CGT 353: Principles of Interactive and Dynamic Media
XML and E4X – Pt 2

Accessing XML Data:

E4X offers two general sets of tools for accessing data in an XML hierarchy:

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

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

Adding Elements to an XML Object:

```
// Create an XML instance to add elements to
var example:XML = <example />;

// Create a new XML node named newElement and add it to the
// example instance
example.newElement = <newElement />;

/* Displays:
 <example>
   <newElement/>
 </example>
*/
trace( example );
```

You can also create a new element by creating a property on the XML instance and assigning it a value, as follows:

```
// Create an XML instance to work with
var example:XML = <example />;

// Create a new property emptyElement and assign it an empty
// string value
example.emptyElement = "";

/* Displays:
 <example>
   <emptyElement/>
 </example>
*/
trace( example );
```

Accessing the Root XML Node:

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

        </BOOK>;
```

- To access the root <BOOK> element of that fragment we refer to it as just **novel.**

```
addToOrder(novel); // Right

addToOrder(novel.BOOK); // Wrong.
addToOrder(novel.child("BOOK")); // Also wrong.
```

- There is no direct way to access the root node relative to any given child.
- However, we can use the *XML* class's instance method *parent( )* to ascend a tree recursively to its root

```
// Returns the root of an XML hierarchy, relative to a given child
public function getRoot (childNode:XML):XML {
        var parentNode:XML = childNode.parent( );
        if (parentNode != null) {
        return getRoot(parentNode);
                } else {
                        return childNode;
                }
                }
// Usage:
        getRoot(someChild);
```

Accessing Child Nodes:

- Use the children() function to access child nodes

```
novel.children( ) // Returns an XMLList representing <BOOK>'s child nodes

novel.* // Also returns an XMLList, representing <BOOK>'s child node
```

To access a specific child in an *XMLList* we use the familiar array-element access operator, [].

```
        novel.children( )[1] // A reference to <BOOK>'s second child node
        or:
        novel.*[1] // Also a reference to <BOOK>'s second child node
```

- There is no firstChild or lastChild variable in E4X
- First child in a list of child nodes can be accessed as follows:

theNode.children( )[0]

And the last child in a list of child nodes can be accessed as follows:

theNode.children( )[*theNode*.children().length( )-1]

- Accessing child nodes according to its position in a list can be cumbersome, and has been deemphasized by E4X.
- In E4X, child nodes are typically accessed by their element names rather than their position.
- To retrieve an *XMLList* of all children of <BOOK> named "AUTHOR", we use:

```
novel.child("AUTHOR") // Returns all child elements of <BOOK> named "AUTHOR"

or, using variable access syntax….

novel.AUTHOR // Also returns all child elements of <BOOK> named "AUTHOR"
```

Ex 2.

```
var novel:XML = <BOOK>
<AUTHOR>Jacobs, Tom</AUTHOR>
<AUTHOR>Schumacher, Jonathan</AUTHOR>
</BOOK>;
novel.AUTHOR[0]; // Access <AUTHOR>Jacobs, Tom</AUTHOR>
novel.AUTHOR[1]; // Access <AUTHOR>Schumacher, Jonathan</AUTHOR>
```

Accessing Text Nodes:

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

novel.TITLE.children( )[0] // A reference to the text node Ulysses
```

Or, alternatively, we can use the properties wildcard to do the same thing:

```
novel.TITLE.*[0] // Also a reference to the text node Ulysses

novel.TITLE.*[0].parent( ) // Reference to the <TITLE> element
novel.TITLE.*[0].nodeKind( ) // Returns the string "text"
novel.TITLE.*[0].toString( ) // Returns the string "Ulysses"
```

Adding Text Nodes:

```
// Create an XML instance to work with
var example:XML = <example/>;

// Create a text node from a string
example.firstname = "Darron";

// Create a text node from a number
example.number = 24.9;

// Create a text node from a boolean
example.boolean = true;
```

```
// Create a text node from an array
example.abc = ["a", undefined, "b", "c", null, 7, false];

/* Displays:
<example>
 <firstname>Darron</firstname>
 <number>24.9</number>
 <boolean>true</boolean>
 <abc>a,,b,c,,7,false</abc>
</example>
*/
trace( example );
```

Accessing Parent Nodes:

```
var pub:XML = novel.PUBLISHER[0];
```

- To access <PUBLISHER>'s parent (which is <BOOK>), we use:

```
pub.parent( )
```

- The *parent( )* method can also be used successively to access any ancestor node:

```
// Create a 3-tier XML hierarchy.
var doc:XML = <grandparent><parent><child></child></parent></grandparent>;

// Assign a reference to <child>
var kid:XML = doc.parent.child[0];

// Use parent( ) successively to access <grandparent> from <child>
        var grandparent:XML = kid.parent().parent( );
```

Accessing Sibling Nodes:

- A sibling node is a node that resides directly beside another node on a given level of an XML hierarchy
- E4X reduces the emphasis on accessing siblings due to its increased focus on accessing elements by name.

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

- In E4X there is no built-in support for moving between sibling nodes in an XML hierarchy.
- The DOM-based nextSibling, previousSibling variables are not part of the E4X API
- What should you use?

Next sibling:

```
someNode.parent( ).*[someNode.childIndex( )+1];
```

Previous sibling:

```
someNode.parent( ).*[someNode.childIndex( )-1];
```

Accessing and Adding Attributes:

- To access an *XMLList* representing all of an element's attributes, we use the *XML* class's instance method *attributes( )*
- Alternatively, we can access an *XMLList* representing an element's attributes using the E4X *attributes wildcard* (@*)

  someElement.@* // Returns an *XMLList* representing all of *someElement*'s attributes

- Use the @ operator in E4X syntax to assign attributes to element nodes.

```
// Create an XML instance to work with
var example:XML = <example><someElement/></example>;

// Add some attributes to the someElement element node
example.someElement.@number = 12.1;
example.someElement.@string = "example";
example.someElement.@boolean = true;
example.someElement.@array = ["a", null, 7, undefined, "c"];

/* Displays:
<example>
  <someElement number="12.1" string="example" boolean="true"
  array="a,,7,,c"/>
</example>
*/
trace( example );
```

Accessing Comments and Processing Instructions:

- The final two kinds of nodes we can access in E4X are comments and processing instructions.

  <!--*Comment text goes here*-->

  XML processing instructions take the form:

  <?*someTargetApp someData*?>

- These two forms of data can be accessed using the *XML* class's instance methods *comments( )* and *processingInstructions( )*.