TC39 TG1 E4X Meeting Notes, 2003-05-09

Date	May 19, 2003 10:00 AM – 5:00 PM
Location	Microsoft Corporation Building 41/3731 One Microsoft Way Redmond, WA 98052
Convener	Rok Yu (Microsoft)
Editor	John Scheider (BEA/AgileDelta)
Participants	John Schneider (BEA/AgileDelta) Rok Yu (Microsoft) Waldemar Horwat (Netscape)

Agenda

The agenda was adopted as written with an additional item:

- John: Discuss format and how formal algorithms are to be written

Schedule and Next Meetings

John states according to the schedule, we should be starting review of finalized wording. We are approximately one month behind schedule.

The details of the meeting schedule were discussed the previous day during the E4 meeting. The increased meeting frequency needed for E4 align with E4X and meetings for the two groups will continue to coincide.

Members agreed to move to a biweekly meeting schedule, alternating between conference calls and face-to-face meetings. The following meetings have been scheduled. Unless otherwise noted:

- First day of face-to-face meetings is for E4, second for E4X
- Face-to-face meetings are from 10:00 AM – 5:00 PM on the first day to allow for morning travel, and 9:30 AM – 5:00 PM on subsequent days.
- Conference calls are from 10:00 AM – 12:00 PM
- Times are in PDT

Date	Venue	Notes
May 22-23	Conference call	
June 5-6	Meeting at Netscape	
June 19-20	Conference call	
July 1-2	Meeting at Microsoft	Moved earlier to accommodate July 4th long weekend.
July 16-17	Meeting at Netscape	End of July meeting face-to-face meeting moved here because original dates conflicted with member schedules.

July 25	Conference call from 10:00 AM – 11:00 AM	Meeting to decide which specifications TG1 will submit for September ratification.

Review of Editor Action Items from Previous Meetings

- John to factor out descendent operator behavior into a separate section which is called from both the syntactic invocation (“..” operator) and the method.

John added internal descendant method [[Descendants]](P) to XML objects for accessing descendants based on dynamically computed property names and factored descendant algorithm into internal properties of XML and XMLList types. Details are discussed in review of new sections.

- Issue 19: Ability to test for children inside filtering predicate with dynamically computed property name

John added child method to XML objects for accessing children from inside filtering predicates

Details to be discussed in new stuff

- John to look up spelling of descendant in XPath and apply to entire spec.

XPath spells descendants spelled w/ an “a”. John renamed descendent to descendant through document.

- John will review and add necessary grammar to CallExpression.

John added CallExpression productions to grammar

- John to add example for case of coercion to number and all text is stored as strings.

John added explanation and examples regarding how to effect numeric addition and string concatenation on XML leaf nodes.

- John to remove this [10.1 Namespace Statement] section

John has removed the section.

- John to rewrite algorithm based to test object being iterated and dispatch to either existing E3 algorithm or new algorithm and add text as in E3 indicating enumeration behavior is unspecified if enumerated object is mutated during enumeration.

John modified for-in semantics to make order relevant only when enumerating XML child element properties.

- John to add method to validate identifier and do validation against a schema.

John added an isXMLName() method to global object for dynamically checking whether a particular string is a valid name for an XML element or attribute.

Other Changes

John added text node support to ToString on XML.

ToString on XML fragment

John proposes changing behavior of ToString on XMLList to not have “,”. This makes coercing the sum of two XML lists to string behave more like summing the individual lists to strings and summing.

Waldemar is not convinced this is the correct behavior and notes we decided the other way the last meeting.

Action Items

- John to look for cases where String is special cased in E3 spec to understand consequences of making this change.

ToXML applied to string

John had a question regarding how to handle ToXML applied to a string and what to do about invalid XML characters.

Waldemar suggests that the string be parsed as escaped text.

The working group agrees to this behavior.

Action Items

- John to modify ToXML on strings to parse the string as an escaped document fragment

ECMA 323 Voice Control Use Case

Rok presented the port of the ECMA 323 use case ported to use E4X features. The details of the port are at http://www.ecmadoc.net/docfiles/TC39-g1/2003/2003tg1-010.pdf. The following is a summary of observations and responses.

- Use of selectSingleNode very common – what is E4X equivalent?

→ use [0] e.g. Instead of msg.*.phoneNumber, use msg.*.phoneNumber[0]

- Test for 'empty result' is ugly – if (answer.length() == 0)

→ Consider modifying equality operator to compare XML list of length 0 equal to null

- Use of "." for context object looks weird – too much punctuation
- It is important for host provided object model to be typed. Early coercision out of XML type makes it easier to reason how a program behaves.
- XML literal { } syntax is very useful
- Method calls for properties is ugly. Perhaps we can denote reserved namespace for these?

```
namespace e4x("http://www.e4x.com")
if (msg.e4x::name == "test")…
```

- XML embedded in HTML/XML is very confusing
- There are quite a few casts to XML type? Do we need a shortcut for this cast?

John proposes the { } in XML literal syntax is a substitution that occurs at runtime into the XML text. The final text fragment is parsed as a whole after the substitutions occur.

Waldemar points out that this means that substations can change the structure of the XML.

```
e.g.  <foo>{expr1}<bar>{expr2}</bar></foo>
      expr1 = "</foo>" and expr2 = "</bar><foo><bar>"
```

Waldemar would like to avoid this as well as in the case where the substituted string is XML, avoid having XML be double escaped.

The XML literal design still needs to be designed and ratified. This will occur at a future meeting.

Proposals for ToXML on Objects and Unification of XML and XMLList Initializers

Rok has withdrawn these proposals as they are not blockers for adopting the standard.

The working group agrees that these are lower priority and have moved them to the pri 2 list.

Algorithm Format

John wanted to get clarification on what sort of notation the formal algorithms should be written in. In particular, he'd like to use higher level concepts such as loops and variables to avoid using goto step N and result of step N constructs.

Waldemar states that you can pretty much use anything as long as notation is defined. He states E3 regular expressions uses additional notation that John can take advantage of.

John proposes that he will use a hybrid between E3 and E4. In particular he will use basic looping constructs and variables.

Working group agrees that this notation is acceptable.

Review of Sections

Sections 11.1 The Global Object to 11.2.4.21 processingInstruction ([name]) were reviewed.

11.1 The Global object

John added 11.1 and subsections to describe members E4X adds to the global object. "11.1.1.1 isXMLName (string)" contains a definition to examine a string and return whether it can be used as an XML element or attribute name. Other sections are short stubs which point to objects defined elsewhere in the specification.

11.1.1.1 isXMLName (string)

The current definition of isXMLName uses the Letter, Digit, CombiningChar, and Extender definitions found in the XML 1.0 specification. Waldemar questions whether this is necessary, or whether it is even acceptable for an ECMA specification to have a normative reference to a W3C specification and suggests if possible, we should embed the definitions into the normative section and have a non-normative references.

John states that XML 1.0 has a long complex set of character ranges for that define valid tag and attribute names – too large to embed. John suggests we look at XML 1.1 to see if the definitions has changed.

The working group took a look at the XML 1.1 specification which is much more permissive on the set of characters allowed in element and attribute names and the section which describes the constraints has become non-normative. It also allows a much larger set of code points including Cf characters which may be stripped.

The working group needs to investigate how E4X will interact with XML 1.1.

Action Items

- John to review the XML 1.1 specification and determine whether to reference or embed productions associated with characters
- Rok to ask Markus on how Cf characters are used and whether we can get away with removing them when processing XML literals.

11.2.2 The XML Constructor

Waldemar states that the test in step 2 of the algorithm doesn't do what is expected because primitive values don't have [[Class]] properties.

John thinks he uses this idiom elsewhere. He agrees that he needs to test for the primitive types and will review all algorithms to do this test.

Rok wonders why case for W3C DOM Element exists and whether it is required. Rok doesn't understand how we'll define what a W3C DOM Element is.

John states it's because he wants to capture the expectation that most implementations will allow XML to be imported from DOM object models. John agrees that W3C DOM Element semantics will be difficult to embed in the spec and he will move it to a non-normative descriptive appendix.

Rok asks if XML objects are considered primitives, prototype based objects, or something special like to host objects in E3.

After some discussion, the working group agrees that XML types are prototype based objects with [[prototype]] property set to NULL. In addition, working groups agree that XML objects cannot be used as part of a prototype chain as corresponding prototype lookup semantics would be too weird given XML ordering and insertion semantics. The constructors for XML types are built-in function objects, but do not have a prototype member.

Waldemar asks what John means by deep copy in step 4. He notes that in the typical definition of deep copies typically copy all references and for the case of XML types, would include a copy of the parent. This is clearly not what is intended.

John agrees that the semantics for deep copy need to be clearly defined and will do write this up in the formal algorithm.

Action Items

- John to modify algorithms as necessary to test for primitive values rather than use [[Class]] internal property.
- John to move W3C DOM Element references to non-normative section.
- John to add algorithm that defines deep copy.

11.2.3.1 settings

Waldemar asks how the different settings are related. As an example, when ignoreWhitespace is set, and prettyPrint is enabled?

John states that values in settings are only read when XML is being parsed or printed. They are not intended to affect any operations which only rely on manipulating the data model. In the example given by Waldemar, on parsing of the XML, the whitespace would be preserved into the data model. On printing the XML value, the indent spaces would be printed followed by any white space in the data model – i.e. the whitespace stored in the in the data model is not stripped to get the output prettier.

Rok and Waldemar are concerned about the use of globals to represent this state and relate the problems that were encountered with regular expressions. No better suggestions at this point.

Rok wonders why settings are stored as an XML instance. He notes that especially for the boolean and numeric settings stored, XML is not a good representation because of the coercion problems.

John states the settings were written by someone else and wasn't sure what the motivation for the choice was. He will look investigate.

Waldemar suggests that settings be directly on XML constructor object.

Working group agrees to discuss once John gets rationale for storing settings as an XML object.

Action Items

- Rok and Waldemar to consider if there is a more appropriate place to store settings
- John to determine why proposal for settings stored values in XML object

11.2.4 XML Built-in Methods

Rok states that because XML objects have null for [[prototype]] property, E3 mechanism for defining methods on the XML instances will not work. This section should be augmented to describe how built-in methods are integrated into the object model.

Action Items

- John to define how built in object model is called on XML data types.

11.2.4.1 appendChild

Waldemar states that the phrasing of calls to [[Put]] aren't consistent with E3.

After reviewing E3 doc, John agrees and will reword instances of calls to [[Get]] and [[Put]] to match E3.

Action Items

- John to update calls to [[Get]] and [[Put]] in algorithms to match E3.

11.2.4.10 domNode ( ), 11.2.4.11 domNodeList ( )

Consistent with discussions for 11.2.2 The XML Constructor on W3C XML DOM, these sections will move to non-normative section.

Action Items

- John to move domNode, domNodeList methods to non-normative appendix

11.2.4.12 insertAfter (child), 11.2.4.13 insertBefore (child)

Waldemar points out an ambiguity with the naming - a.insertAfter(b) can mean insert the value of b in the XML parent of a. or insert the value of a in the XML parent of b.

Rok and John both thought it was the former which is how the algorithm was specified. Waldemar believed it was the latter.

Waldemar proposes these methods be changed to parent.insertChildAfter(anchor, child) and parent.insertChildBefore(anchor, child). This expression of the API has the advantage of having a easy way to specify insert at the head or tail (via the use of null as a special value).

Working group agrees to make this change.

Action Items

- John to rewrite insertion APIs to be on parent node.

Resolving Captured Issues

Issue #2: Parent pointer will be kept

Issue #17: Resolved with descendant method

Issue #19: Resolved with child method

Issue #1: Modeling XML attributes as objects or named, string valued properties

Rok is okay modelling XML attributes as objects. This allows some symmetry between primitive elements and attributes, makes for (i in x.@*) useful.

Waldemar doesn't think this storing attributes as objects is necessary and will be an performance issue.

Issue #9: Comparing XML Strings

Working group agrees data model will not specify ordering of attributes. In addition, an toCanonicalString method will be added that will have the property that trees with identical data models will have identical string representations, and strings for two different XML trees will only capture necessary differences – i.e. if XML element gets a new attribute added, only the attribute shoes up – no unnecessary changes such as reordering of existing attributes occurs.

Action Items

- Rok to define algorithm for to Canonical String