CITS1231 Web Technologies
Working with XML Namespace and XML Trees
Objectives

• XML Namespace
• URIs, URLs, and URNs
• Default Namespace
• CSS and Namespace
• XHTML
• XML Tree
• HTML Tree
Standard XML Vocabularies

- An XML vocabulary is a set of XML tags for a particular industry or business function. Examples:
Combining XML Vocabularies

- A document that combines several vocabularies is known as a **compound document**
Working with Namespaces

• **Name collision** occurs when elements from two or more vocabularies used in a compound document share the same name.

• Name collision is not a problem if you are not concerned with validation. The document content only needs to be well-formed.

• However, name collision will keep a document from being validated.
Name Collision Example

```xml
<model>
  <table>
    <tr>
      <td>Gold</td>
      <td>Sliver</td>
    </tr>
  </table>
  <table>
    <material>Jarrah</material>
    <cost>$100</cost>
  </table>
</model>
```

- Name collision!
- Two `<table>` elements:
  - different content
  - different meaning
Resolve Name Collision Via Prefix

\[
\begin{align*}
\text{Conflict resolved since } & <a:table> \text{ and } <b:table> \\
& \text{have different names.}
\end{align*}
\]
Declaring a Namespace

- To use a prefix in XML, a namespace must be declared.
- Syntax for an attribute used to declare namespace in the prolog:
  xmlns:prefix="URI"
- Where URI is a Uniform Resource Identifier that assigns a unique name to the namespace.
- prefix is a string of letters that associates each element or attribute in the document with the declared namespace.
- For example:
  <mode1 xmlns:mod="http://jacksonselect.com/models”>
- Declares a namespace with the prefix “mod” and the URI http://jacksonselect.com/models
URIs, URLs, and URNs

• The URI is a unique string of characters used to identify a name or a resource on the internet.

• URIs can be classified as locators (URLs), names (URNs) or both.

• URN functions is like a person’s name, whereas a URL is like a person’s street address.

• URNs take the form: urn:NID:NSS

• Example:
  – urn:isbn:0-486-27557-4 refers to a specific edition of Shakespeare’s play: Romeo and Juliet.
Declaring Namespace in Root

```xml
<model
    xmlns:a="http://www.w3.org/TR/html4/
    xmlns:b="http://undergraduate.csse.uwa.edu.au/units/CITS1231/furniture">
  <a:table>
    <a:tr>
      <a:td>Gold</a:td>
      <a:td>Sliver</a:td>
    </a:tr>
  </a:table>
  <b:table>
    <b:material>Jarrah</b:material>
    <b:cost>$100</b:cost>
  </b:table>
</model>
```
Declaring Namespace in Element

```xml
<model>
  <a:table xmlns:a="http://www.w3.org/TR/html4/>
    <a:tr>
      <a:td>Gold</a:td>
      <a:td>Sliver</a:td>
    </a:tr>
  </a:table>
  <b:table xmlns:b="http://undergraduate.csse.uwa.edu.au/units/CITS1231/furniture">
    <b:material>Jarrah</b:material>
    <b:cost>$100</b:cost>
  </b:table>
</model>
```

Namespace declared in element where it is used
Declaring a Namespace

- Use **namespaces** to avoid name collision.
- A **namespace** is a defined collection of element and attribute names.
- Names that belong to the same namespace must be unique.
- Elements can share the same name if they reside in different namespaces.
- Namespaces must be declared before they can be used.
Qualified/Unqualified Names

- Prefixed names are called **qualified names**.
- An element name without a namespace prefix is called an **unqualified name**.
Declaring a Default Namespace

- You can specify a **default namespace** by omitting the prefix in the namespace declaration.

- The element containing the namespace attribute and all of its child elements are assumed to be part of the default namespace.

  ✔ make the code easier to read because you do not have to add the namespace prefix to each element.

  ✗ an element’s namespace is not readily apparent from the code.
Using Namespaces with Attributes

- Attributes, like elements, can become qualified by adding the namespace prefix to the attribute name.

- Example:

```xml
<!-- http://www.w3.org is bound to n1 and is the default -->
<x xmlns:n1="http://www.w3.org",
    xmlns:n2="http://www.uwa.edu.au">
    <good n2:a="2"/>
</x>
```

Attribute ‘a’ is bound to n2
Using Namespaces with Attributes

<!-- http://www.w3.org is bound to n1 and n2 -->
<x xmlns:n1="http://www.w3.org"
    xmlns:n2="http://www.w3.org">
    <bad a="1" a="2" />
    <bad n1:a="1" n2:a="2" />
</x>

No element may contain two attributes with the same name.

No element may contain two qualified attribute names with the same local part, pointing to identical namespaces, even if the prefixes are different.
Adding a Namespace to a Style Sheet

- To declare a namespace in a style sheet, you add the following rule to the style sheet file

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

  where `prefix` is the namespace prefix and `uri` is the URI of the namespace

- Example:

  ```
  @namespace mod url(http://jacksonselect.com/models)
  ```
Applying a Namespace to a Selector

• Once you’ve declared a namespace in a style sheet, you can associate selectors with that namespace using the syntax:
  \texttt{prefix\|selector \{attribute1:value1; attribute2:value2;\ldots\}}

• For example:
  \texttt{mod\|title \{width: 150px\}}

• You also can use the wildcard symbol (*) to apply a style to any element within a namespace or to elements across different namespaces
Namespaces with the Escape Character

- Not all browsers support the use of the `@namespace` rule
- Internet Explorer browser uses the backslash escape character before the namespace prefix in CSS style sheets:

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

- Browsers like Firefox, Opera, and Netscape do not support this method with XML documents
Namespace in a Style Sheet

• Summary
  – To declare a namespace in a CSS style sheet, add the following rule before any style declarations:

    @namespace prefix url(uri);

  – To apply a namespace to a selector, use the form

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

  – For Internet Explorer browsers, use the following form to apply a namespace to a selector:

    prefix\:\selector {attribute1:value1; attribute2:value2; ...}
Converting HTML to XHTML

- Insert the following xml declaration as the very first line in the file:

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

- Add the following attribute to the opening `<html>` tag:

  ```xml
  xmlns="http://www.w3.org/1999/xhtml"
  ```
XHTML Example

```xml
<?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>
```
XML Tree Structure

• Elements of XML document form a document tree.
• Each XML document has a unique root element.
• XML tree starts at the root element and branches to lowest elements in following way:
  – Each element has a branch to each of its attributes.
  – Each element has a branch to any text it contains.
  – Each element has a branch to each of its children.
XML Tree Example (www.w3schools.com)

<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
</bookstore>
HTML Tree Example (www.w3schools.com)

<html>
  <head>
    <title>My Title</title>
  </head>
  <body>
    <a href="link">My link</a>
    <h1>My header</h1>
  </body>
</html>
Summary

- XML Namespace
- URIs, URLs, and URNs
- Default Namespace
- CSS and Namespace
- XML and HTML Trees