CGT 353: Principles of Interactive and Dynamic Media
XML and E4X with ActionScript 3.0

Intro:

- Flash has had the ability to work with XML since AS 1.0
- In AS 2.0, we use the *XML* class, which was based on the W3C Document Object Model (DOM)
- As with most other things in AS 3.0, the toolset for manipulating XML has been changed.
- AS 3.0 now uses ***ECMAScript for XML***…also known as **E4X**
- E4X is an official ECMA-262 extension for working with XML as a native datatype
- Used to improve the integration of XML with scripting languages like JavaScript and ActionScript

XML data offers several advantages over URL-encoded data, including:

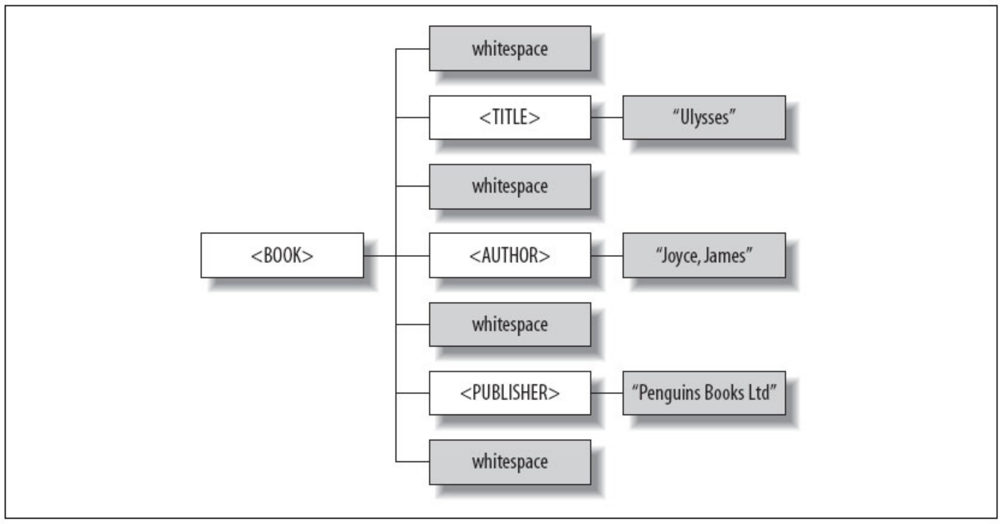
- When creating XML manually (for a static XML document) or programmatically (from a ColdFusion script, PHP script, etc.), it is much easier to represent complex data.
- Most server-side scripting languages offer built-in functionality for reading and generating XML data.
- XML is a standard used for the transfer and storage of data across all kinds of applications and platforms.

An XML Review:

- Take the following code from the text:

```
<BOOK ISBN="0141182806">
        <TITLE>Ulysses</TITLE>
        <AUTHOR>Joyce, James</AUTHOR>
        <PUBLISHER>Penguin Books Ltd</PUBLISHER>
</BOOK>
```

- Nodes, parents, children, attributes, elements…heard it before.
- What’s the first child node of <BOOK>?



- That's right…it's the first *insignificant whitespace*.

- In E4X, whitespace counts:
    - carriage return (\u000D)
    - space (\u0020)
    - line feed (\u000A)
    - tab (\u0009)
- So technically <BOOK> has seven nodes, not four.
- By default whitespace nodes are ignored by the E4X parser, so we don't really have to worry about it.

Terminology:

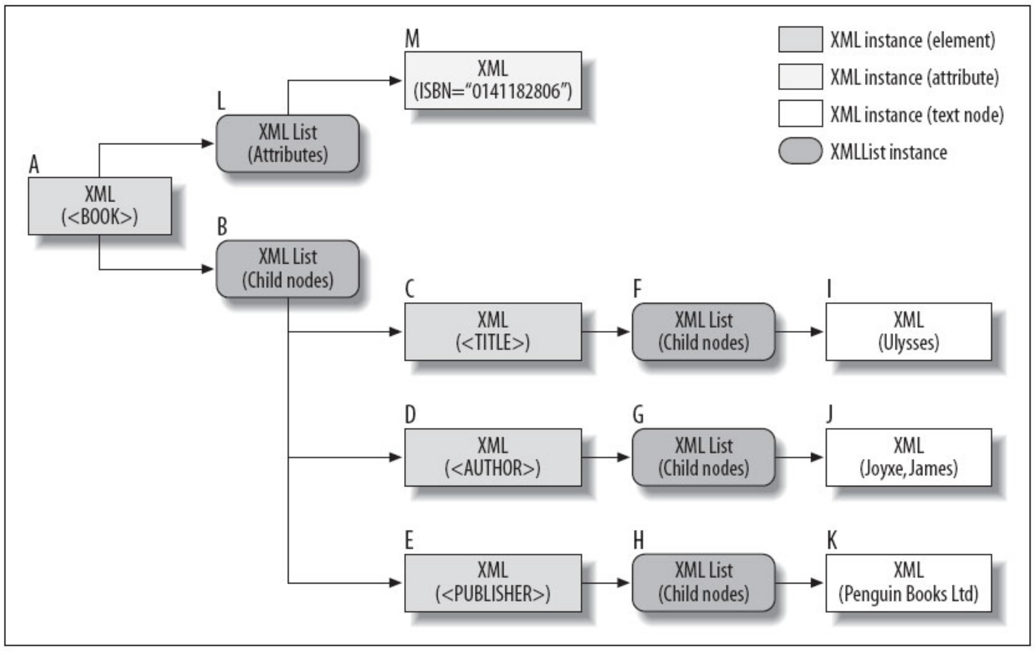
1. **XML document** - A file that contains XML. This term may also refer to XML data that is being loaded or sent. An XML document is not to be confused with the *XMLDocument* class.
2. **XML packet** - An XML packet can be any snippet of XML—from an entire XML document to just a single node—as long as it is represents a complete, well-formed piece of information in XML.
3. **XML node** - The basic building block for XML. Nodes can be elements, text nodes, attributes, and so on. We refer to elements and text nodes collectively as "nodes" when talking in general terms.
4. **XML element** - The term "element" is often used interchangeably with the term "tag." More accurately, however, an element contains tags. Elements must have an opening and closing tag (<element></element>), or the opening and closing tags can be combined into one when the element has no nested elements (<element />).
5. **Root node** - An element that is at the top of the XML hierarchy of elements.
6. **Text node** - A node containing text. Text nodes are generally nested within elements.
7. **Attribute** - Is part of an element. Attributes are placed within the tags of elements in name/value pair format, such as <element name="value">.
8. **XML declaration** - The declaration typically looks like this: <?xml version="1.0" ?>. This is a special tag that the XML parser recognizes as containing information about the XML document, and it is not parsed as an element.
9. **XML tree** - Also sometimes called the "data tree," an XML tree is the hierarchy of nodes in XML data.

Representing XML Data in E4X:

- In E4X, XML data is represented by one of two datatypes, *XML* and *XMLList*
- Corresponding classes are also named *XML* and *XMLList*
- The *XML* class from ActionScript 1.0 and ActionScript 2.0 has been renamed to *XMLDocument* in ActionScript 3.0 and moved to the flash.xml package.
- Each *XML* instance represents one of five possible kinds of XML content, or ***node kinds***.
    - Elements
    - Attribute
    - Text nodes
    - Comments
    - Processing instructions
- If an XML element has any child nodes or elements, they are wrapped in an ***XMLList*** by the parent instance.

```
<BOOK ISBN="0141182806">
        <TITLE>Ulysses</TITLE>
        <AUTHOR>Joyce, James</AUTHOR>
        <PUBLISHER>Penguin Books Ltd</PUBLISHER>
</BOOK>
```

- <BOOK> contains **two** *XMLList* instances—one for <BOOK>'s attributes and the other for its child elements.
- The <BOOK> element has only one attribute, so the *XMLList* for <BOOK>'s attributes contains one *XML* instance only



Creating XML Data with E4X:

Three Options:

1. Use the *XML* constructor to create a new *XML* instance, then create the remainder of the fragment programmatically
2. Use the *XML* constructor to create a new *XML* instance, then import the fragment from an externally loaded file
3. Write our *XML* data in literal form, just like a string or a number, anywhere literals are allowed by ActionScript.

Option 3:

```
var novel:XML = <BOOK ISBN="0141182806">
        <TITLE>Ulysses</TITLE>
        <AUTHOR>Joyce, James</AUTHOR>
        <PUBLISHER>Penguin Books Ltd</PUBLISHER>
</BOOK>;
```

- Notice that the line breaks and quotation marks are normal. ActionScript knows they are part of the XML data.

- ActionScript also allows dynamic expressions to be used so that element names, attribute names, attribute values, and element content can be generated programmatically.

- To specify a dynamic expression within an XML literal, surround it in curly braces ({ }).

Example 1.

```
var elementName:String = "BOOK";
var novel:XML = <{elementName}/>;
```

Example 2.

```
var rootElementName:String = "BOOK";
var rootAttributeName:String = "ISBN";
var childElementNames:Array = ["TITLE", "AUTHOR", "PUBLISHER"];
var bookISBN:String = "0141182806";
var bookTitle:String = "Ulysses";
var bookAuthor:String = "Joyce, James";
var bookPublisher:String = "Penguin Books Ltd";
var novel:XML = <{rootElementName} {rootAttributeName}={bookISBN}>
        <{childElementNames[0]}>{bookTitle}</{childElementNames[0]}>
        <{childElementNames[1]}>{bookAuthor}</{childElementNames[1]}>
        <{childElementNames[2]}>{bookPublisher}</{childElementNames[2]}>
</{rootElementName}>;
```

Example 3.

```
// Create the XML structure with a string so use the value of both
// username and score inside of the XML packet.
var str:String = "<gamescore><username>" + username + "</username>"
       + "<score>" + score + "</score></gamescore>";

// Pass the string to the constructor to create an XML object
var example:XML = new XML( str );
```

Accessing XML Data:

Two General Sets of Tools:

1. The *XML* and *XMLList* content-access methods (*attribute( )*, *attributes( )*, *child( )*, *children( )*, *comments( )*, *descendants( )*, *elements( )*, *parent( )*, *processingInstructions( )*, and *text( )*)

2. Variable-style access with the dot (.), descendant (..), and attribute (@) operators

- Variable-style access is offered as a convenience and always equates to one of the methods of either the *XML* or *XMLList* classes.

- The two approaches do not overlap completely.

- The following types of content must be accessed using the appropriate method of the *XML* or *XMLList* class:

1. An*XML* instance's parent (accessed via *parent( )*)
2. Comments (accessed via *comments( )*)
3. Processing instructions (accessed via *processingInstructions( )*)
4. Elements or attributes whose names include characters considered illegal in an
5. ActionScript identifier (accessed via *attribute( )*, *child( )*, *descendants( )*, or*elements( )*)