mo>-</m
23                    <mi>y</m
24                 </mrow>
25
26              </msubsup>
27
28              <msqrt>
29                 <mrow>
30
31                    <mn>4</mn>
32                    <mo>&InvisibleTimes;</mo>
33
```

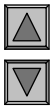
The entity **∫** represents the integral symbol.

Element **mo** marks up the integral operator.

Element **mn** marks up the number (i.e., **0**) that represents the subscript.

pt and subscript.

Element **msqrt** represents a square root expression.

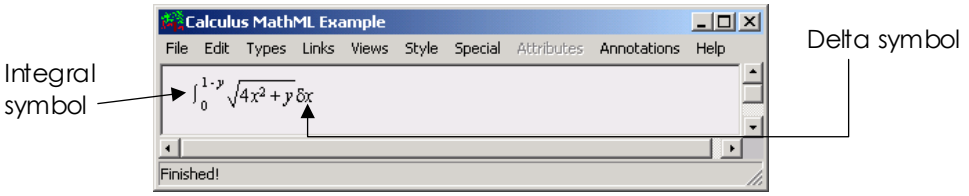


Outline

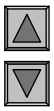
Mathml3.html

```
34                    <msup>
35                       <mi>x</mi>
36                       <mn>2</mn>
37                    </msup>
38
39                    <mo>+</m
40                    <mi>y</m
41
42                 </mrow>
43              </msqrt>
44
45              <mo>&delta;</mo>
46              <mi>x</mi>
47           </mrow>
48        </math>
49     </body>
50  </html>
```

Entity **δ** represents a delta symbol.



Program Output



Outline

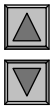
```
1   <?jumbo:namespace ns = "http://www.xml-cml.org"
2      prefix = "C" java = "jumbo.cmlxml.*Node" ?>
3
4   <!-- Fig. 20.12 : ammonia.xml -->
5   <!-
6
7   <C:
8
9      <C:atomArray builtin = "elsym">
10        N H H H
11     </
12
13     <
14
15     </C:atomArray>
16
17     <C:atomArray builtin =
18        1.5 1.5 0.0 1.5
19     </C:atomArray>
20
21     <C:bondArray builtin = "atid1">
22        1 1 1
23     </C:bondArray>
24
25     <C:bondArray builtin = "atid2">
26        2 3 4
27     </C:bondArray>
28
29     <C:bondArray builtin = "order" type = "integer">
30        1 1 1
31     </C:bondArray>
32
33  </C:molecule>
```

A value of **x2** and **type float** specifies that the element contains a list of floating-point numbers, each of which indicates the *x*-coordinate of an atom.

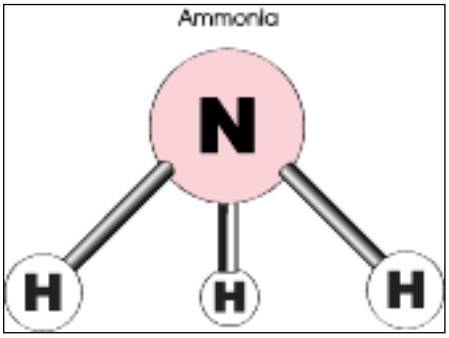
lication-specific
ent.

A value of **y2** specifies that the element contains a list of *y*-coordinate values.

Element **bondArray** defines the bonds between atoms.



Outline



Program Output

20.7 DOM Methods

Markup Language	Description
VoiceXML	The VoiceXML forum founded by AT&T, IBM, Lucent and Motorola developed VoiceXML. It provides interactive voice communication between humans and computers through a telephone, PDA (Personal Digital Assistant) or desktop computer. IBM's VoiceXML SDK can process VoiceXML documents. Visit **www.voicexml.org** for more information on VoiceXML. We introduce VoiceXML in Chapter 34, Accessibility.
Synchronous Multimedia Integration Language (SMIL)	SMIL is an XML vocabulary for multimedia presentations. The W3C was the primary developer of SMIL, with contributions from other companies. Visit **www.w3.org/AudioVideo** for more on SMIL. We introduce SMIL in Chapter 33, Multimedia.
Research Information Exchange Markup Language (RIXML)	RIXML, which a consortium of brokerage firms developed, marks up investment data. Visit **www.rixml.org** for more information on RIXML.
ComicsML	A language developed by Jason MacIntosh for marking up comics. Visit **www.jmac.org/projects/comics_ml** for more information on ComicsML.
Geography Markup Language (GML)	The OpenGIS developed the Geography Markup Language to describe geographic information. Visit **www.opengis.org** for more information on GML.
Extensible User Interface Language (XUL)	The Mozilla project created the Extensible User Interface Language for describing graphical user interfaces in a platform-independent way. For more information visit: **www.mozilla.org/xpfe/language-Spec.html**

Fig. 20.13 Various markup languages derived from XML.

20.7 DOM Methods

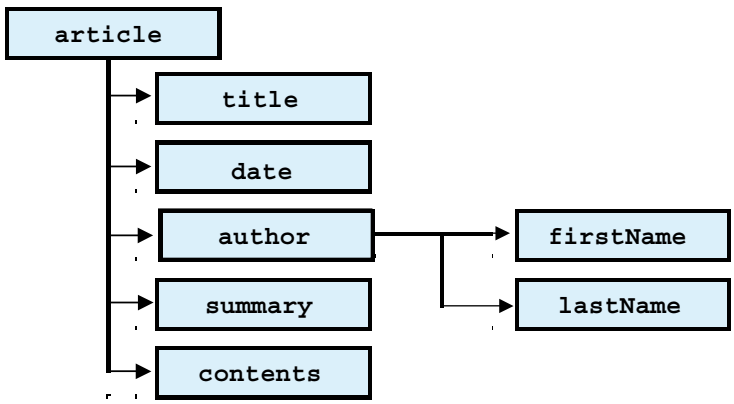
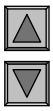


Fig. 20.14 Tree structure for article.xml.



Outline

DOMExample.html

```
1   <?xml version="1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4   <html xmlns="http://www.w3.org/1999/xhtml">
5
6   <!-- Fig. 20.15 : DOMExa
7   <!-- DOM with JavaScript
8
9      <head>
10        <title>A DOM Example</title>
11     </head>
12
13     <body>
14
15     <script type = "text/
16        <!--
17        var xmlDo
18
19        xmlDocument.load( "article.xml" );
20
21        // ge
22        var
23
24        document.writeln(
25           "<p>Here is the root node of the document: " +
26           "<strong>" + element.nodeName + "</strong>" +
27           "<br />The following are its child elements:" +
28           "</p><ul>" );
29
30        // traverse all child nodes of root element
31        for ( var i = 0; i < element.childNodes.length; i++ ) {
32           var curNode = element.childNodes.item( i );
33
```

Instantiate a Microsoft XML Document Object Model object and assign it to reference **xmlDocument**.

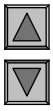
method **load** loads **article.xml** (Fig. 20.1) into memory.

Property **documentElement** corresponds to the root

Property **nodeName** corresponds to the name of an element, attribute, etc.

Method **item** to obtain the child node at index **i**.

y **childNodes**.



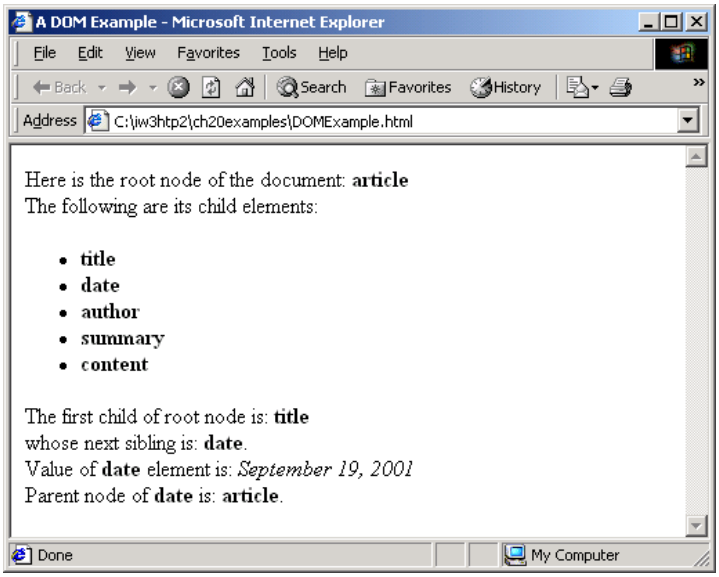
Outline

DOMExample.html

```
34           // print node name of each child element
35           document.writeln( "<li><strong>" + curNode.nodeName
36
37        }
38
39        docum
40
41        // get the first child node of root element
42        var currentNode = element.firstChild;
43
44        document.writeln( "<p>The first child of root node is: " +
45           "<strong>" + currentNode.nodeName + "</strong>" +
46           "<br />whose next sibling is:" );
47
48        // get the next sibling of first child
49        var nextSib = currentNode.nextSibling;
50
51        document.writeln( "<strong>" + nextSib.nodeName +
52           "</strong>.<br />Value of <strong>" +
53           nextSib.nodeName +
54
55        var value = nextSib.
56
57        // print the text value of the sibling
58        document.writeln( "<em>" + value.nodeValue + "</em>" +
59           "<br />Parent node of <strong>" + nextSib.nodeName +
60           "</strong> is: <strong>" +
61           nextSib.parentNode.nodeName + "</strong>.</p>" );
62           -->
63     </script>
64
65     </body>
66  </html>
```

Retrieve the root node's first child node (i.e., **title**) using property **firstChild**.

Property **parentNode** returns a node's parent node.



Program Output

20.7 DOM Methods

Method	Description
`getNodeType`	Returns an integer representing the node type.
`getNodeName`	Returns the name of the node. If the node does not have a name, a string consisting of `#` followed by the type of the node is returned.
`getNodeValue`	Returns a string or null depending on the node type.
`getParentNode`	Returns the parent node.
`getChildNodes`	Returns a `NodeList` (Fig. 20.17) with all the children of the node.
`getFirstChild`	Returns the first child in the `NodeList`.
`getLastChild`	Returns the last child in the `NodeList`.
`getPreviousSibling`	Returns the node preceding this node, or null.
`getNextSibling`	Returns the node following this node, or null.
`getAttributes`	Returns a `NamedNodeMap` (Fig. 20.18) containing the attributes for this node.
`insertBefore`	Inserts the node passed as the first argument before the existing node passed as the second argument. If the new node is already in the tree, it is removed before insertion. The same behavior is true for other methods that add nodes.
`replaceChild`	Replaces the second argument node with the first argument node.
`removeChild`	Removes the child node passed to it.
`appendChild`	Appends the node passed to it to the list of child nodes.
`getElementsByTagName`	Returns a `NodeList` of all nodes in the subtree with the name specified as the first argument ordered as they would be encountered in a preorder traversal. An optional second argument specifies either the direct child nodes (`0`) or any descendant (`1`).
`getChildAtIndex`	Returns the child node at the specified index in the child list.
`addText`	Appends the string passed to it to the last `Node` if it is a `Text` node, otherwise creates a new `Text` node for the string and adds it to the end of the child list.
`isAncestor`	Returns `true` if the node passed is a parent of the node, or is the node itself.

Fig. 20.16 Some `Node` object methods.

20.7 DOM Methods

Method	Description
`item`	Passed an index number, returns the element node at that index. Indices range from 0 to *length* − *1*.
`getLength`	Returns the total number of nodes in the list.

Fig. 20.17 Some `NodeList` methods.

Method	Description
`getNamedItem`	Returns either a node in the `NamedNodeMap` with the specified name or null.
`setNamedItem`	Stores a node passed to it in the `NamedNodeMap`. Two nodes with the same name cannot be stored in the same `NamedNodeMap`.
`removeNamedItem`	Removes a specified node from the `NamedNodeMap`.
`getLength`	Returns the total number of nodes in the `NamedNodeMap`.
`getValues`	Returns a `NodeList` containing all nodes in the `NamedNodeMap`.

Fig. 20.18 Some `NamedNodeMap` methods.

20.7 DOM Methods

Method	Description
`getDocumentElement`	Returns the root node of the document.
`createElement`	Creates and returns an element node with the specified tag name.
`createAttribute`	Creates and returns an attribute node with the specified name and value.
`createTextNode`	Creates and returns a text node that contains the specified text.
`createComment`	Creates a comment to hold the specified text.

Fig. 20.19 Some `Document` methods.

Method	Description
`getTagName`	Returns the name of the element.
`setTagName`	Changes the name of the element to the specified name.
`getAttribute`	Returns the value of the specified attribute.
`setAttribute`	Changes the value of the attribute passed as the first argument to the value passed as the second argument.
`removeAttribute`	Removes the specified attribute.
`getAttributeNode`	Returns the specified attribute node.
`setAttributeNode`	Adds a new attribute node with the specified name.

Fig. 20.20 Some `Element` methods.

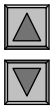
20.7 DOM Methods

Method	Description
`getValue`	Returns the specified attribute's value.
`setValue`	Changes the value of the attribute to the specified value.
`getName`	Returns the name of the attribute.

Fig. 20.21 Some `Attr` methods.

Method	Description
`getData`	Returns the data contained tin the node (text or comment).
`setData`	Sets the node's data.
`getLength`	Returns the number of characters contained in the node.

Fig. 20.22 Some `Text` and `Comment` methods.



Outline

Games.xml

```
1   <?xml version = "1.0"?>
2   <?xml:stylesheet type = "text/xsl" href = "games.xsl"?>
3
4   <!-- Fig. 20.23 : games.xml -->
5   <!-- Sports Database        -->
6
7   <sports>
8
9      <game id = "783">
10        <name>Cricket</name>
11
12        <paragraph>
13           More popular among commonwealth nations.
14        </paragraph>
15     </game>
16
17     <game id = "239">
18        <name>Baseball</name>
19
20        <paragraph>
21           More popular in America.
22        </paragraph>
23     </game>
24
25     <game id = "418">
26        <name>Soccer (Football)</name>
27
28        <paragraph>
29           Most popular sport in the world.
30        </paragraph>
31     </game>
32
33  </sports>
```

A process… stylesheet **games.xsl**.

Value **type** specifies that **games.xsl** is a **text/xsl** file.

Outline

Program Output

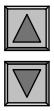
Outline

```
1   <?xml version = "1.0"?>
2
3   <!-- Fig. 20.2
4   <!-- A simple
5
6   <!-- reference XSL style
7   <xsl:stylesheet version
8      xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
9
10     <xsl:output method = "html
11        doctype-system =
12           "http://www.w3.org/T
13        doctype-public = "-//W3C//DTD XHTML 1.0 Strict//EN"/>
14
15  <xsl:template match = "/">
16
17    <html xmlns="http://www.w3.org/1999/xhtml">
18
19      <head>
20         <title>Sports</title>
21      </head>
22
23      <body>
24
25         <table border = "1" bgcolor = "cyan">
26
27         <thead>
28
29           <tr>
30             <th>ID</th>
31             <th>Sport</th>
32             <th>Information</th>
33           </tr>
34
35         </thead>
```

The **stylesheet** start tag—which begins the XSL stylesheet.

Element **xsl:output** writes an XHTML document type declaration to the result tree.

The **match** attribute to select the document root of the source document (i.e., **game.xml**).



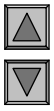
Outline

Elements.xsl

```
36
37          <!-- insert each name and paragraph element value  -->
38          <!-- into a table row.                              -->
39          <xsl:for-each select = "sports/game">
40
41             <tr>
42               <td><xsl:value-of select = "@id"/></td>
43               <td><xsl:value-of select = "name"/></td>
44               <td><xsl:value-of select = "paragraph"/></td>
45             </tr>
46
47          </xsl:for-each>
48
49          </table>
50
51       </body>
52
53    </html>
54
55 </xsl:template>
56
57 </xsl:stylesheet>
```

Element **xsl:for-each** iterates through the source XML document and search for **game** elements.

Element **value-of** retrieves attribute **id**'s value and place it in a **td** element in the result tree.

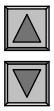


Outline

Sorting.xml

```
1  <?xml version = "1.0"?>
2
3  <!-- Fig. 20.25 : sorting.xml         -->
4  <!-- Usage of elements and attributes -->
5
6  <?xml:stylesheet type = "text/xsl" href = "sorting.xsl"?>
7
8  <book isbn = "999-99999-9-X">
9     <title>Deitel&apos;s XML Primer</title>
10
11    <author>
12       <firstName>Paul</firstName>
13       <lastName>Deitel</lastName>
14    </author>
15
16    <chapters>
17       <frontMatter>
18          <preface pages = "2"/>
19          <contents pages = "5"/>
20          <illustrations pages = "4"/>
21       </frontMatter>
22
23       <chapter number = "3" pages = "44">
24          Advanced XML</chapter>
25       <chapter number = "2" pages = "35">
26          Intermediate XML</chapter>
27       <appendix number = "B" pages = "26">
28          Parsers and Tools</appendix>
29       <appendix number = "A" pages = "7">
30          Entities</appendix>
31       <chapter number = "1" pages = "28">
32          XML Fundamentals</chapter>
33    </chapters>
```

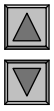
Reference to the XSL stylesheet **sorting.xsl.**



Outline

Sorting.xml

```
34
35       <media type = "CD"/>
36    </book>
```

Outline

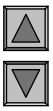
Sorting.xsl

```
1     <?xml version = "1.0"?>
2
3     <!-- Fig. 20.26 : sorting.xsl                          -->
4     <!-- Transformation of Book information into XHTML -->
5
6     <xsl:stylesheet version = "1.0"
7        xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
8
9        <xsl:output method = "ht
10          doctype-system =
11             "http://www.w3.org
12          doctype-public = "-//
13
14       <xsl:template match = "/">
15
16          <html xmlns = "http://www.w3.org/1999/xhtml">
17             <xsl:apply-templates/>
18          </html>
19       </xsl:template>
20
21       <xsl:template match = "book">
22          <head>
23             <title>ISBN <xsl:value-of
24                <xsl:value-of select =
25          </head>
26
27          <body>
28             <h1><xsl:value-of select = "title"/></h1>
29
30             <h2>by <xsl:value-of select = "author/lastName"/>,
31                <xsl:value-of select = "author/firstName"/></h2>
32
```

Specify that the msxml processor should apply **xsl:template**s to the document root's children.

Create the title for the XHTML document.

Create a header element that displays the book's author.



Outline

Sorting.xsl

```
33          <table border = "1">
34             <xsl:for-each select = "chapters/frontMatter/*">
35                <tr>
36                   <td align = "right">
37
38
39
40                      <td>
41                         ( <xsl:value-of select = "@pages"/> pages )
42                      </td>
43                   </tr>
44                </xsl:for
45
46                <xsl:for-each select = "chapters/chapter">
47                   <xsl:sort select = "@number" data-type = "number"
48                      order = "ascending"/>
49                   <tr>
50                      <td al
51                         Cha
52                      </td>
53
54                      <td>
55                         ( <xsl:value-of select = "@pages"/> pages )
56                      </td>
57                   </tr>
58                </xsl:for-each>
59
60                <xsl:for-each select = "chapters/appendix">
61                   <xsl:sort select = "@number" data-type = "text"
62                      order = "ascending"/>
63                   <tr>
64                      <td align = "right">
65                         Appendix <xsl:value-of select = "@number"/>
66                      </td>
67
```

Select each element (indicated by an asterisk) that is a child of **frontMatter**.

Call node-set function **name** to retrieve the current node's element name (e.g., **preface**).

Attribute **select** in context node **ch**

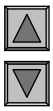
Attribute **data-type** specifies a numeric sort and attribute ***order*** specifies **ascending** order.

Outline

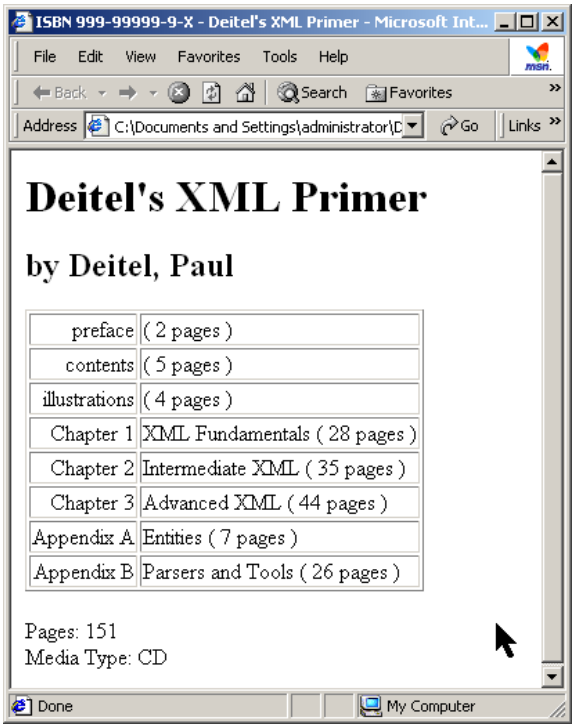
Sorting.xsl

```
68                      <td>
69
70
71
72                   </xsl:for-each>
73                </table>
74
75                <br />Pages:
76                   <xsl:variable name = "pagecount"
77                      select = "sum(chapters//*/@pages)"/>
78                   <xsl:value-of select = "$pagecount"/>
79                <br />Media Type: <xsl:value-of select = "media/@type"/>
80             </body>
81          </xsl:template>
82
83       </xsl:stylesheet>
```

Use an XSL variable to store the value of the book's page count and output the page count to the result tree.



Program Output



20.10 Microsoft BizTalk

BizTalk	Description
Framework	A specification that defines message formats.
Schema library	A repository of Framework XML schemas.
Server	An application that helps vendors convert their messages to BizTalk format. For more information visit: `www.microsoft.com/biztalkserver`.
JumpStart Kit	A set of tools for developing BizTalk applications.

Fig. 20.27 BizTalk Terminologies.

Outline

BizTalkexample.xml

```
1   <?xml version = "1.0"?>
2   <BizTalk
3    xmlns = "urn:sche
4
5   <!-- Fig. 20.28: BizTalkexample.xml -->
6   <!-- BizTalk example                -->
7   <Route>
8      <From locationID = "
9            handle = "23"
10
11     <To locationID = "454545445" locationType = "DUNS"
12           handle = "45" />
13  </Route>
14
15  <Body>
16   <Offers xmlns =
17   "x-schema:http://schemas.biztalk.org/eshop_msn_com/t7ntoqnq.xml">
18     <Offer>
19        <Model>12-a-3411d</Model>
20        <Manufacturer>ExComp, Inc.</Manufacturer>
21        <ManufacturerModel>DCS-48403</ManufacturerModel>
22        <MerchantCategory>Clothes | Sports wear</MerchantCategory>
23        <MSNClassId></MSNClassId>
24        <StartDate>2000-06-05 T13:12:00</StartDate>
25        <EndDate>2000-12-05T13:12:00</EndDate>
26        <RegularPrice>89.99</RegularPrice>
27        <CurrentPrice>25.99</CurrentPrice>
28        <DisplayPrice value = "3" />
29        <InStock value = "15" />
30        <ReferenceImageURL>
31           http://www.Example.com/clothes/index.jpg
32        </ReferenceImageURL>
33        <OfferName>Clearance sale</OfferName>
34        <OfferDescription>This is a clearance sale</OfferDescription>
35        <PromotionalText>Free Shipping</PromotionalText>
```

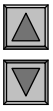
Element **Route** contains the routing information.

Element **From** specifies the document's source.

Defines a default namespace for the BizTalk framework elements.

Element **Body** contains the actual message, whose schema the businesses define.

Element **To** specifies the document's destination.



Outline

BizTalkexample.xml

```
36        <Comments>Clothes that you would love to wear.</Comments>
37        <IconType value = "BuyNow" />
38        <ActionURL>http://www.example.com/action.htm</ActionURL>
39        <AgeGroup1 value = "Infant" />
40        <AgeGroup2 value = "Adult" />
41        <Occasion1 value = "Birthday" />
42        <Occasion2 value = "Anniversary" />
43        <Occasion3 value = "Christmas" />
44      </Offer>
45     </Offers>
46  </Body>
47  </BizTalk>
```