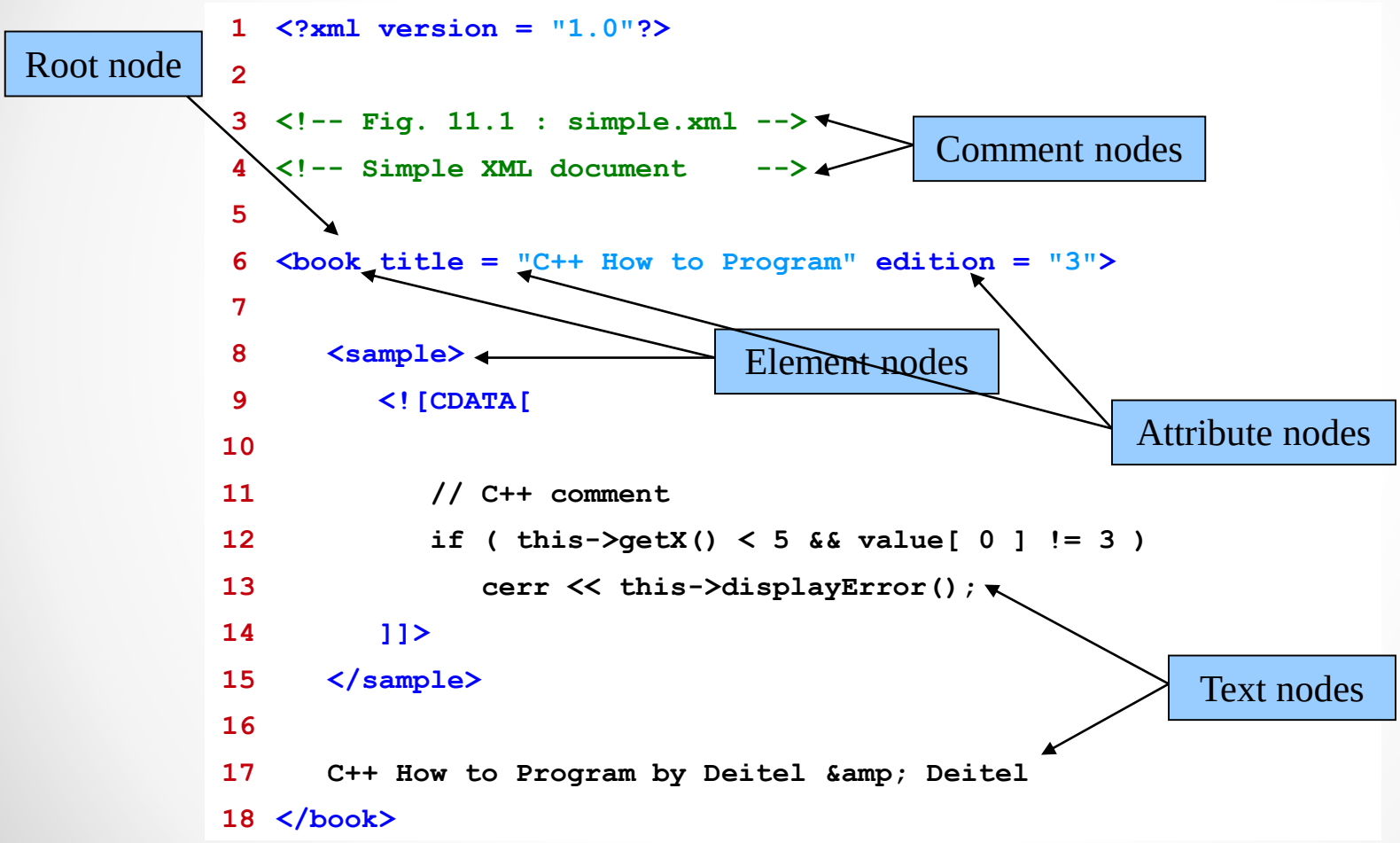
XPath

XPath Nodes




```
1  <?xml version = "1.0"?>
2
3  <!-- Fig. 11.1 : simple.xml -->
4  <!-- Simple XML document    -->
5
6  <book title = "C++ How to Program" edition = "3">
7
8     <sample>
9        <![CDATA[
10
11          // C++ comment
12          if ( this->getX() < 5 && value[ 0 ] != 3 )
13             cerr << this->displayError();
14       ]]>
15    </sample>
16
17    C++ How to Program by Deitel &amp; Deitel
18 </book>
```

Relationship of Nodes

```
<bookstore>
        <book>
                <title>Harry Potter</title>
                <author>J K. Rowling</author>
                <year>2005</year>
                <price>29.99</price>
        </book>
</bookstore>
```

- Parent
  - Each element and attribute has one parent.
  - In the above example; the book element is the parent of the title, author, year, and price.
- Children
  - Element nodes may have zero, one or more children.
  - In the above example; the title, author, year, and price elements are all children of the book element.

Relationship of Nodes

```
<bookstore>
        <book>
                <title>Harry Potter</title>
                <author>J K. Rowling</author>
                <year>2005</year>
                <price>29.99</price>
        </book>
</bookstore>
```

- Siblings
  - Nodes that have the same parent.
  - In the above example; the title, author, year, and price elements are all siblings.
- Ancestors
  - A node's parent, parent's parent, etc.
  - In the above example; the ancestors of the title element are the book element and the bookstore element.
- Descendants
  - A node's children, children's children, etc.
  - In the above example; descendants of the bookstore element are the book, title, author, year, and price elements.

Selecting Nodes

XPath uses path expressions to select nodes in an XML document. The node is selected by following a path or steps. The most useful path expressions are listed below:

Expression	Description
nodename	Selects all nodes with the name "*nodename*"
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Selecting Nodes

The table below have some path expressions and the result of the expressions:

Path Expression	Result
bookstore	Selects all nodes with the name "bookstore"
/bookstore	Selects the root element bookstore **Note:** If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

Predicates

- Predicates are used to find a specific node or a node that contains a specific value.
- Predicates are always embedded in square brackets.

Path Expression	Result
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element.**Note:** In IE 5,6,7,8,9 first node is[0], but according to W3C, it is [1]. To solve this problem in IE, set the SelectionLanguage to XPath: *In JavaScript: xml*.setProperty("SelectionLanguage","XPath");
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='eng']	Selects all the title elements that have an attribute named lang with a value of 'eng'
//title[not(@lang)]	Selects all the title elements that do not have attribute named lang
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
/bookstore/book[price>35.00]/title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

```
var xml = loadXMLDoc("books.xml");
path = "/bookstore/book/title";

if (xml.evaluate != undefined) {
    // code does not work for IE 11 because there is no support for XPATH in
    // ie11 yet
    xml.setProperty("SelectionLanguage", "XPath");
    nodes = xml.selectNodes(path);
    for (i = 0; i < nodes.length; i++) {
        document.write(nodes[i].childNodes[0].nodeValue);
        document.write("<br>");
    }
}
else
{
    // code for Chrome, Firefox, Opera, etc.
    //evalute(xpath, context, namespace, type, result)
    var nodes = xml.evaluate(path, xml, null, XPathResult.ANY_TYPE, null);
    var result = nodes.iterateNext();

    while (result) {
        document.write(result.childNodes[0].nodeValue);
        document.write("<br>");
        result = nodes.iterateNext();
    }
}
```

Selecting Unknown Nodes

XPath wildcards can be used to select unknown XML elements.

Wildcard	Description
*	Matches any element node
@*	Matches any attribute node
node()	Matches any node of any kind

In the table below we have listed some path eprexssions and the result of the expressions:

Path Expression	Result
/bookstore/*	Selects all the child nodes of the bookstore element
//*	Selects all elements in the document
//title[@*]	Selects all title elements which have any attribute
//title[not(@*)]	Selects all the title elements that do not have any attribute

Selecting Several Paths

By using the | operator in an XPath expression you can select several paths.

In the table below we have listed some path expressions and the result of the expressions:

Path Expression	Result
//book/title | //book/price	Selects all the title AND price elements of all book elements
//title | //price	Selects all the title AND price elements in the document
/bookstore/book/title | //price	Selects all the title elements of the book element of the bookstore element AND all the price elements in the document

XPath Operators

Operator	Description	Example
|	Computes two node-sets	//book | //cd
+	Addition	6 + 4
-	Subtraction	6 - 4
*	Multiplication	6 * 4
div	Division	8 div 4
=	Equal	price=9.80
!=	Not equal	price!=9.80
<	Less than	price<9.80
<=	Less than or equal to	price<=9.80
>	Greater than	price>9.80
>=	Greater than or equal to	price>=9.80
or	or	price=9.80 or price=9.70
and	and	price>9.00 and price<9.90
mod	Modulus (division remainder)	5 mod 2