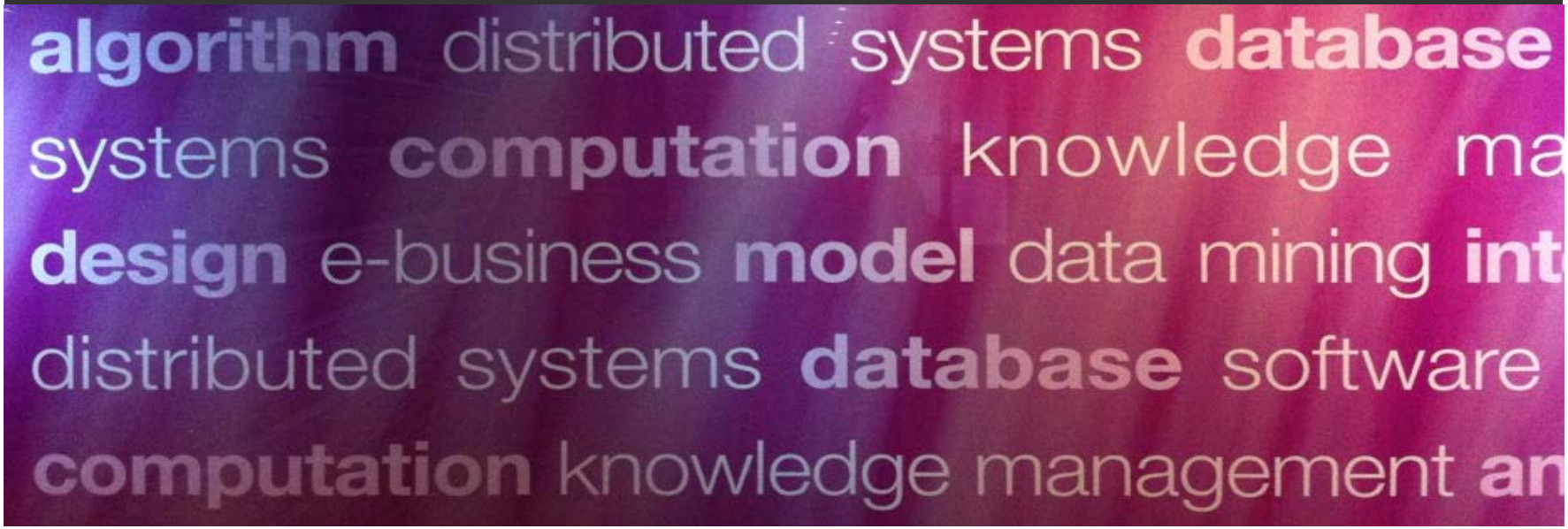
MONASH University

Information Technology

FIT3176 Advanced Database Design

Topic 7: XML documents

Dr. Minh Le
Minh.Le@monash.edu



*Adapted from slides developed by A/Prof David Taniar and Lindsay Smith

This unit covers…

1. Advanced Database Design
   - E/R is not complete enough
   - EER, covering superclass and subclass

2. SQL and PL/SQL
   - Trigger, Procedures/Functions, Packages

3. **XML and XML DB**
   - **XML** (*this week*)
   - XML Schema (week 8)
   - XML Database (week 9)

4. JSON and JSON DB
   - JSON (week 10)
   - JSON Database (week 11)

Learning Objectives

By the end of this topic you should be able to:

- describe the difference between well formed and valid XML documents
- describe the difference between a validating and non validating XML parser
- explain the structure of an XML document
- list the six types of markup which can occur in an XML document
- create an XML document given a set of data
- use a browser to display an XML document
- describe the problem of name collision in compound documents
- declare a namespace for an XML vocabulary
- apply a namespace to an element and an attribute

References

- Carey, P., *New Perspectives on XML 2nd Edition Comprehensive*, 2007, Thomson Course Technology, Tutorial 1 & 2 (Available in the library)
- Carey, P. and Vodnik, S., *New Perspectives on XML 3rd Edition Comprehensive*, 2015, Thomson Course Technology
- Coronel & Morris, Database Systems: Design, Implementation & Management, 11th Edition 2015, Thomson Course Technology. Chapter 14, Section 14.3 or 12th Edition 2016 Chapter 15, Section 15-3

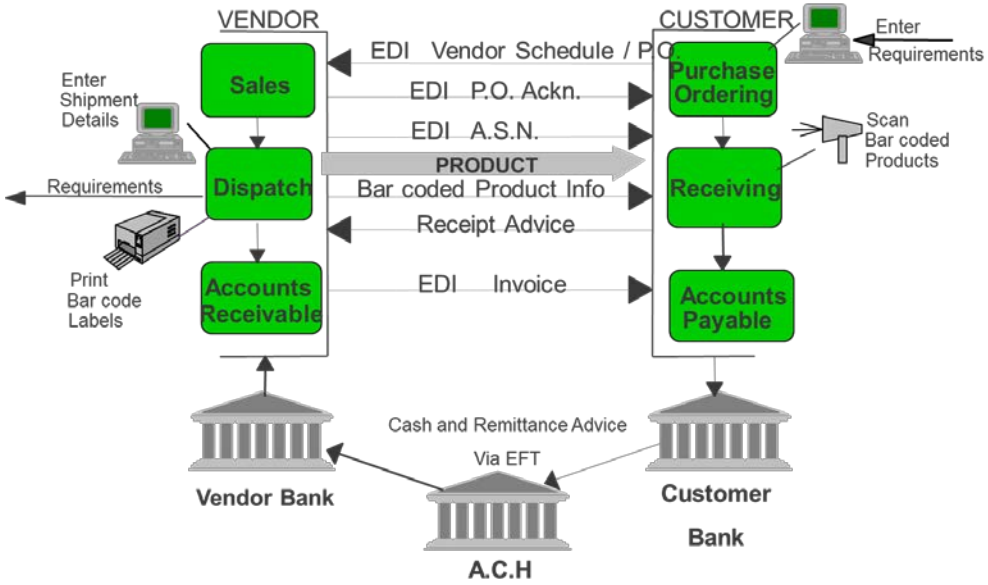
Main Reference:

- Carey, P., *New Perspectives on XML 2nd Edition Comprehensive*, 2007, Thomson Course Technology, Tutorial 1 & 2, **pages 1-88**, (Available in the library)

1. The Structure of an XML Document (p11)
2. Working with Elements (p13)
3. Working with Attributes (p19)
4. Using Character and Entity References (p21)
5. Understanding Text Characters (p24)
6. Processing an XML Document (p28)
7. Formatting XML Data (p32)
8. Inserting a Processing Instruction (p34)
9. Combining XML Vocabularies (p48)
10. Working with Namespaces (p56)
11. Working with Attributes (p61)
12. Adding a Namespace to a Style Sheet (p63)
13. Combining Standard Vocabularies (p67)

Introduction to XML (1)

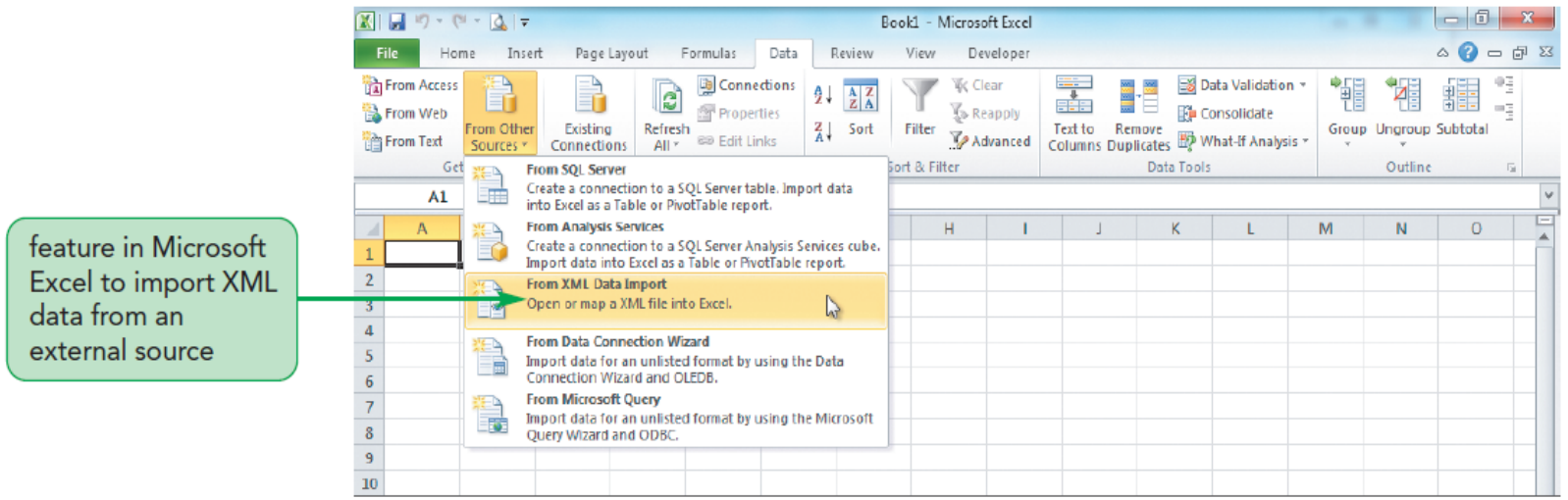
- Extensible Markup Language (XML) is a meta-language used to represent and manipulate data elements.
- Subset of Standard Generalised Markup Language (SGML), a language introduced in the 1980s that describes the structure and content of any machine readable information. SGML is device-independent and system-independent.
- Designed to facilitate the exchange of structured documents, such as orders and invoices, over the Internet.



Introduction to XML (2)

- Exchange data among other software applications e.g., as of 2007 releases of Microsoft Office and Open Office, users can exchange data among Office applications and enterprise systems using XML.

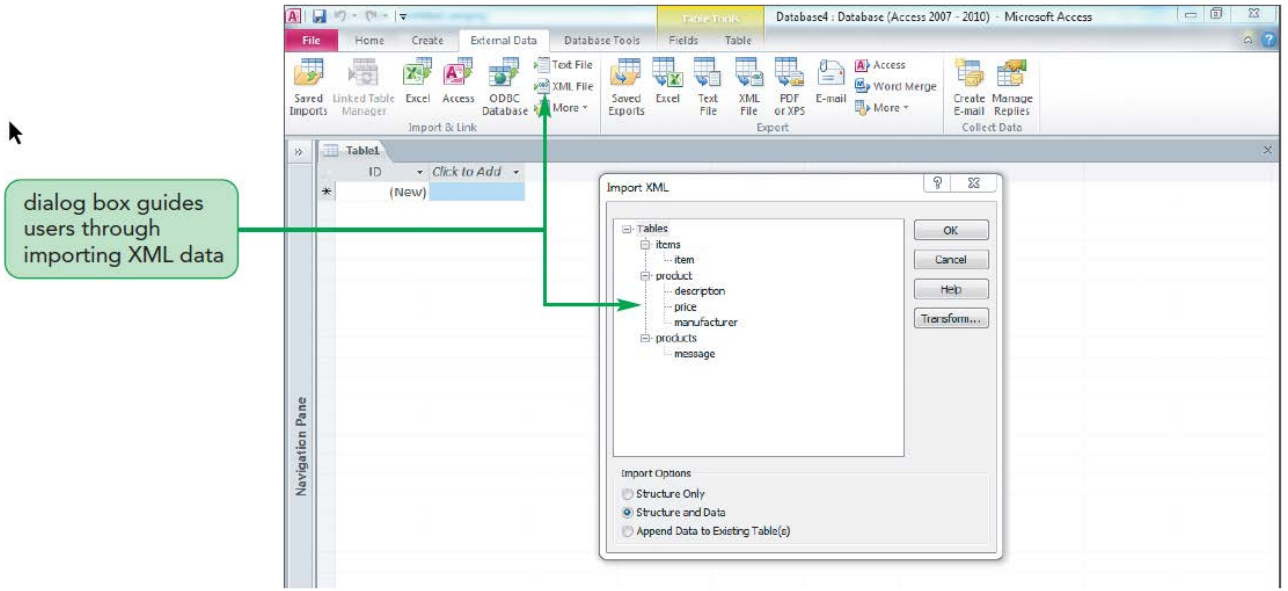
Figure 1-2 Importing XML data into Microsoft Excel



Introduction to XML (3)

- Databases store data, and XML is widely used for data interchange.
  - XML and relational databases are tightly working together in most web applications. Figure 1-3 shows how Access has incorporated easy XML importing and exporting of data.

Figure 1-3 Importing XML-formatted data into Access



Introduction to XML(4)

- XML and Mobile Development
  - A popular use of XML in the iPhone is in a preference list as p-list to organise data into named properties and lists of values, as shown in Figure 1-4.
  - Android uses XML for screen layout and for working with data as shown in Figure 1-5.

Figure 1-4 iOS p-list file written in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
        "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
   <dict>
      <key>Accessories</key>
      <string>Collar Tag</string>
      <key>Shirt</key>
      <string>Week Day</string>
      <key>Bowls</key>
      <string>Ceramic 2 Holder</string>
   </dict>
</plist>
```

Figure 1-5 Android layout definitions written in XML

Introduction to XML(5)

- XML can be used to create XML applications or vocabularies, which are markup languages tailored to contain specific pieces of information
  - One of the more important XML vocabularies on the Internet is Really Simple Syndication (RSS), which is the language used for distributing news articles or any content that changes on a regular basis. Figure 1-8 shows a segment of an RSS document.

Figure 1-8 RSS document

Introduction to XML(6)

- Figure 1-9 lists a few of the many vocabularies that have been developed using XML.

Figure 1-9 XML vocabularies

XML Vocabulary	Description
Bioinformatic Sequence Markup Language (BSML)	Coding of bioinformatic data
Extensible Hypertext Markup Language (XHTML)	HTML written as an XML application
Mathematical Markup Language (MathML)	Presentation and evaluation of mathematical equations and operations
Music Markup Language (MML)	Display and organization of music notation and lyrics
Weather Observation Definition Format (OMF)	Distribution of weather observation reports, forecasts, and advisories
Really Simple Syndication (RSS)	Distribution of news headlines and syndicated columns
Synchronized Multimedia Integration Language (SMIL)	Editing of interactive audiovisual presentations involving streaming audio, video, text, and any other media type
Voice Extensible Markup Language (VoiceXML)	Creation of audio dialogues that feature synthesized speech, digitized audio, and speech recognition
Wireless Markup Language (WML)	Coding of information for smaller-screened devices, such as PDAs and cell phones

Q1. The section of the XML document indicated by A is called the:

A. Body
B. Epilog
C. Prolog
D. Spec

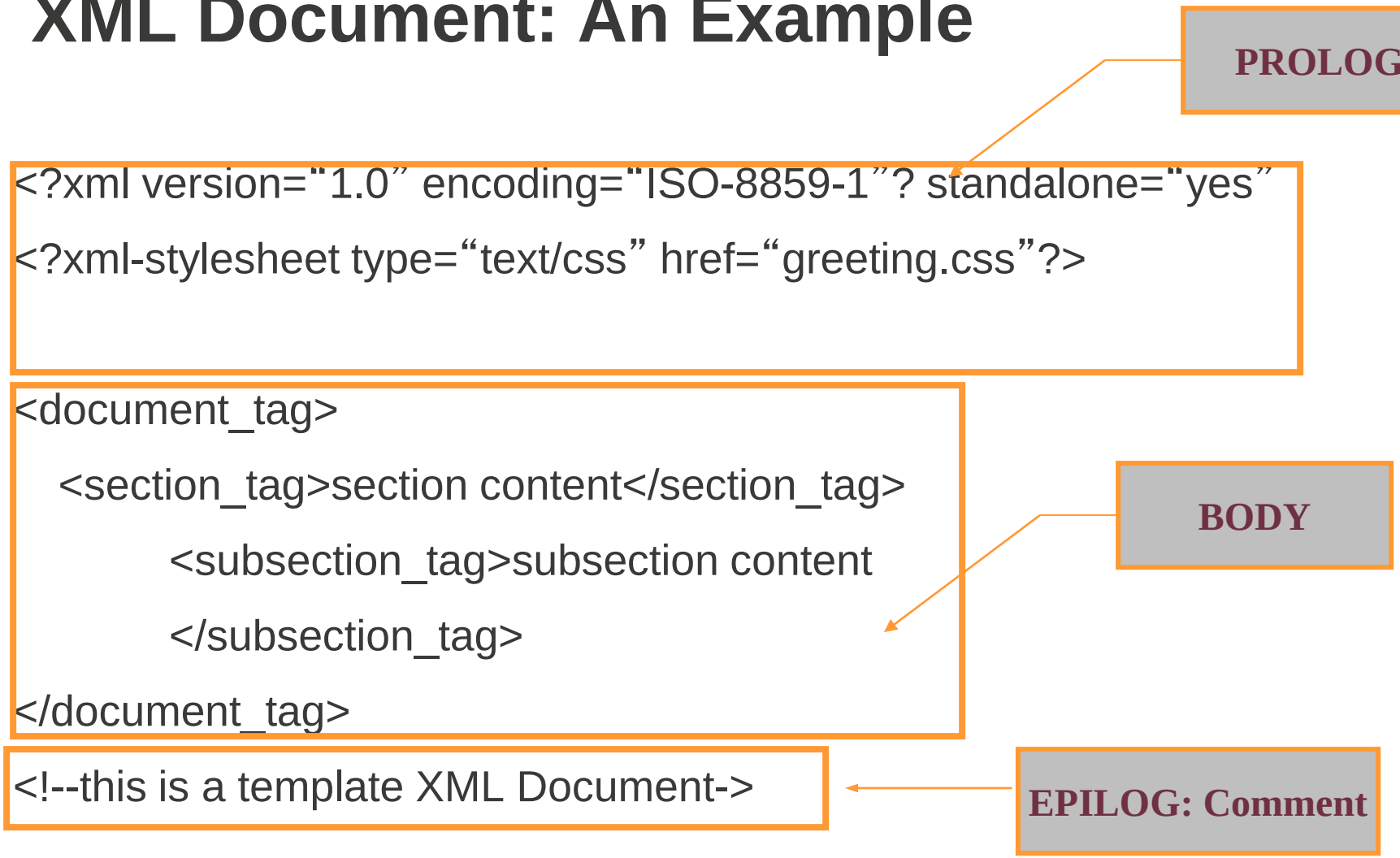
A

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- This document contains data on Jazz Warehouse special offers -->

<items>
   <item>
      <title>Kind of Blue</title>
      <artist>Miles Davis</artist>
      <tracks>
         <track length="9:22">So What</track>
         <track length="9:46">Freddie Freeloader</track>
         <track length="5:37">Blue in Green</track>
         <track length="11:33">All Blues</track>
         <track length="9:26">Flamenco Sketches</track>
      </tracks>
   </item>

   <item>
      <title>Cookin'</title>
      <artist>Miles Davis</artist>
      <tracks>
         <track length="5:57">My Funny Valentine</track>
         <track length="9:53">Blues by Five</track>
         <track length="4:22">Airegin</track>
         <track length="13:03">Tune-Up</track>
      </tracks>
   </item>
</items>
```

1. Structure of an XML Document (1)

- Common structure of XML document:
  - Prolog
    - First line (**required**)
      - <?xml version="1.0" encoding="UTF-8" ?>
    - Comments, DTD, Processing instructions, etc.
  - Body
    - XML root element, nested child XML elements (and attributes)
    - Contents of the document
  - Epilog
    - Rarely used
- This order **must** be followed or the parser will generate an error message

XML Document: An Example



1. Structure of an XML Document (2)

XML Declaration

```
<?xml version="version number" encoding="encoding type"
standalone="yes|no" ?>
```

- Default version is “1.0”, although some systems support “1.1”
- Default encoding is “UTF-8” (English language characters)
- Standalone attribute indicates if the document contains any reference to external files, default is “no”

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
```

Q2. The item in the XML document indicated by B is called the:

A. Body
B. Opening element
C. Prolog
D. Root element

B

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- This document contains data on Jazz Warehouse special offers -->

<items>
   <item>
      <title>Kind of Blue</title>
      <artist>Miles Davis</artist>
      <tracks>
         <track length="9:22">So What</track>
         <track length="9:46">Freddie Freeloader</track>
         <track length="5:37">Blue in Green</track>
         <track length="11:33">All Blues</track>
         <track length="9:26">Flamenco Sketches</track>
      </tracks>
   </item>

   <item>
      <title>Cookin'</title>
      <artist>Miles Davis</artist>
      <tracks>
         <track length="5:57">My Funny Valentine</track>
         <track length="9:53">Blues by Five</track>
         <track length="4:22">Airegin</track>
         <track length="13:03">Tune-Up</track>
      </tracks>
   </item>
</items>
```

2. Working with Elements (1)

XML Elements

```
<element>content</element>
```

- An example

```
<artist>Miles Davis</artist>
```

- How about this?

```
<ARTIST>Miles Davis</artist>
```

Empty Elements

```
<element />
```

2. Working with Elements (2)

Nesting Elements

- An example

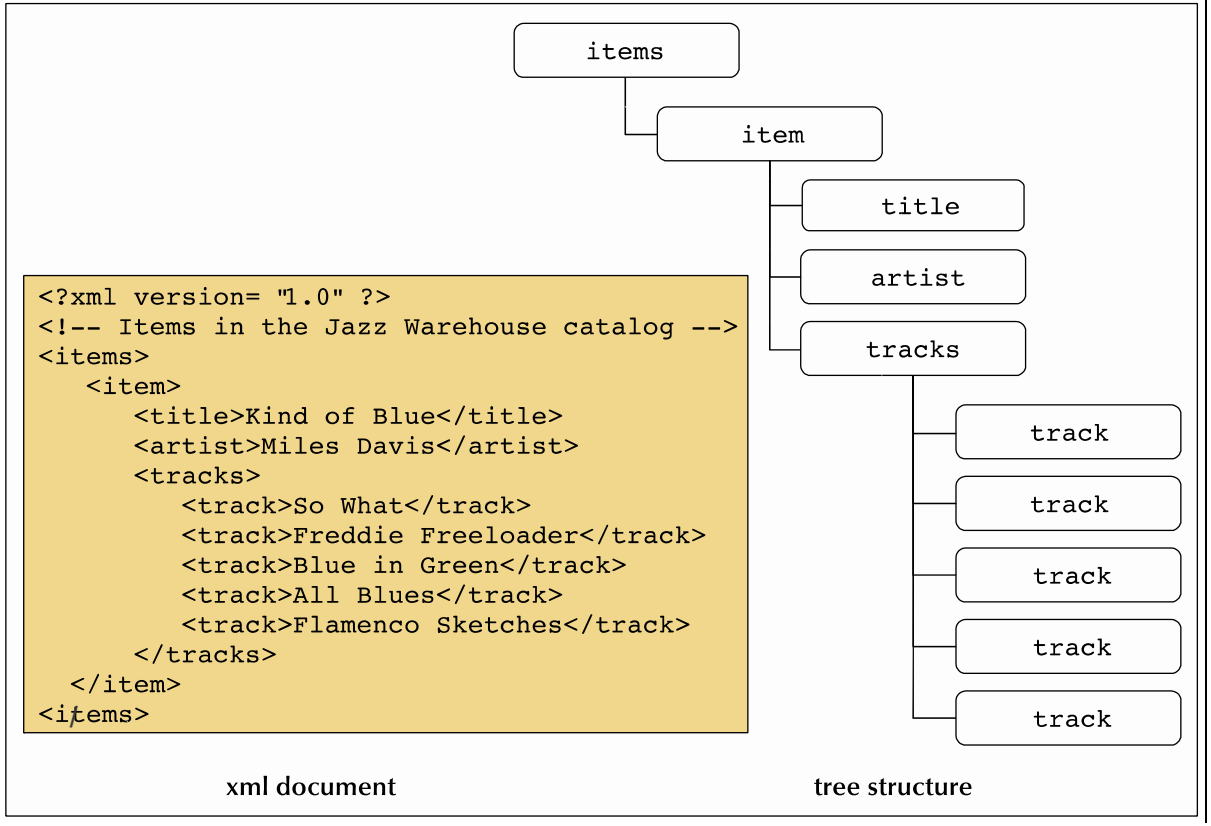
```
<tracks>
   <track>So What (9:22)</track>
   <track>Freddie Freeloader (9:46)</track>
   <track>Blue in Green (5:37)</track>
   <track>All Blues (11:33)</track>
   <track>Flamenco Sketches (9:26)</track>
</tracks>
```

- How about this?

```
<title>Kind of Blue <artist>Miles Davis</title></artist>
```

2. Working with Elements (3)

Code for an XML document along with its corresponding tree diagram — Figure 1-4



2. Working with Elements (4)

Figure 1-5 Charting the number of child elements

Symbol	Description	Chart	Interpretation
[none]	The parent contains a single occurrence of the child element.	item → title	An item can only have one title
?	The child element occurs once or not at all.	title → ? → tracks	A title may or may not have a collection of tracks
*	The child element occurs any number of times.	items → * → item	The items element can contain zero or more item elements
+	The child element occurs at least once.	tracks → + → track	The tracks collection must contain at least one music track

2. Working with Elements (5)

Q3:

```
<?xml version="1.0" ?>
<!-- Items in the Jazz Warehouse catalog -->
<title>Kind of Blue</title>
<artist>Miles Davis</artist>
<tracks>
   <track>So What</track>
   <track>Freddie Freeloader</track>
   <track>Blue in Green</track>
   <track>All Blues</track>
   <track>Flamenco Sketches</track>
</tracks>
```

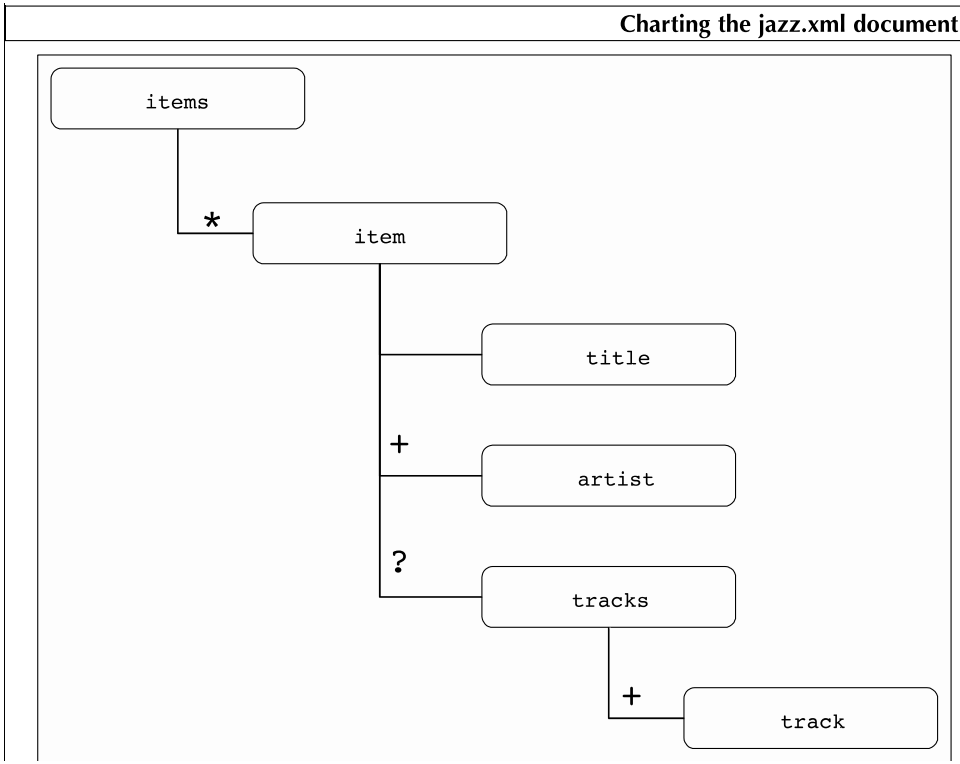
A. Correct

B. Incorrect

C. I don't know

2. Working with Elements (6)

Charting the jazz.xml document — Figure 1-6



* = Zero or many

+ = One or many (at least one)

? = Zero or one

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- This document contains data on Jazz Warehouse special offers -->

<items>
   <item>
      <title>Kind of Blue</title>
      <artist>Miles Davis</artist>
      <tracks>
         <track>So What</track>
         <track>Freddie Freeloader</track>
         <track>Blue in Green</track>
         <track>All Blues</track>
         <track>Flamenco Sketches</track>
      </tracks>
   </item>
   <item>
      <title>Cookin'</title>
      <artist>Miles Davis</artist>
      <tracks>
         <track>My Funny Valentine</track>
         <track>Blues by Five</track>
         <track>Airegin</track>
         <track>Tune-Up</track>
      </tracks>
   </item>
   <item>
      <title>Blue Train</title>
      <artist>John Coltrane</artist>
      <tracks>
         <track>Blue Train</track>
         <track>Moment's Notice</track>
         <track>Locomotion</track>
         <track>I'm Old Fashioned</track>
         <track>Lazy Bird</track>
      </tracks>
   </item>
</items>
```

3. Working with Attributes (1)

Q4:

```
<track length="9:22">So What</track>
```

Is the same as this:

```
<track>
  <track_title>So What</track_title>
  <length>9:22</length>
</track>
```

A. Yes, it's the same

B. No, it's different

C. I don't know

3. Working with Attributes (2)

XML Attributes

```
<element attribute="value"> ... </element>
<element attribute="value" />
```

- An example

```
<track length="9:22">So What</track>
```

- How about this?

```
<track>
  <track_title>So What</track_title>
  <length>9:22</length>
</track>
```

- Attribute vs. Element:
  - Main document text = elements
  - Describe the data (not as data themselves) = attributes

4. Using Character and Entity References (1)

Q5:

```
<artist>Miles Davis & John Coltrane</artist>
```

A. Correct

B. Incorrect

C. I don’t know

4. Using Character and Entity References (2)

- How about these?

```
<artist>Miles Davis & John Coltrane</artist>
<artist>Miles Davis &amp; John Coltrane</artist>
```

Figure 1-12 Character and entity references

Symbol	Character Reference	Entity Reference	Description
©	©		Copyright symbol
®	®		Registered trademark symbol
™	™		Trademark symbol
<	<	<	Less than symbol
>	>	>	Greater than symbol
&	&	&	Ampersand
"	&#22;	"	Double quote
'	&#27;	'	Apostrophe (single quote)
£	£		Pound sign
€	€		Euro sign
¥	¥		Yen sign

4. Using Character and Entity References (3)

Character references

```
<items>

   <item>
      <title>Kind of Blue</title>
      <priceus>US: $11.99</priceus>
      <priceuk>UK: &#163;8.39</priceuk>
      <artist>Miles Davis</artist>
      <tracks>
         <track length="9:22">So What</track>
         <track length="9:46">Freddie Freeloader</track>
         <track length="5:37">Blue in Green</track>
         <track length="11:33">All Blues</track>
         <track length="9:26">Flamenco Sketches</track>
      </tracks>
   </item>

   <item>
      <title>Cookin'</title>
      <priceus>US: $7.99</priceus>
      <priceuk>UK: &#163;5.59</priceuk>
      <artist>Miles Davis</artist>
      <tracks>
         <track length="5:57">My Funny Valentine</track>
         <track length="9:53">Blues by Five</track>
         <track length="4:22">Airegin</track>
         <track length="13:03">Tune-Up</track>
      </tracks>
   </item>

</items>
```

5. Understanding Text Characters & White Spaces (1)

Q6:

```
<htmlcode>
  <![CDATA[
    <h1>The Jazz Warehouse</h1>
    <h2>Your Online Store for Jazz Music</h2>
]]></htmlcode>
```

A. This is an example of a Parsed Character Data

B. This is an example of an Unparsed Character Data

C. I don't know

5. Understanding Text Characters & White Spaces (2)

Parsed Character Data and **Character Data**

- An example

```
<temperature> &gt;100 degrees </temperature>
```

- The character data becomes…?

White Space

```
<paragraph>This is      a
                    paragraph</paragraph>
```

5. Understanding Text Characters & White Spaces (3)

Creating a CDATA Section

```
<![CDATA[
    character data
]]>
```

- An example:

```
<htmlcode>
  <![CDATA[
    <h1>The Jazz Warehouse</h1>
    <h2>Your Online Store for Jazz Music</h2>
]]></htmlcode>
```

- may be placed anywhere within in a document
- cannot be nested within other CDATA sections
- cannot be empty

5. Understanding Text Characters & White Spaces (4)

CDATA section →

```
<items>
   <message>
   <![CDATA[
      Here are some of the latest specials from the Jazz Warehouse.
      Please note that all Miles Davis & John Coltrane CDs will be
      on sale for the month of March.
   ]]>
   </message>

   <item>
      <title>Kind of Blue</title>
      <priceus>US: $11.99</priceus>
      <priceuk>UK: &#163;8.39</priceuk>
      <artist>Miles Davis</artist>
      <tracks>
         <track length="9:22">So What</track>
         <track length="9:46">Freddie Freeloader</track>
         <track length="5:37">Blue in Green</track>
         <track length="11:33">All Blues</track>
         <track length="9:26">Flamenco Sketches</track>
      </tracks>
   </item>

   <item>
      <title>Cookin'</title>
```

6. Processing an XML Document (1)

Figure 1-15 Parsing an XML document

6. Processing an XML Document (2)

Q7:

A document that contains no syntax errors and satisfies the general specifications for XML code, is said to be:

A. synthesized
B. checked
C. well-formed
D. valid

6. Processing an XML Document (3)

- An XML processor (also called **XML parser**) evaluates an XML document to make sure it conforms to all XML specifications for structure and syntax
- There are two categories of XML documents
  - Well-formed
  - Valid
- An XML document is well-formed if it contains no syntax errors and fulfills all of the specifications for XML code as defined by the W3C
- An XML document is valid if it is well-formed and also satisfies the rules laid out in the DTD (Document Type Definition) or schema attached to the document
- Validation Parser
  - Uses a DTD/Schema to validate an XML document
- Non validating Parser
  - Only enforces well formedness

6. Processing an XML Document (4)

- **Well formed**
  - If document conforms to the w3c syntax rules:
    - A single root element exists
    - All elements have a closing tag
    - Case match occurs in tags
    - Attribute values are quoted and appear only once in the same tag
    - Proper nesting has occurred
    - other rules also exist
  - *It is said to be well formed*
- **Valid**
  - Is well formed **but also**
  - Must have a Document Type Declaration (DTD) or schema attached and must obey the constraints of that declaration
    - e.g. Value and range bounding, data typing

7. Formatting XML Data (1)

Cascading Style Sheets (CSS)

- Applying a Style to an Element

```
selector {attribute1:value1; attribute2:value2; ...}
```

- An example:

```
artist {color:red; font-weight: bold}
```

7. Formatting XML Data (2)

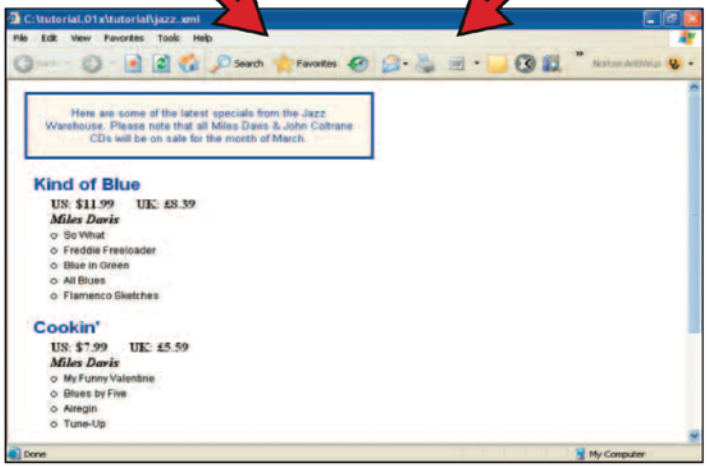
Combining an XML document and a style sheet

XML document

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- This document contains data on Jazz Warehouse special offers -->
<?xml-stylesheet type="text/css" href="jw.css" ?>

<items>
   <message>
   <![CDATA[
      Here are some of the latest specials from the Jazz Warehouse.
      Please note that all Miles Davis & John Coltrane CDs will be
      on sale for the month of March.
   ]]>
   </message>

   <item>
      <title>Kind of Blue</title>
      <priceus>US: $11.99</priceus>
      <priceuk>UK: &#163;8.39</priceuk>
      <artist>Miles Davis</artist>
      <tracks>
         <track length="9:22">So What</track>
         <track length="9:46">Freddie Freeloader</track>
         <track length="5:37">Blue in Green</track>
```

style sheet

```
message          {display:block; width:400px; color:blue; text-align: center;
                  font-size: 10pt; font-family: Arial, Helvetica, sans-serif;
                  border: 3px solid blue; background-color: ivory;
                  margin: 10px; padding: 15px}

item             {display:block; font-size:14pt; color:red;
                  font-family: Arial, Helvetica, Sans-serif;
                  margin: 20px}

title            {display:block; font-size: 16pt; color:blue;
                  font-weight:bold;
                  font-family: Arial, Helvetica, sans-serif}

priceus, priceuk {color:black; font-size: 12pt; font-weight: bold;
                  font-family: Times New Roman, Times, Serif;
                  margin-left: 20px;}

artist           {display:block; font-size: 12pt; color:black;
                  font-style:italic; font-weight: bold;
                  font-family: Times New Roman, Times, Serif;
                  margin-left: 20px}

track            {display:list-item; font-size: 9pt; color: black;
                  list-style-type: circle;
                  font-family: Arial, Helvetica, sans-serif;
                  margin-left: 35px}
```



rendered Web page

8. Inserting a Processing Instruction (1)

General Format

```
<?target instruction ?>
```

Cascading Style Sheet

```
<?xml-stylesheet type=”text/css" href="url" ?>
```

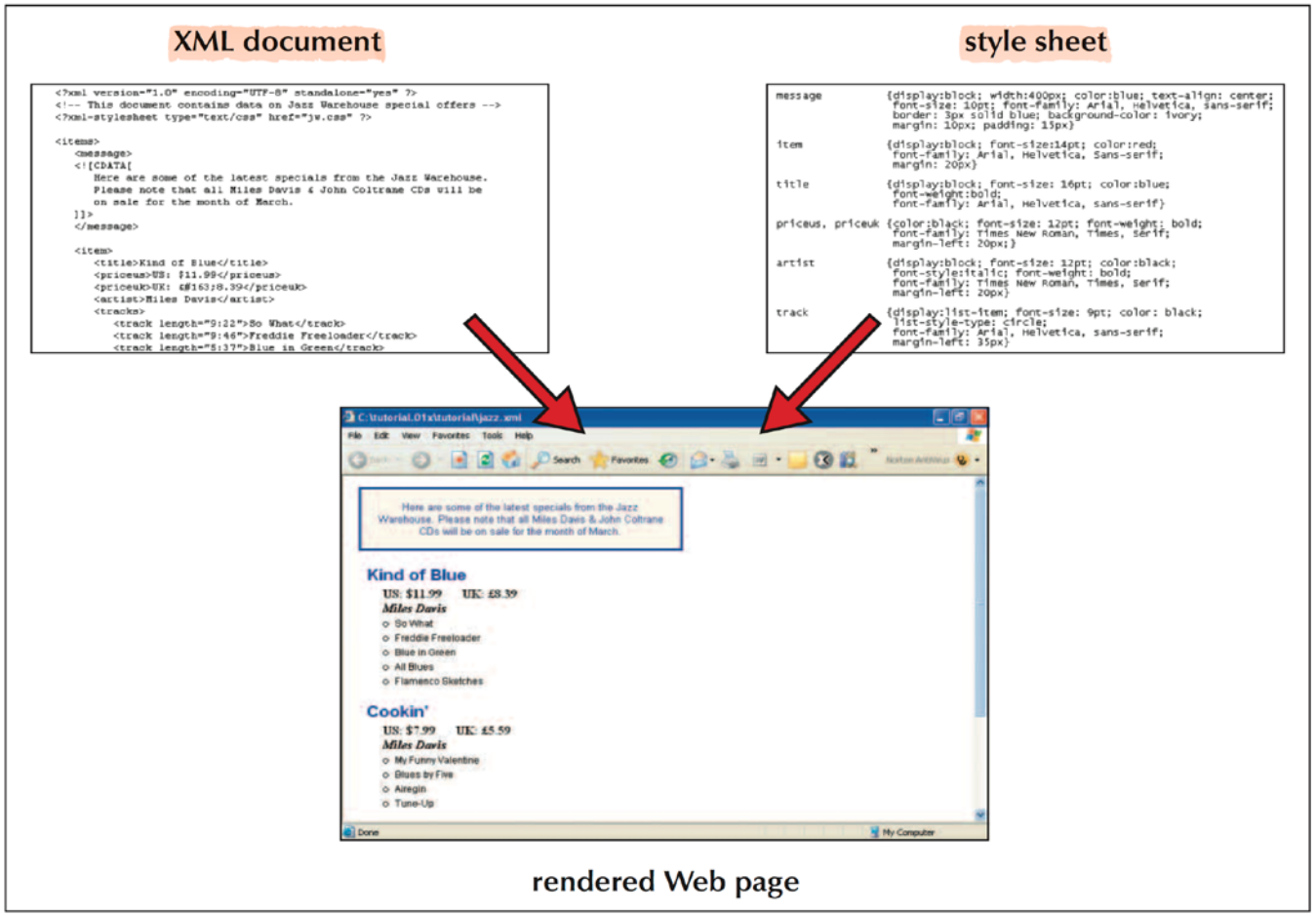
- An example:

```
<?xml-stylesheet type="text/css" href="jw.css" ?>
```

- URL: the name and location of the style sheet

8. Inserting a Processing Instruction (2)

Combining an XML document and a style sheet

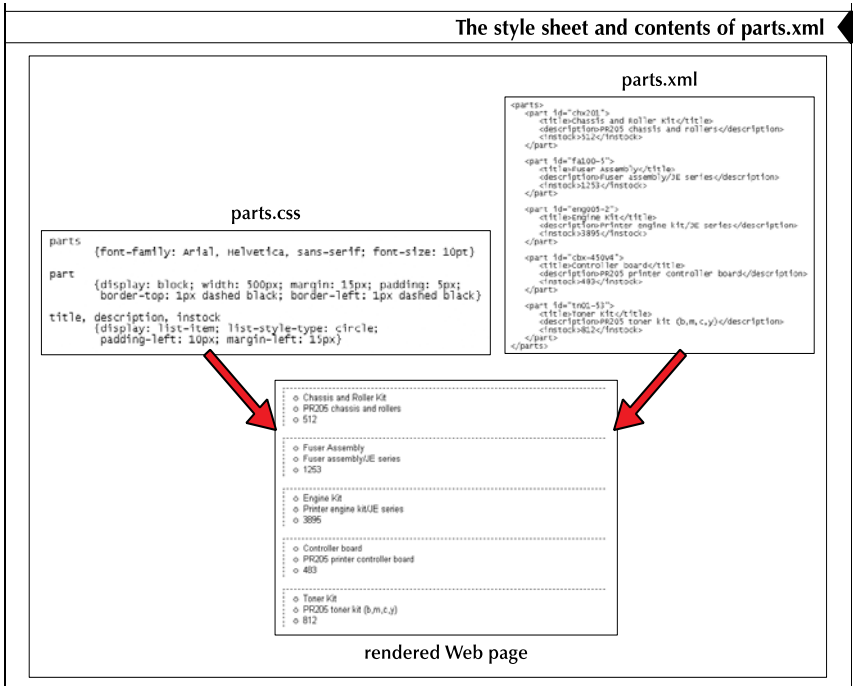


rendered Web page

9. Combining XML Vocabularies (1)

1. Parts vocabulary

The style sheet and contents of parts.xml — Figure 2-2



9. Combining XML Vocabularies (2)

2. Model vocabulary

Figure 2-3 Tree structure of the model vocabulary

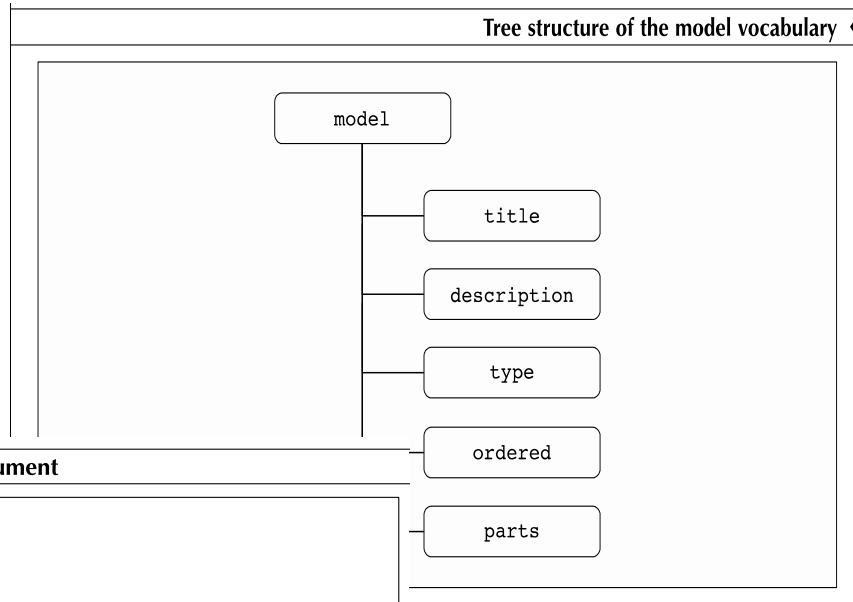
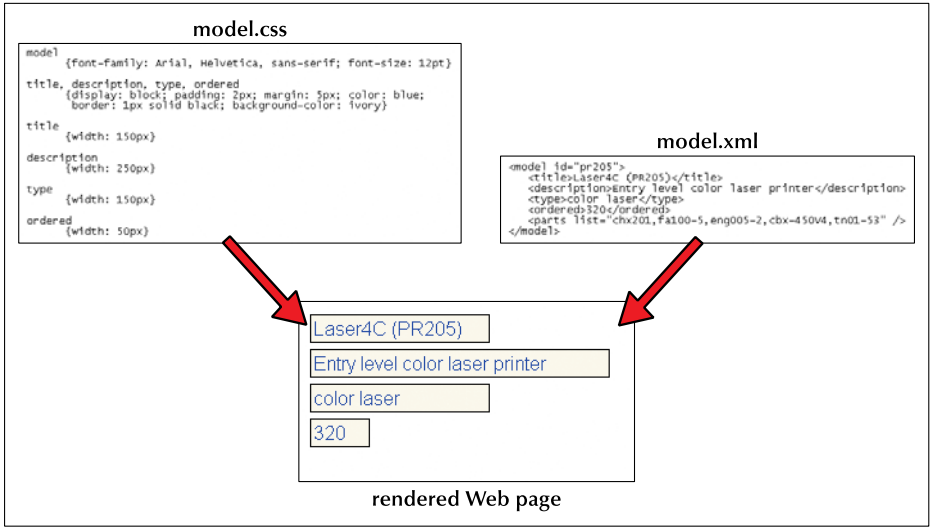


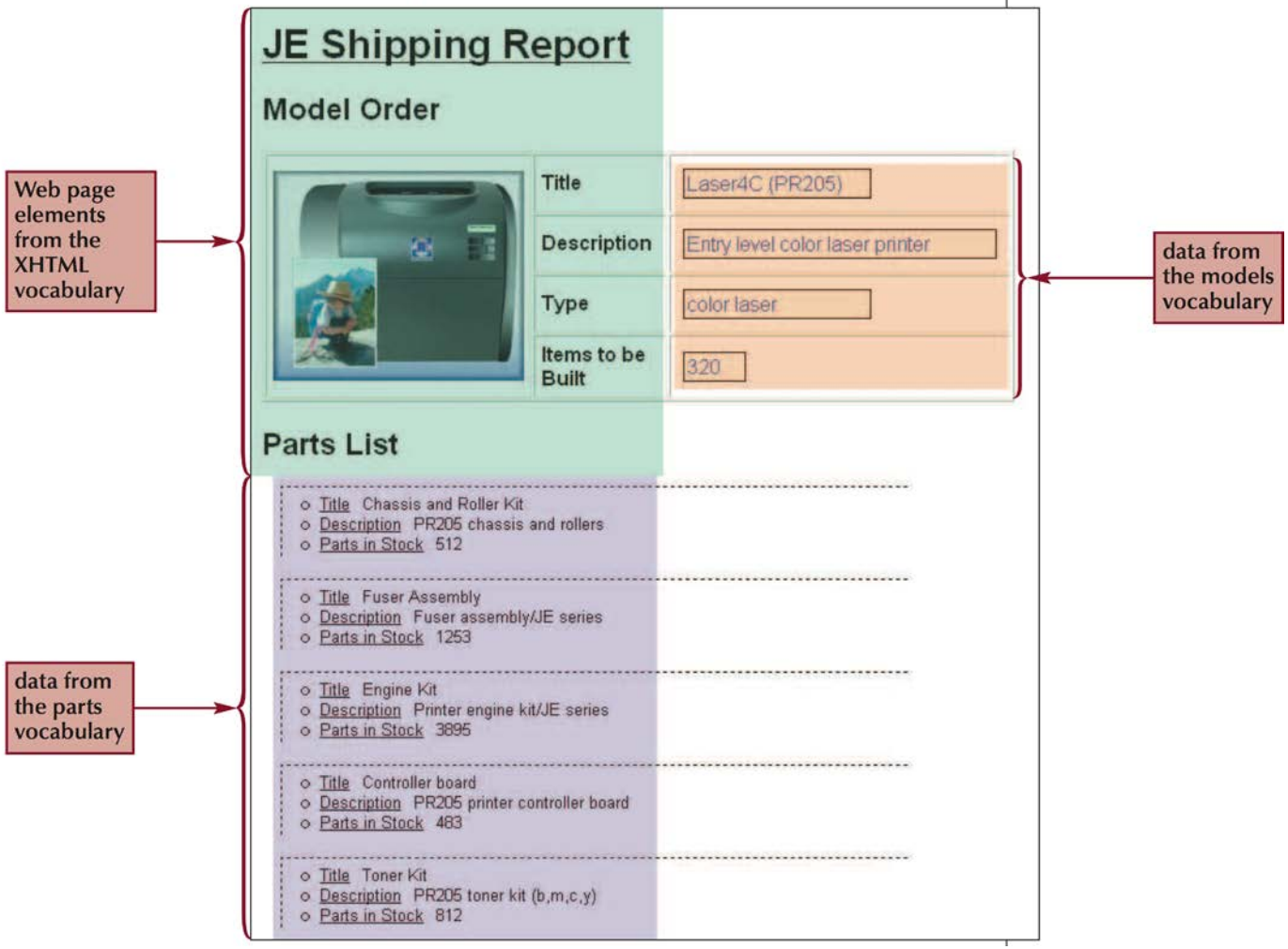
Figure 2-4 The style sheet and content of the model.xml document



9. Combining XML Vocabularies (3)

A desired combined document (order.xml)

Proposed report page combining multiple vocabularies — Figure 2-5



9. Combining XML Vocabularies (4)

Figure 2-8 The compound order.xml document

```
<model id="pr205">
   <title>Laser4C (PR205)</title>
   <description>Entry level color laser printer</description>
   <type>color laser</type>
   <ordered>320</ordered>
   <parts list="chx201,fa100-5,eng005-2,cbx-450V4,tn01-53" />

   <parts>
      <part id="chx201">
         <title>Chassis and Roller Kit</title>
         <description>PR205 chassis and rollers</description>
         <instock>512</instock>
      </part>

      <part id="fa100-5">
         <title>Fuser Assembly</title>
         <description>Fuser assembly/JE series</description>
         <instock>1253</instock>
      </part>

      <part id="eng005-2">
         <title>Engine Kit</title>
         <description>Printer engine kit/JE series</description>
         <instock>3895</instock>
      </part>

      <part id="cbx-450V4">
         <title>Controller board</title>
         <description>PR205 printer controller board</description>
         <instock>483</instock>
      </part>

      <part id="tn01-53">
         <title>Toner Kit</title>
         <description>PR205 toner kit (b,m,c,y)</description>
         <instock>812</instock>
      </part>
   </parts>

</model>
```

- Copy the two original xml files (parts.xml and model.xml) into one xml document (called **order.xml**), where <model> is the root element of the new xml document

Figure 2-6 Linking to the parts.css and model.css style sheets

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!--
   New Perspectives on XML
   Tutorial 2
   Tutorial Case

   Jackson Electronics Order Report
   Author: Gail Oglund
   Date:   3/1/2008

   Filename:         order.xml
   Supporting Files: model.css, parts.css
-->

<?xml-stylesheet type="text/css" href="parts.css" ?>
<?xml-stylesheet type="text/css" href="model.css" ?>
```

- Insert the two stylesheet processing instructions into order.xml

9. Combining XML Vocabularies (5)

Q8:

- Copy the two original xml files (parts.xml and model.xml) into one xml document (called **order.xml**), where <model> is the root element of the new xml document
- Insert the two stylesheet processing instructions into order.xml
- Any problems?

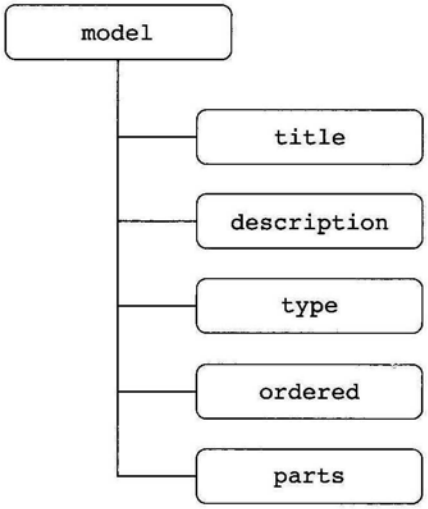
A. There will be **errors** in order.xml because it is not possible to have two stylesheet processing instructions

B. There will be **errors** in order.xml because the <model> element cannot be the root of the new combined xml document

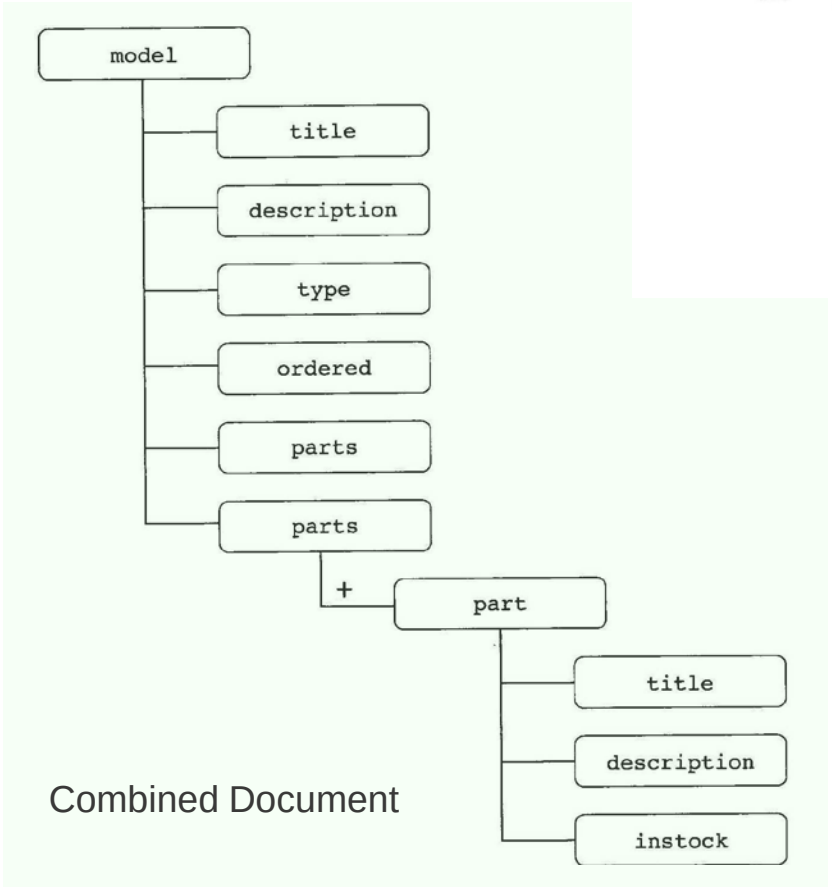
C. There is **no error**, but the rendered webpage is not as expected

D. I don't know

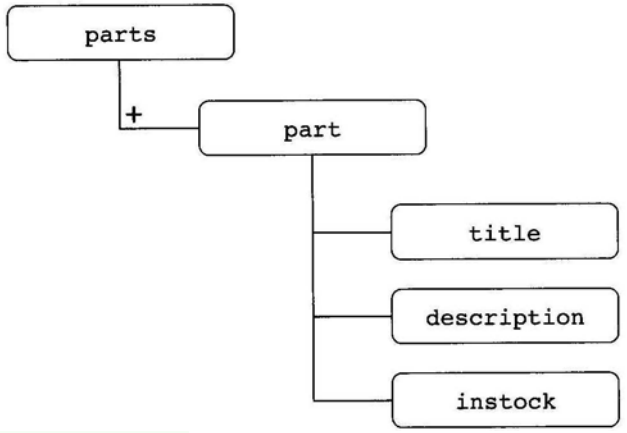
9. Combining XML Vocabularies (6)



Document 1

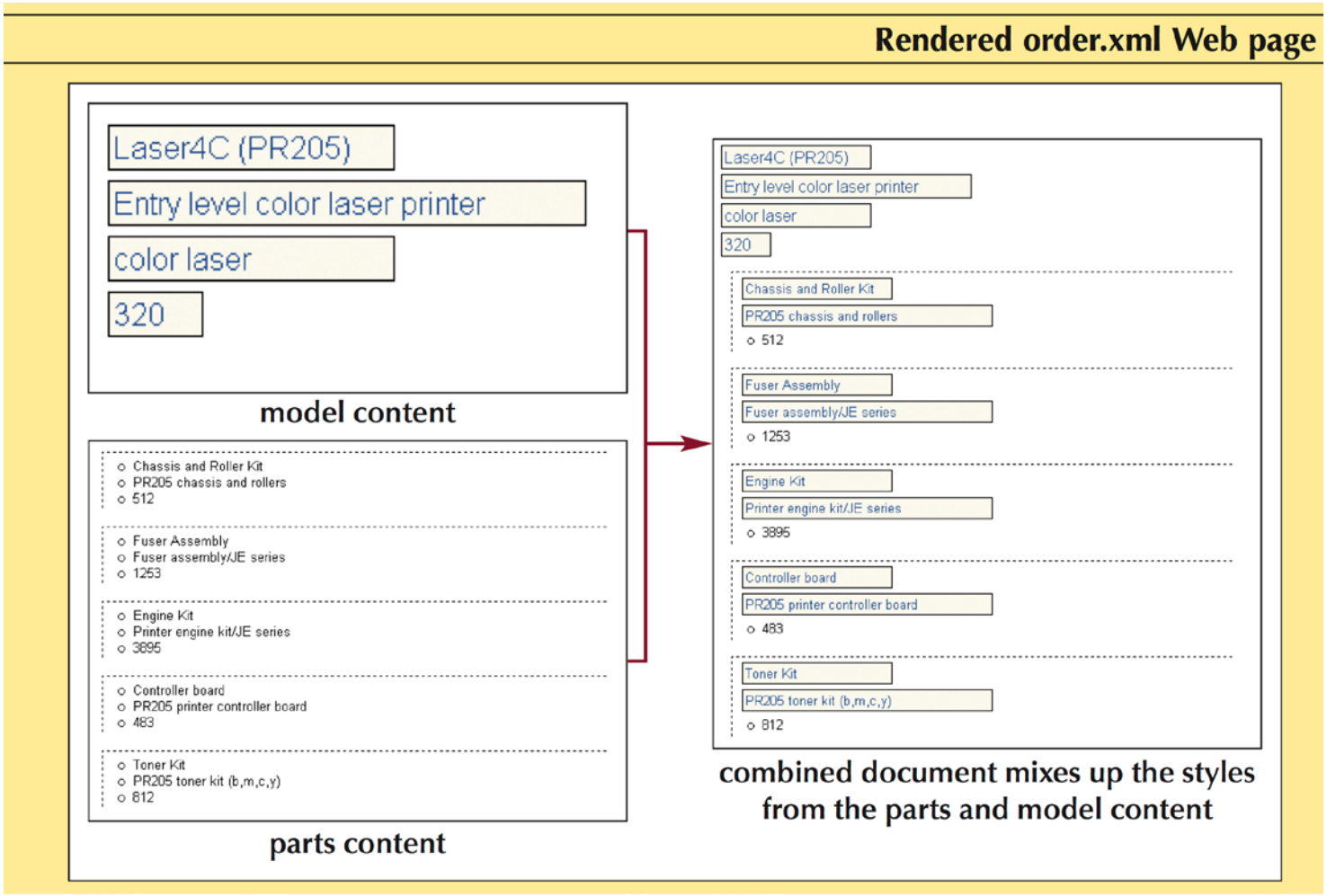


Combined Document



Document 2

9. Combining XML Vocabularies (7)



10. Working with Namespaces (1)

Declaring a Namespace

- Add the following attribute to an element in the xml document:

```
xmlns:prefix="uri”
```

- An example:

```
<model xmlns:mod="http://jacksonelect.com/models">
```

- The namespace is applied to any descendent of the element
- The URI (Uniform Resource Identifier) is not necessarily a Web address. A URI identifies a physical *or an abstract resource*. Using a web address as the stem:
  - helps ensure unique namespaces, and
  - provides a location for namespace documentation

10. Working with Namespaces (2)

Applying a Namespace to an Element

- Once it has been declared and its URI specified, the namespace is applied to elements and attributes by inserting the namespace prefix before each element name that belongs to the namespace.

```
<prefix:element>
    content
</prefix:element>
```

- Here, prefix is the namespace prefix and element is the local part of the element name.
- Prefixed names are called *qualified names* and an element name without a namespace prefix is called an *unqualified name*.
  - qualified: <pa:title>
  - unqualified: <title>

10. Working with Namespaces (3)

Figure 2-13 Applying the parts namespace prefix

```
<mod:model id="pr205" xmlns:mod="http://jacksonelect.com/models"
                      xmlns:pa="http://jacksonelect.com/parts">
   <mod:title>Laser4C (PR205)</mod:title>
   <mod:description>Entry level color laser printer</mod:description>
   <mod:type>color laser</mod:type>
   <mod:ordered>320</mod:ordered>
   <mod:parts list="chx201,fa100-5,eng005-2,cbx-450V4,tn01-53" />

   <pa:parts>
      <pa:part id="chx201">
         <pa:title>Chassis and Roller Kit</pa:title>
         <pa:description>PR205 chassis and rollers</pa:description>
         <pa:instock>512</pa:instock>
      </pa:part>

      <pa:part id="fa100-5">
         <pa:title>Fuser Assembly</pa:title>
         <pa:description>Fuser assembly/JE series</pa:description>
         <pa:instock>1253</pa:instock>
      </pa:part>

      <pa:part id="eng005-2">
         <pa:title>Engine Kit</pa:title>
         <pa:description>Printer engine kit/JE series</pa:description>
         <pa:instock>3895</pa:instock>
      </pa:part>

      <pa:part id="cbx-450V4">
         <pa:title>Controller board</pa:title>
         <pa:description>PR205 printer controller board</pa:description>
         <pa:instock>483</pa:instock>
      </pa:part>

      <pa:part id="tn01-53">
         <pa:title>Toner Kit</pa:title>
         <pa:description>PR205 toner kit (b,m,c,y)</pa:description>
         <pa:instock>812</pa:instock>
      </pa:part>
   </pa:parts>

</mod:model>
```

10. Working with Namespaces (4)

Applying a Default Namespace

- A default namespace can be declared by omitting the prefix in the namespace declaration
- Any descendant element or attribute is considered part of this namespace, unless a different namespace is declared
- For example, to define a namespace as the default namespace:

```
<model xmlns="http://jacksonelect.com/models">
   <title>Laser4C (PR205)</title>
   <description>Entry level color laser printer</description>
   <type>color laser</type>
   <ordered>320</ordered>
   <parts list="chx201,fa100-5,eng005-2,cbx-450V4,tn01-53" />
</model>
```

- Advantage is that coding is easier, no need to add prefix

11. Working with Attributes (1)

Q9:

```
1. <mod:model id="pr205" xmlns:mod="http://jacksonelect.com/models">
2.    <mod:title>Laser4C (PR205)</mod:title>
3.    <mod:description>Entry .. color laser printer</mod:description>
4.    <mod:type>color laser</mod:type>
5.    <mod:ordered>320</mod:ordered>
6.    <mod:parts list="chx201,fa100-5,eng005-2,cbx-450V4,tn01-53" />
7. </mod:model>
```

A. Line 1 is correct, line 6 is incorrect

B. Line 1 is incorrect, line 6 is correct

C. Lines 1 and 6 are both correct

D. I don’t know

11. Working with Attributes (2)

- Declaring Attributes, like elements, *can* become qualified by adding the namespace prefix to the attribute name. For example,
  - *prefix:attribute="value"*
- *However*, an attribute name without a prefix is assumed to belong to the same namespace as the element that contains it

```
<mod:model id="pr205" xmlns:mod="http://jacksonelect.com/models">
   <mod:title>Laser4C (PR205)</mod:title>
   <mod:description>Entry .. color laser printer</mod:description>
   <mod:type>color laser</mod:type>
   <mod:ordered>320</mod:ordered>
   <mod:parts list="chx201,fa100-5,eng005-2,cbx-450V4,tn01-53" />
</mod:model>
```

12. Adding a Namespace to a Style Sheet (1)

Q10: model.css

```
@namespace mod url(httl://jacksonelect.com/models);
mod:model
      {font-family: Arial, Helvetica, sans-serif; font-size: 12pt}
mod:title, mod:description, mod:type, mod:ordered
      {display: block; padding: 2px; margin: 5px; color: blue;
       border: 1px solid black; background-color: ivory}
mod:title
      {width: 150px}
mod:description
      {width: 250px}
mod:type
      {width: 150px}
mod:ordered
      {width: 50px}
```

A. A correct stylesheet

B. An incorrect stylesheet

C. I don't know if this is correct or not

12. Adding a Namespace to a Style Sheet (2)

Declaring a Namespace

Add the following rule to the style sheet file:

```
@namespace prefix url(uri);
```

An example:

```
@namespace mod url(http://jacksonelect.com/models);
```

Applying a Namespace to a Selector

Add the following rule to the style sheet file:

```
prefix|selector {attribute1:value1; attribute2:value2; ...}
```

Examples:

```
mod|title {width: 150px}
mod|* {font-size: 12pt}
*|title {width: 150px}
title {width: 150px}
```

Applying the models namespace to the style sheet

```
@namespace mod url(http://jacksonelect.com/models);

mod|model
      {font-family: Arial, Helvetica, sans-serif; font-size: 12pt}

mod|title, mod|description, mod|type, mod|ordered
      {display: block; padding: 2px; margin: 5px; color: blue;
       border: 1px solid black; background-color: ivory}

mod|title
      {width: 150px}

mod|description
      {width: 250px}

mod|type
      {width: 150px}

mod|ordered
      {width: 50px}
```

12. Adding a Namespace to a Style Sheet (3)

Defining Namespaces with the Escape Character

Internet Explorer browser:

```
prefix\:selector {attribute1:value1; attribute2:value2; ...}
```

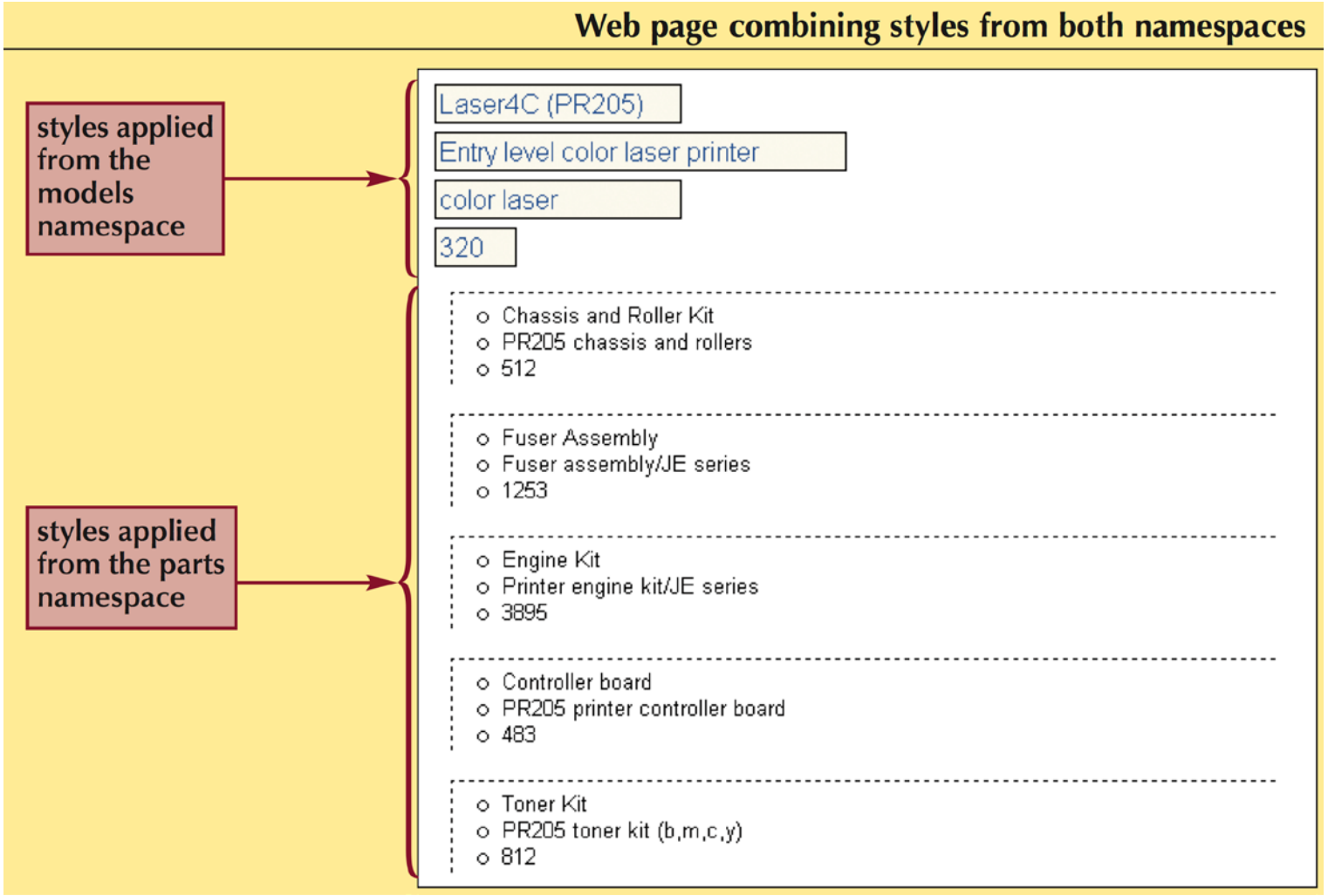
Examples:

```
mod\:title {width: 150px}
mod\:* {font-size: 12pt}
```

Using the escape character in the model.css style sheet

```
/* Insert CSS namespace escape styles here */

mod\:model
      {font-family: Arial, Helvetica, sans-serif; font-size: 12pt}

mod\:title, mod\:description, mod\:type, mod\:ordered
      {display: block; padding: 2px; margin: 5px; color: blue;
       border: 1px solid black; background-color: ivory}

mod\:title
      {width: 150px}

mod\:description
      {width: 250px}

mod\:type
      {width: 150px}

mod\:ordered
      {width: 50px}
```

12. Adding a Namespace to a Style Sheet (4)



12. Adding a Namespace to a Style Sheet (5)

How about if we want to have this instead?

JE Shipping Report

Model Order

Title	Laser4C (PR205)
Description	Entry level color laser printer
Type	color laser
Items to be Built	320

Parts List

- Title Chassis and Roller Kit
- Description PR205 chassis and rollers
- Parts in Stock 512

- Title Fuser Assembly
- Description Fuser assembly/JE series
- Parts in Stock 1253

- Title Engine Kit
- Description Printer engine kit/JE series
- Parts in Stock 3895

- Title Controller board
- Description PR205 printer controller board
- Parts in Stock 483

- Title Toner Kit
- Description PR205 toner kit (b,m,c,y)
- Parts in Stock 812

13. Combining Standard Vocabularies (1)

- How to combine custom XML vocabularies with any other standard vocabularies, such as XHTML, RSS, MathML, etc.
- All of these standard vocabularies have unique URIs.

Namespace URIs for standard vocabularies

Vocabulary	Namespace URI
CDF	http://www.microsoft.com/standards/channels.dtd
CML	http://www.xml-cml.org/dtd/cml1_0_1.dtd
MathML	http://www.w3.org/1998/Math/MathML
RSS 1.0	http://purl.org/rss/1.0/
SMIL	http://www.w3.org/2001/SMIL20/Language
SVG	http://www.w3.org/2000/svg
VoiceXML	http://www.w3.org/2001/vxml
XForms	http://www.w3.org/2002/xforms
XHTML	http://www.w3.org/1999/xhtml
XMLSchema	http://www.w3.org/2001/XMLSchema
XSLT	http://www.w3.org/1999/XSL/Transform

13. Combining Standard Vocabularies (2)

Combining XML and XHTML

- **Step 1**: Convert the HTML file to an XHTML file
  - Add an XML declaration at the top of the file; and
  - Set the default namespace of the document to the XHTML vocabulary
  - Assume report.css already exists

report.htm — **Changing an HTML document to XHTML**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!--
   New Perspectives on XML
   Tutorial 2
   Tutorial Case

   Jackson Electronics Shipping Report
   Author:  Gail Oglund
   Date:    3/1/2008

   Filename:         report.xml
   Supporting Files: model.css, parts.css, pr205.jpg, report.css
-->

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>Jackson Electronics Shipping Order</title>
   <link rel="stylesheet" href="report.css" type="text/css" />
</head>
```

13. Combining Standard Vocabularies (3)

Combining XML and XHTML

- **Step 2**: Copy the parts (from order.xml) to this report.htm

  - Add the **xmlns:pa** attribute to the opening html tag; and

  - Add a link element to link to **parts.css**.

report.htm

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:pa="http://jacksonelect.com/parts">
<head>
   <title>Jackson Electronics Shipping Order</title>
   <link rel="stylesheet" href="report.css" type="text/css" />
   <link rel="stylesheet" href="parts.css" type="text/css" />
</head>

<body>
   <h1>JE Shipping Report</h1>
   <h2>Model Order</h2>

   <table border="1" cellpadding="5">
   <tr>
      <td rowspan="4"><img src="pr205.jpg" alt="" /></td>
      <th>Title</th>
      <td></td>
   </tr>
   <tr>
      <th>Description</th>
      <td></td>
   </tr>
   <tr>
      <th>Type</th>
      <td></td>
   </tr>
   <tr>
      <th>Items to be Built</th>
      <td></td>
   </tr>
   </table>

   <h2>Parts List</h2>

   <pa:parts>
      <pa:part id="chx201">
         <pa:title>Chassis and Roller Kit</pa:title>
         <pa:description>PR205 chassis and rollers</pa:description>
         <pa:instock>512</pa:instock>
      </pa:part>

      <pa:part id="fa100-5">
         <pa:title>Fuser Assembly</pa:title>
         <pa:description>Fuser assembly/JE series</pa:description>
         <pa:instock>1253</pa:instock>
      </pa:part>

      <pa:part id="eng005-2">
         <pa:title>Engine Kit</pa:title>
         <pa:description>Printer engine kit/JE series</pa:description>
         <pa:instock>3895</pa:instock>
      </pa:part>

      <pa:part id="cbx-450V4">
         <pa:title>Controller board</pa:title>
         <pa:description>PR205 printer controller board</pa:description>
         <pa:instock>483</pa:instock>
      </pa:part>

      <pa:part id="tn01-53">
         <pa:title>Toner Kit</pa:title>
         <pa:description>PR205 toner kit (b,m,c,y)</pa:description>
         <pa:instock>812</pa:instock>
      </pa:part>
   </pa:parts>

</body>
</html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:pa="http://jacksonelect.com/parts">
<head>
   <title>Jackson Electronics Shipping Order</title>
   <link rel="stylesheet" href="report.css" type="text/css" />
   <link rel="stylesheet" href="parts.css" type="text/css" />
</head>

<body>
   <h1>JE Shipping Report</h1>
   <h2>Model Order</h2>

   <table border="1" cellpadding="5">
   <tr>
      <td rowspan="4"><img src="pr205.jpg" alt="" /></td>
      <th>Title</th>
      <td></td>
   </tr>
   <tr>
      <th>Description</th>
      <td></td>
   </tr>
   <tr>
      <th>Type</th>
      <td></td>
   </tr>
   <tr>
      <th>Items to be Built</th>
      <td></td>
   </tr>
   </table>

   <h2>Parts List</h2>

   <pa:parts>
      <pa:part id="chx201">
         <pa:title>Chassis and Roller Kit</pa:title>
         <pa:description>PR205 chassis and rollers</pa:description>
         <pa:instock>512</pa:instock>
      </pa:part>

      <pa:part id="fa100-5">
         <pa:title>Fuser Assembly</pa:title>
         <pa:description>Fuser assembly/JE series</pa:description>
         <pa:instock>1253</pa:instock>
      </pa:part>

      <pa:part id="eng005-2">
         <pa:title>Engine Kit</pa:title>
         <pa:description>Printer engine kit/JE series</pa:description>
         <pa:instock>3895</pa:instock>
      </pa:part>

      <pa:part id="cbx-450v4">
         <pa:title>Controller board</pa:title>
         <pa:description>PR205 printer controller board</pa:description>
         <pa:instock>483</pa:instock>
      </pa:part>

      <pa:part id="tn01-53">
         <pa:title>Toner Kit</pa:title>
         <pa:description>PR205 toner kit (b,m,c,y)</pa:description>
         <pa:instock>812</pa:instock>
      </pa:part>
   </pa:parts>

</body>
</html>
```

JE Shipping Report

Model Order

Title	
Description	
Type	
Items to be Built	

Parts List

- Chassis and Roller Kit
- PR205 chassis and rollers
- 512

- Fuser Assembly
- Fuser assembly/JE series
- 1253

- Engine Kit
- Printer engine kit/JE series
- 3895

- Controller board
- PR205 printer controller board
- 483

- Toner Kit
- PR205 toner kit (b,m,c,y)
- 812

13. Combining Standard Vocabularies (4)

Combining XML and XHTML

- **Step 3**: Inserting descriptive XHTML element

  - Modify the part section in report.tml, by adding a span element in each of the parts elements (span element is a HTML tag)

Inserting descriptive XHTML elements

```
<pa:parts>
   <pa:part id="chx201">
      <pa:title><span>Title</span>Chassis and Roller Kit</pa:title>
      <pa:description><span>Description</span>PR205 chassis and rollers</pa:description>
      <pa:instock><span>Parts in Stock</span>512</pa:instock>
   </pa:part>

   <pa:part id="fa100-5">
      <pa:title><span>Title</span>Fuser Assembly</pa:title>
      <pa:description><span>Description</span>Fuser assembly/JE series</pa:description>
      <pa:instock><span>Parts in Stock</span>1253</pa:instock>
   </pa:part>

   <pa:part id="eng005-2">
      <pa:title><span>Title</span>Engine Kit</pa:title>
      <pa:description><span>Description</span>Printer engine kit/JE series</pa:description>
      <pa:instock><span>Parts in Stock</span>3895</pa:instock>
   </pa:part>

   <pa:part id="cbx-450V4">
      <pa:title><span>Title</span>Controller board</pa:title>
      <pa:description><span>Description</span>PR205 printer controller board</pa:description>
      <pa:instock><span>Parts in Stock</span>483</pa:instock>
   </pa:part>

   <pa:part id="tn01-53">
      <pa:title><span>Title</span>Toner Kit</pa:title>
      <pa:description><span>Description</span>PR205 toner kit (b,m,c,y)</pa:description>
      <pa:instock><span>Parts in Stock</span>812</pa:instock>
   </pa:part>
</pa:parts>
```

Parts List

- Title Chassis and Roller Kit
  - Description PR205 chassis and rollers
  - Parts in Stock 512
- Title Fuser Assembly
  - Description Fuser assembly/JE series
  - Parts in Stock 1253
- Title Engine Kit
  - Description Printer engine kit/JE series
  - Parts in Stock 3895
- Title Controller board
  - Description PR205 printer controller board
  - Parts in Stock 483
- Title Toner Kit
  - Description PR205 toner kit (b,m,c,y)
  - Parts in Stock 812

13. Combining Standard Vocabularies (5)

Combining XML and XHTML

- **Step 4**: Modify the model section in report.htm
  - Add the **xmlns:pa** attribute to the opening html tag;
  - Add a link element to link to **model.css**.
  - Add the model elements to the table next to the picture in report.htm

report.htm — **Adding elements from the model vocabulary**

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:pa="http://jacksonelect.com/parts"
      xmlns:mod="http://jacksonelect.com/models">
<head>
   <title>Jackson Electronics Shipping Order</title>
   <link rel="stylesheet" href="report.css" type="text/css" />
   <link rel="stylesheet" href="parts.css" type="text/css" />
   <link rel="stylesheet" href="model.css" type="text/css" />
</head>

<body>
   <h1>JE Shipping Report</h1>
   <h2>Model Order</h2>

   <table border="1" cellpadding="5">
   <tr>
      <td rowspan="4"><img src="pr205.jpg" alt="" /></td>
      <th>Title</th>
      <td><mod:title>Laser4C (PR205)</mod:title></td>
   </tr>
   <tr>
      <th>Description</th>
      <td><mod:description>Entry level color laser printer</mod:description></td>
   </tr>
   <tr>
      <th>Type</th>
      <td><mod:type>color laser</mod:type></td>
   </tr>
   <tr>
      <th>Items to be Built</th>
      <td><mod:ordered>320</mod:ordered></td>
   </tr>
   </table>
```

The Final Result

JE Shipping Report

Model Order

Title	Laser4C (PR205)
Description	Entry level color laser printer
Type	color laser
Items to be Built	320

Parts List

- Title Chassis and Roller Kit
- Description PR205 chassis and rollers
- Parts in Stock 512

- Title Fuser Assembly
- Description Fuser assembly/JE series
- Parts in Stock 1253

- Title Engine Kit
- Description Printer engine kit/JE series
- Parts in Stock 3895

- Title Controller board
- Description PR205 printer controller board
- Parts in Stock 483

- Title Toner Kit
- Description PR205 toner kit (b,m,c,y)
- Parts in Stock 812

The END!!!