UNIT 2

Cascading style sheet (CSS), what is CSS, Why we use CSS, adding CSS to your web pages, Grouping styles-extensible, Introduction to XML, uses of XML, simple XML, XML key components, DTD and Schemas, Using XML with application. Transforming XML using XSL and XSLT. Dynamic HTML: Document object model (DCOM)-Accessing HTML & CSS through DCOM Dynamic content styles & positioning-Event bubbling-data binding.

by Pratishtha Gupta

What is CSS?

- CSS stands for Cascading Style Sheets.
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files.

Why Use CSS?

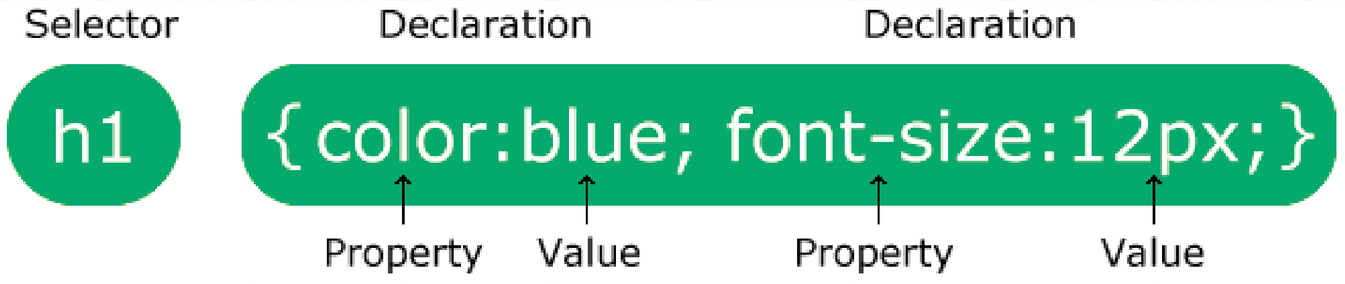
- CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.
- With an external stylesheet file, you can change the look of an entire website by changing just one file.
- The style definitions are normally saved in external .css files.

- HTML was NEVER intended to contain tags for formatting a web page!
- HTML was created to describe the content of a web page, like: <h1>This is a heading</h1>
- When tags like <font>, and color attributes were added to the HTML 3.2 specification. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.
- To solve this problem, the World Wide Web Consortium (W3C) created CSS.
- CSS removed the style formatting from the HTML page.

```html
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightblue;
}
h1 {
  color: white;
  text-align: center;
}
p {
  font-family: verdana;
  font-size: 20px;
}
</style>
</head>
<body>
<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

CSS Syntax

- A CSS rule consists of a selector and a declaration block.



- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

Example

- p is a selector in CSS (it points to the HTML element you want to style: <p>).
- color is a property, and red is the property value.
- text-align is a property, and center is the property value.

```
p {
  color: red;
  text-align: center;
}
```

Adding CSS to your Web Pages

- When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

```html
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet"
href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.
</p>

</body>
</html>
```

External CSS

```html
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}
h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Internal CSS

```html
<!DOCTYPE html>
<html>
<body>

<h1
style="color:blue;text
-align:center;">This
is a heading</h1>
<p
style="color:red;">Thi
s is a paragraph.</p>

</body>
</html>
```

Inline CSS

External CSS

- With an external style sheet, you can change the look of an entire website by changing just one file.
- Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

Example

- An external style sheet can be written in any text editor, and must be saved with a .css extension.
- The external .css file should not contain any HTML tags.

Internal CSS

- An internal style sheet may be used if one single HTML page has a unique style.
- The internal style is defined inside the <style> element, inside the head section.

Inline CSS

- An inline style may be used to apply a unique style for a single element.
- To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

Multiple Style Sheets

- If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

Example

- Assume that an external style sheet has the following style for the <h1> element.

```
h1 {
  color: navy;
}
```

- Then, assume that an internal style sheet also has the following style for the <h1> element:

```
h1 {
  color: orange;
}
```

a. If the internal style is defined after the link to the external style sheet, the <h1> elements will be "orange".

b. However, if the internal style is defined before the link to the external style sheet, the <h1> elements will be "navy".

```
<head>
<link rel="stylesheet"
type="text/css"
href="mystyle.css">
<style>
h1 {
  color: orange;
}
</style>
</head>
```

a.

```
<head>
<style>
h1 {
  color: orange;
}
</style>
<link rel="stylesheet"
type="text/css"
href="mystyle.css">
</head>
```

b.

Cascading Order

- What style will be used when there is more than one style specified for an HTML element?
- All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:
- Inline style (inside an HTML element)
- External and internal style sheets (in the head section)
- Browser default
- So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

CSS Selectors

- A CSS selector selects the HTML element(s) you want to style.
- CSS selectors are used to "find" (or select) the HTML elements you want to style.
- We can divide CSS selectors into five categories:
- **Simple selectors** (select elements based on name, id, class)
- **Combinator selectors** (select elements based on a specific relationship between them)
- **Pseudo-class selectors** (select elements based on a certain state such as visited and unvisited links, element in focus, mouse hovers over the element.)
- **Pseudo-elements selectors** (select and style a part of an element)
- **Attribute selectors** (select elements based on an attribute or attribute value)

CSS element Selector

- The element selector selects HTML elements based on the element name.

Example

- Here, all <p> elements on the page will be center-aligned, with a red text color:

```html
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p>Every paragraph will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

CSS id Selector

- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element is unique within a page, so the id selector is used to select one unique element.

- To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example

- The CSS rule below will be applied to the HTML element with id="para1".

CSS class Selector

- The class selector selects HTML elements with a specific class attribute.
- To select elements with a specific class, write a period (.) character, followed by the class name.

```html
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p id="para1">Hello World!</p>
<p>This paragraph is not
affected by the style.</p>

</body>
</html>
```

Example

- In this example all HTML elements with class="center" will be red and center-aligned.
- You can also specify that only specific HTML elements should be affected by a class.

Example

- p.center { text-align: center; color:red;}
- In this example only elements with class="center" will be red and center-aligned.
- HTML elements can also refer to more than one class.

Example

- This paragraph refers to two classes.

```html
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>

</body>
</html>
```

CSS Universal Selector

- The universal selector (*) selects all HTML elements on the page.

Example

- The CSS rule below will affect every HTML element on the page:

```
* {
  text-align: center;
  color: blue;
}
```

CSS Grouping Selector

- The grouping selector selects all the HTML elements with the same style definitions.

```
h1, h2, p {
  text-align: center;
  color: red;
}
```

CSS Comments

- Comments are used to explain the code, and may help when you edit the source code at a later date.
- Comments are ignored by browsers.
- A CSS comment is placed inside the <style> element, and starts with /* and ends with */.

Single Line Comment

- /* This is a single-line comment */

Multi-Line Comment

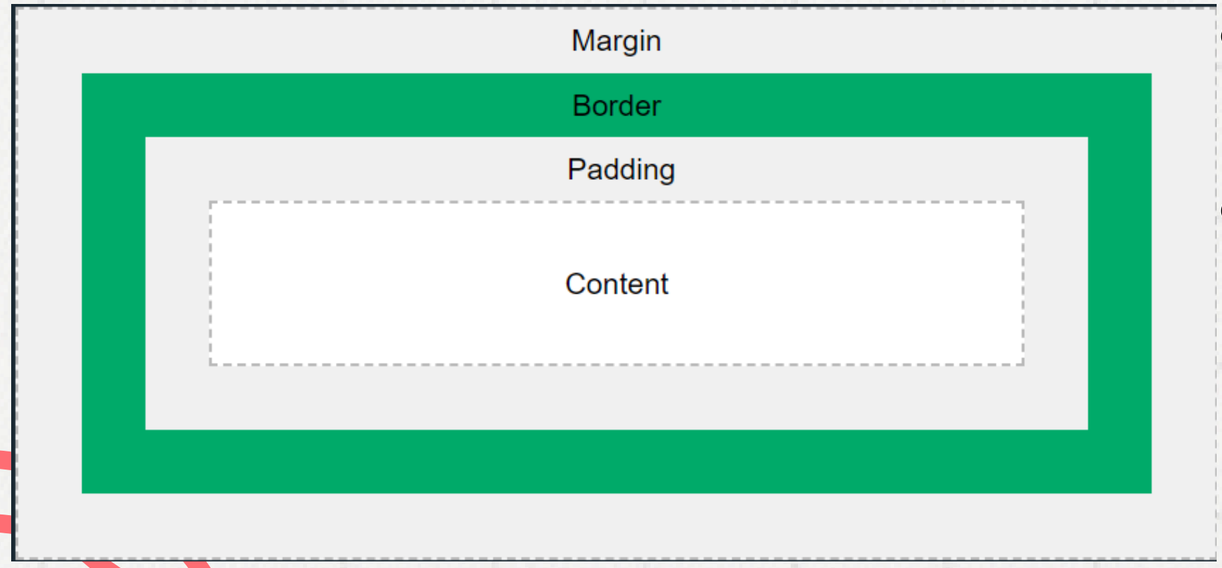
- Comments can also span multiple lines:

```
/* This is
a multi-line
comment */
```

CSS Commonly used Properties

- **CSS Color:** In CSS, you can use a color name to specify colors. Eg: Orange, Violet, Gray
- To add **color** to the text, use **color** property. Eg: <h1 style="color:Gray;">Hello World</h1>
- To add **background-color**, use **background-color** property. Eg: <h1 style="background-color:DodgerBlue;">Hello World</h1>
- To add color to **border**, use **border**: *name of color*. Eg: <h1 style="border: 2px solid DodgerBlue;">Hello World</h1>
- Colors can be specified using 3 methods: RGB values, HEX values, HSL values
- <h1 style="background-color:rgb(255, 99, 71);">rgb(255, 99, 71)</h1>
- <h1 style="background-color:#ff6347;">#ff6347</h1>
- <h1 style="background-color:hsl(9, 100%, 64%);">hsl(9, 100%, 64%)</h1>

- **CSS backgrounds**
  a. background-color
  b. background-image
  c. background-repeat
  d. background-attachment
  e. background



- **CSS Box Model**
  - The CSS box model is essentially a box that wraps around every HTML element. It consists of: content, padding, borders and margins.
  - **Padding** - Clears an area around the content. The padding is transparent.
  - **Border** - A border that goes around the padding and content.
  - **Margin** - Clears an area outside the border. The margin is transparent.

- **CSS borders**
  a. border-style
  b. border-width
  c. border-color
  d. border-sides
  e. rounded border

- **CSS Padding**
- Padding is used to create space around an element's content, inside of any defined borders.

- **CSS Margin**

a. Margins are used to create space around elements, outside of any defined borders.
b. CSS has properties for specifying the margin for each side of an element:
   i. margin-top
   ii. margin-right
   iii. margin-bottom
   iv. margin-left
c. To shorten the code, it is possible to specify all the margin properties in one property using margin property.

XML: eXtensible Markup Language

- XML is a software- and hardware-independent tool for storing and transporting data.
- XML is a markup language much like HTML.
- XML was designed to store and transport data and to be self-descriptive.
- XML is a W3C Recommendation(a standard or guideline that the World Wide Web Consortium (W3C) endorses for web technologies).
- XML is often used for distributing data over the Internet.

Example: The XML below is quite self-descriptive:

  - It has sender & receiver information
  - It has a heading
  - It has a message body

- But still, the XML above does not DO anything. XML is just information wrapped in tags.
- Someone must write a piece of software to send, receive, store, or display it.

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

- **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
- The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document unlike HTML which works with predefined tags
- With XML, the author must define both the tags and the document structure.
- **XML is Extensible**
- Most XML applications will work as expected even if new data is added (or removed).
- Imagine an application designed to display the original version of note.xml (<to> <from> <heading> <body>).
- Then imagine a newer version of note.xml with added <date> and <hour> elements, and a removed <heading>.
- The way XML is constructed, older version of the application can still work:

XML Simplifies Things

- XML simplifies data sharing
- XML simplifies data transport
- XML simplifies platform changes
- XML simplifies data availability

- Many computer systems contain data in incompatible formats.
- Exchanging data between incompatible systems (or upgraded systems) is a time-consuming task for web developers.
- Large amounts of data must be converted, and incompatible data is often lost.
- XML stores data in plain text format.
- This provides a software- and hardware-independent way of storing, transporting, and sharing data.
- XML also makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

XML Separates Data from Presentation

- XML does not carry any information about how to be displayed.
- The same XML data can be used in many different presentation scenarios.
- Because of this, with XML, there is a full separation between data and presentation.

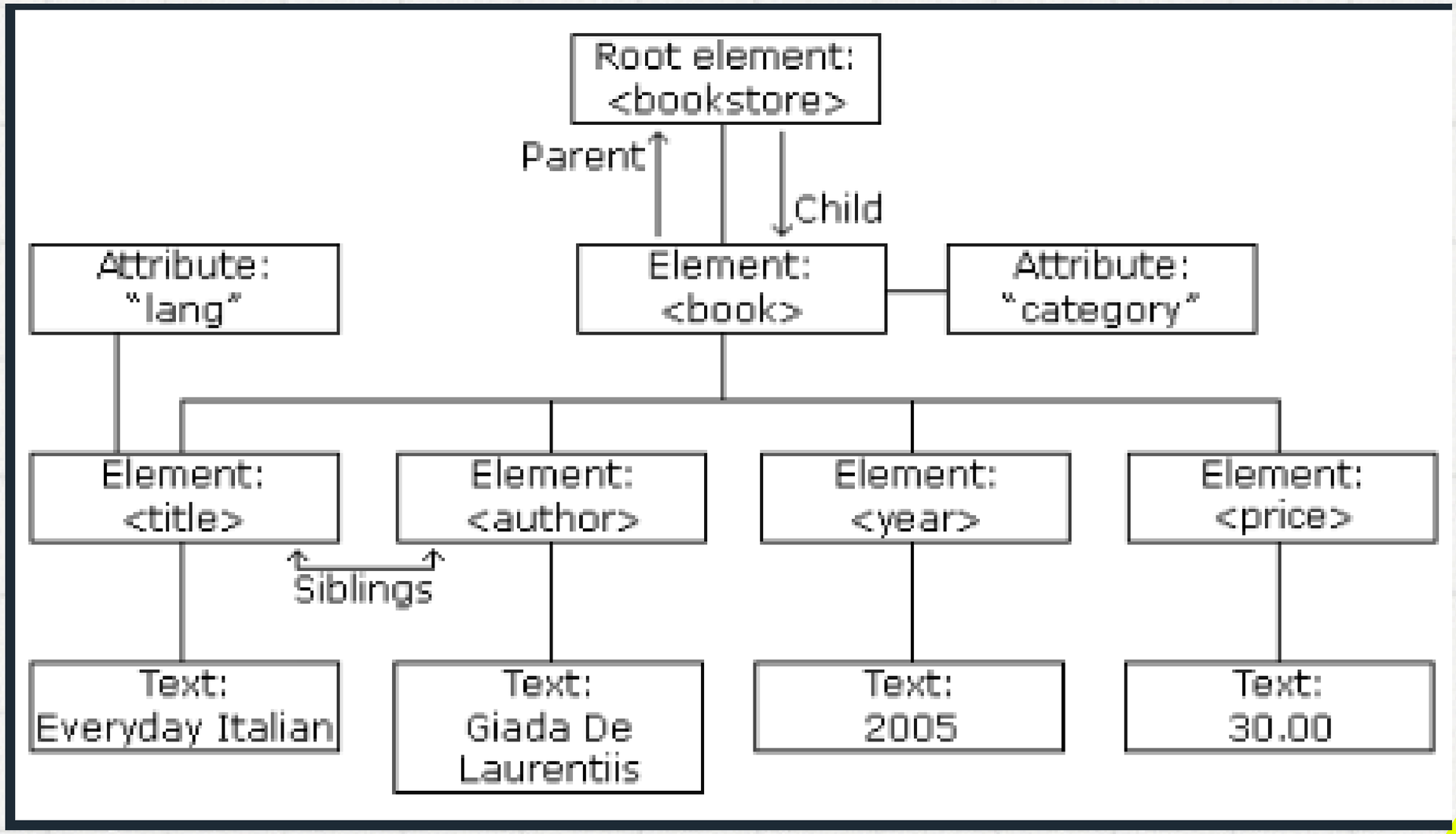
XML is Often a Complement to HTML

- In many HTML applications, XML is used to store or transport data, while HTML is used to format and display the same data.

XML Separates Data from HTML

- When displaying data in HTML, you should not have to edit the HTML file when the data changes.
- With XML, the data can be stored in separate XML files.
- With a few lines of JavaScript code, you can read an XML file and update the data content of any HTML page.

XML Tree Structure



- XML documents are formed as element trees.
- An XML tree starts at a root element and branches from the root to child elements.
- All elements can have sub-elements (child elements):
- The terms parent, child, and sibling are used to describe the relationships between elements.
- Parents have children. Siblings are children on the same level (brothers and sisters).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

XML Syntax Rules

- XML Documents must contain one root element that is the parent of all other elements.
- All XML Elements Must Have a Closing Tag.
- XML Tags are Case Sensitive
- XML Elements Must be Properly Nested.

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

- XML elements can have attributes in name/value pairs just like in HTML which must always be quoted
- Some characters have a special meaning in XML.
- If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

```
<b><i>This text is bold and italic</i></b>
```

```
<message>salary < 1000</message>
```

- To avoid this error, replace the "<" character with an entity reference:

```
<message>salary &lt; 1000</message>
```

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

Comments in XML

The syntax for writing comments in XML is similar to that of HTML:

```
<!-- This is a comment -->
```

White-space is Preserved in XML

XML does not truncate multiple white-spaces (HTML truncates multiple white-spaces to one single white-space):

XML:	Hello	Tove
HTML:	Hello Tove	

XML Element

An XML element is everything from (including) the element's start tag to (including) the element's end tag.

```
<price>29.99</price>
```

An element can contain:

- text
- attributes
- other elements
- or a mix of the above

In the example

- <title>, <author>, <year>, and <price> have text content because they contain text.
- <bookstore> and <book> have element contents, because they contain elements.
- <book> has an attribute (category="children").

```
<bookstore>
  <book category="children">
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Empty XML Elements

- An element with no content is said to be empty.
- You can also use a so called self-closing tag

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

```
<element />
```

XML Naming Rules

- Element names are case-sensitive
- Element names must start with a letter or underscore
- Element names cannot contain spaces.
- Element names cannot start with the letters xml (or XML, or Xml, etc).
- Element names can contain letters, digits, hyphens, underscores, and periods.

Best Naming Practices

- **Create descriptive name:** like this: <person>, <firstname>, <lastname>.
- **Create short and simple names:** like this: <book_title> not like this: <the_title_of_the_book>.
- **Avoid "-":** If you name something "first-name", some software may think you want to subtract "name" from "first".
- **Avoid ".":** If you name something "first.name", some software may think that "name" is a property of the object "first".
- **Avoid ":":** Colons are reserved for namespaces.

XML Elements are Extensible

- XML elements can be extended to carry more information
- Consider that we created an application that extracted the <to>, <from>, and <body> elements from the XML document to produce this output.
- Consider that the author of the XML document added some extra information to it.

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <body>Don't forget me this weekend!</body>
</note>
```

```
<note>
  <date>2008-01-10</date>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

- The application should still be able to find the <to>, <from>, and <body> elements in the XML document and produce the same output (without breaking applications.)

- **XML Attributes**
- XML elements can have attributes, just like HTML.
- Attributes are designed to contain data related to a specific element.
- XML Attributes Must be Quoted: Either single or double quotes can be used.

Style	Example	Description
Lower case	<firstname>	All letters lower case
Upper case	<FIRSTNAME>	All letters upper case
Snake case	<first_name>	Underscore separates words (commonly used in SQL databases)
Pascal case	<FirstName>	Uppercase first letter in each word (commonly used by C programmers)
Camel case	<firstName>	Uppercase first letter in each word except the first (commonly used in JavaScript)

Avoid XML Attributes?

Some things to consider when using attributes are:

- attributes cannot contain multiple values (elements can)
- attributes cannot contain tree structures (elements can)
- attributes are not easily expandable (for future changes)

XML Attributes for Metadata

Sometimes ID references are assigned to elements. These IDs can be used to identify XML elements in much the same way as the id attribute in HTML. This example demonstrates this:

```xml
<messages>
  <note id="501">
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
  <note id="502">
    <to>Jani</to>
    <from>Tove</from>
    <heading>Re: Reminder</heading>
    <body>I will not</body>
  </note>
</messages>
```

- The id attributes above are for identifying the different notes. It is not a part of the note itself.
- Metadata (data about data) should be stored as attributes, and the data itself should be stored as elements.
- **XML Namespaces**
- XML Namespaces provide a method to avoid element name conflicts.
- In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.
- This XML carries HTML table information:

```xml
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

- If these XML fragments were added together, there would be a name conflict. Both contain a <table> element, but the elements have different content and meaning.
- A user or an XML application will not know how to handle these differences.

```xml
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

Solving the Name Conflict Using a Prefix

- Name conflicts in XML can easily be avoided using a name prefix.
- This XML carries information about an HTML table, and a piece of furniture:
- In the example above, there will be no conflict because the two <table> elements have different names.

XML Namespaces - The xmlns Attribute

- When using prefixes in XML, a namespace for the prefix must be defined.
- The namespace can be defined by an xmlns attribute in the start tag of an element.
- The namespace declaration has the following syntax. xmlns:prefix="URI".

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

In the example above:

- The xmlns attribute in the first <table> element gives the h: prefix a qualified namespace.
- The xmlns attribute in the second <table> element gives the f: prefix a qualified namespace.

- When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace.
- Namespaces can also be declared in the XML root element:
- The namespace URI is not used by the parser to look up information.
- The purpose of using an URI is to give the namespace a unique name.
- However, companies often use the namespace as a pointer to a web page containing namespace information.

```
<root>

<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="https://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```

Namespaces in Real Use

- XSLT is a language that can be used to transform XML documents into other formats.
- The XML document below, is a document used to transform XML into HTML.
- The namespace "http://www.w3.org/1999/XSL/Transform" identifies XSLT elements inside an HTML document:

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

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
<html>
<body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr>
      <th style="text-align:left">Title</th>
      <th style="text-align:left">Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

This XML carries HTML table information

```xml
<table xmlns="http://www.w3.org/TR/html4/"
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```