Extensible Markup Language (XML)

XML permits document authors to create markup (i.e., a text-based notation for describing data) for virtually any type of information,

enabling them to create entirely new markup languages for describing any type of data, such as mathematical formulas, software configuration instructions, chemical molecular structures, music, news, recipes and financial reports.

XML describes data in a way that human beings can understand and computers can process.

Example:

XML document that describes information for a baseball player.

<player>

<firstName>**John**</firstName>
<lastName>**Doe**</lastName>
<battingAverage>**0.375**</battingAverage>

</player>

XML documents contain text that represents content (i.e., data), such as John

and elements that specify the document's structure, such as firstName

XML documents delimit elements with start tags and end tags.
A start tag consists of the element name in angle brackets (e.g., <player> and <firstName>)

An end tag consists of the element name preceded by a forward slash (/) in angle brackets
(e.g., </firstName> and </player>)

Every XML document must have exactly one **root element** that contains all the other elements.

XML Vocabularies

XML-based markup languages—called XML vocabularies—provide a means for describing particular types of data in standardized, structured ways.

- XHTML (ExtensibleHyperText Markup Language)
- MathML™(for mathematics)
- VoiceXML™ (for speech)-used in IVR system
- CML (Chemical Markup Language—for chemistry),
- XBRL (Extensible Business Reporting Language—for financial data exchange)

Viewing and Modifying XML Documents

- XML documents are highly portable.
- Viewing or modifying an XML document—which is a text file that usually ends with the .xml filename extension.
- Any text editor that supports ASCII/Unicode characters can open XML documents for viewing and editing.
- Also, most web browsers can display XML documents in a formatted manner that shows the XML's structure

Processing XML Documents

Processing an XML document requires software called an XML parser (or XML processor).

A parser makes the document's data available to applications. While reading an XML document's contents, a parser checks that the document follows the syntax rules specified by the W3C's XML Recommendation.

XML syntax requires a single root element, a start tag and end tag for each element, and properly nested tags (i.e., the end tag for a nested element must appear before the end tag of the enclosing element).

- XML is case sensitive, so the proper capitalization must be used in elements.
- A document that conforms to this syntax is a well-formed XML document and is syntactically correct.
- If an XML parser can process an XML document successfully, that XML document is well-formed.
- Parsers can provide access to XML-encoded data in well-formed documents only.
- XML parsers are often built into browsers and other software.

Validating XML Documents

An XML document can reference a Document Type Definition (DTD) or a schema that defines the document's proper structure.

When an XML document references a DTD or a schema, some parsers (called validating parsers) can read it and check that the XML document follows the structure it defines.

If the XML document conforms to the DTD/schema (i.e., has the appropriate structure), the document is valid.

Structuring Data

```
1 <?xml version = "1.0"?>
2 <article>
3       <title>Simple XML</title>
4       <date>July 4, 2007</date>
5       <author>
6             <firstName>John</firstName>
7             <lastName>Doe</lastName>
8       </author>
9       <summary>XML is pretty easy.</summary>
10      <content>This chapter presents examples that use </content>
11 </article>
```

XML Declaration

This document begins with an XML declaration (line 1), which identifies the document as an XML document.

Root Node

In example, article (lines 2–11) is the root element.

XML Element Names

The elements we use in the example do not come from any specific markup language. Instead, we chose the element names and markup structure that best describe our particular data

XML element names can be of any length and may contain letters, digits, underscores, hyphens and periods.

However, they must begin with either a letter or an underscore, and they should not begin with “xml” in any combination of uppercase and lowercase letters

Nesting XML Elements

XML elements are nested to form hierarchies—with the root element at the top of the hierarchy.

This allows document authors to create parent/child relationships between data items.

For example, elements title, date, author, summary and content are children of article. Elements firstName and lastName are children of author.

Any element that contains other elements (e.g., article or author) is a container element.

Container elements also are called parent elements. Elements nested inside a container element are child elements (or children) of that container element.

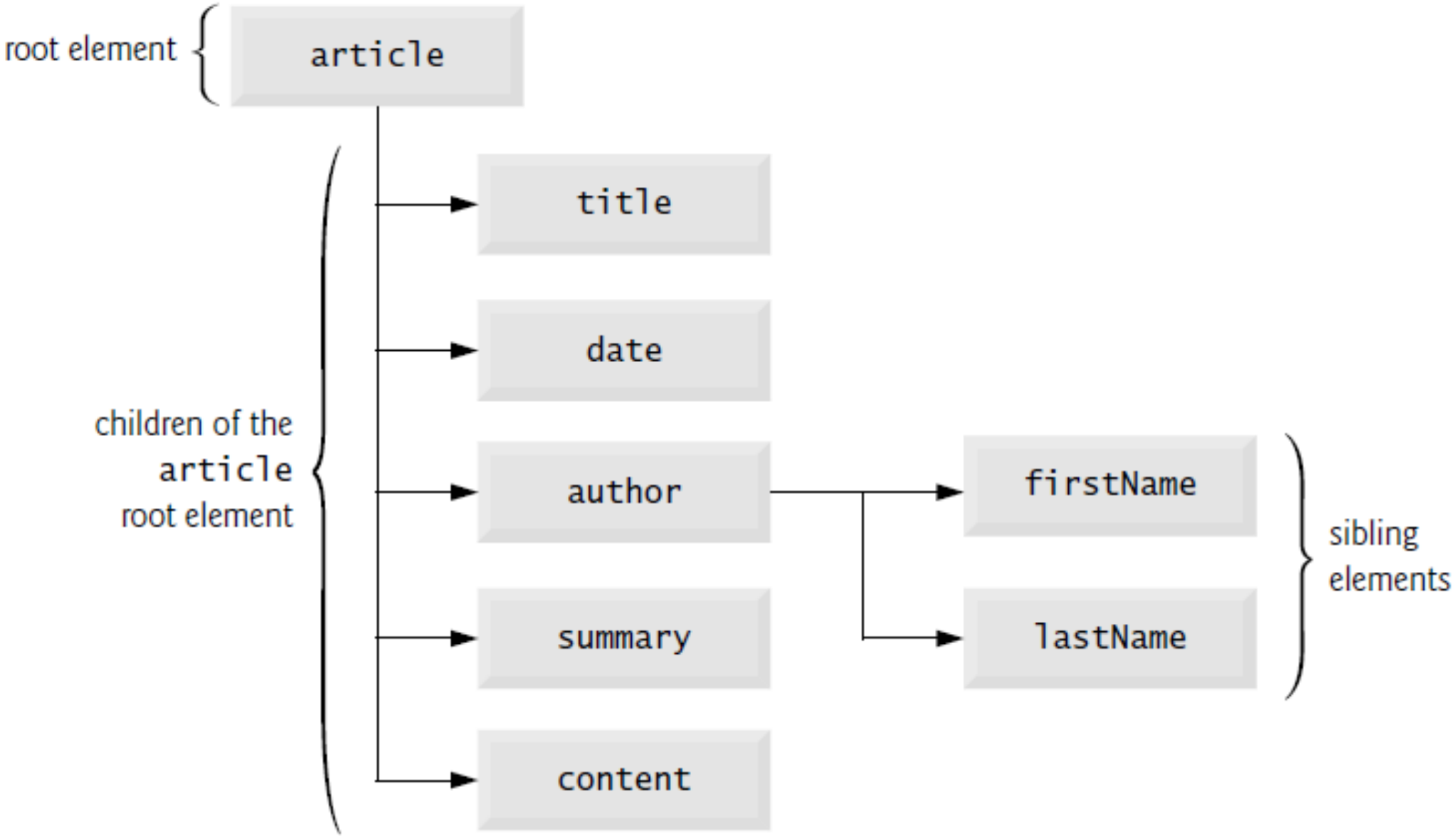
If those child elements are at the same nesting level, they’re siblings of one another.

Viewing an XML Document in a Web Browser

The XML document in Fig. 15.2 is simply a text file named article.xml.

It does not contain formatting information for the article. This is because XML is a technology for describing the structure of data.

The formatting and displaying of data from an XML document are application-specific issues.


root element
article
children of the
article
root element
title
date
author
summary
content
firstName
lastName
sibling
elements

The Difference Between XML and HTML

XML and HTML were designed with different goals:

XML was designed to carry data - with focus on what data is

HTML was designed to display data - with focus on how data looks

XML tags are not predefined like HTML tags are

With XML, the author must define both the tags and the document structure.

XML Namespaces

XML allows document authors to create custom elements. This extensibility can result in **naming collisions** among elements in an XML document that have the same name.

<subject>Geometry**</subject>**
<subject>Cardiology**</subject>**

<highschool:subject>Geometry**</highschool:subject>**
<medicalschool:subject>Cardiology**</medicalschool:subject>**

Both highschool and medicalschool are **namespace prefixes**.