TC39 TG1 Subgroup E4X (ECMAScript For XML) Meeting, October 2 and 3, 2002

Kona, Hawaii

Attendees

Andrew Clinick (Microsoft)
Rok Yu (Microsoft)
Peter Torr (Microsoft)
John Schneider (AgileDelta / BEA)
Waldemar Horwat (Netscape)
Steve Adamski (Netscape)

Agenda

Revised Schedule
Initial Draft
Prioritise Design Principals
Use Cases
Data Model (Day 2)
Types and Type Conversions (ran out of time)

Revised Schedule

The revised schedule is still aggressive, but looks OK for the time being. A first draft should be ready by the March 2003 TC39 meeting.

Initial Draft

The initial skeleton draft of the standard was acceptable to everyone.

Prioritise Design Principles

Some discussion of the principles listed in John's draft and the relative importance of each principle. It was suggested that the list of seven principles be broken into two buckets, the "Pri 1" bucket consisting of the first four principles (simple, consistent, familiar, minimal) and a "rest" bucket consisting of the last three principles (complimentary, secure, loose). It was further decided after some debate that the priority order of the first bucket should stand unchanged, but that in the "rest" bucket the "secure" principle renamed to "Maintains data integrity", and that the "complementary" principle be moved to the bottom of the list.

Use Cases

Rok had undertaken some initial investigations into the use cases of "competing" technologies, such as XQuery, but did not have anything to present at this meeting. He will prepare something for the next meeting.

We discussed a user case presented by John dealing with mapping XML to objects. Mike Shenfield and Jeff Dyer are also working on use cases.

Data Model (Day 2)

John presented some initial ideas on the data model for E4X, and the discussions generated from this took up most of the meeting. This resulted in some questions / issues which John will work on for the next meeting.

Preserving exact document content. Should E4X preserve the exact content of an XML document (including optional whitespace inside tags, attribute order, and entity references) or should it merely preserve the semantics of the document? By definition, two XML documents can be semantically equivalent even though they may not compare successfully with a character-by-character comparison. There appears to be a need to preserve the original content when serialising XML back into a file format, but also a need to canonicalise XML data into a well-defined string format when used inside a program. For example, if a program reads an XML document into memory and writes it back to disk without making any changes, the two

versions of the file (before and after) should be identical. On the other hand, if a script loads two equivalent-but-different XML files and does a string comparison on them, the comparison should be successful.

Nevertheless, maintaining these equivalencies is problematic. It was generally agreed that preserving non-significant white space would be a non-goal, although Peter will work on a use case for this.

Are attributes represented as objects? Is an attribute simply a string value stored as a named property on an XML object, or is it a complex object {name, value, parent [, type]}? The object is needed if developers want to walk the DOM backwards from an attribute. For example, a query to find all order totals greater than 100 will return a list of attributes. From those attributes, a developer will want to get back to the containing order element to find out additional details.

Dot operator and lists. How does the dot operator work when applied to lists? What happens when it returns a list with a single element? Ideally the single-element-list should appear to the programmer as the element itself ('foo' as opposed to 'list-of-foo'), but there are problems with type identity. Having the dot operator work on lists and return lists is also inconsistent with existing ECMAScript objects. One suggestion was to introduce a new operator, .#, that could return a list of items. The dot operator would then throw an exception if it found more than one element to return. Whilst this might make programs more robust, it also adds confusion.

Plus operator. The plus operator is currently overloaded to perform list concatenation and to append elements to lists; this could be confusing, especially when dealing with strings, singleton lists, or other cases.

Name collisions. What happens if an element contains a sub element named 'length' or (unlikely) 'toString'? One potential approach is to keep method names in a separate namespace, and define no standard properties for XML objects. In this scenario, x.y is differentiated from x.y(), and so the length property would be replaced by a length() method. All properties (without parentheses) would be sub-elements in the XML tree, and all methods (with parentheses) would be methods. (This could cause a problem for method extraction / replacement).

Data types stored in XML objects. It was agreed that all data stored in XML objects should be stored as strings (assuming there is no schema attached). Even if a developer makes an assignment such as

```
unit.price = 50
```

then the element's value is the string "50" and not the number 50.

Parent paths. Some discussion about what happens with parent pointers (do we really need them?) and how copies of XML fragments are made to ensure that each 'rooted' XML tree has exactly one parent and occurs exactly once. XML literals embeddd in the source code would be copied every time they are used, so that (for example) a loop containing a literal with an embedded expression would work correctly.

Should the new XML objects be prototype based or class based? Should XML lists derived from the Array type?

Should lists store lvalues or rvalues? Becomes important for being able to replace parts of the XML tree with other chunks of XML, for example:

```
customer.order[3] = <order>bla bla</order>
```

It would be nice if this actually replaced the 4th order in the XML document, rather than just updated the list.

It was suggested that changing Edition 4's syntax for namespace lookup to use a single colon instead of a double colon would increase familiarity with those users used to XML namespace lookups. Waldemar will investigate to see if this is possible with Edition 4's current grammar. (It turns out that this is not possible as it is ambiguous with respect to labelled statements, the ternerary operator, and so on).