# TC39 TG1 E4X Meeting Notes 2003-05-23

| | |
|---|---|
| **Date** | May 23, 2003 10:00 AM – 12:00 PM |
| **Location** | Conference call<br>888-659-6004 access code 361330 |
| **Convener** | Rok Yu (Microsoft) |
| **Editor** | John Scheider (BEA/AgileDelta) |
| **Participants** | Jeff Dyer (Macromedia)<br>John Schneider (BEA/AgileDelta)<br>Rok Yu (Microsoft)<br>Waldemar Horwat (Netscape) |

## Agenda

- [05 mins] Gather and get started

Jeff Dyer now representing Macromedia [Flash]. Edwin Smith deals with the server side and may occasionally participate.

- [05 mins] Review and revise agenda

No new items. Agenda was adopted as written.

## Review Remainder of Sections 11.2 (11.2.4.22 through 11.2.4.28)

- [20 mins] Review sections 11.2.4.22 prependChild ( child ) through 11.2.4.28 xpath ( XPathExpression )

### 11.2.4.22 prependChild ( child ) / setInnerXML ( childList )

Rok asks whether prependChild is supposed to only support type XML. The confusion exists because the title for prependChild has child for the argument name while setInnerXML has childList for the argument name.

John states that prependChild is supposed to allow XML lists and that the semantics are given by [[Insert]] algorithm. He agrees that having child in some cases and childList in others causes confusion and will resolve the problem by renaming both child and childList arguments to the more generic term value.

Waldemar ask which internal method is used to do a parsing coercion to XML and which is used to do a non-parsing coercion. For the string “<foo></foo>”, the former would return an XML element with name foo, the latter would return an XML text node with content “<foo></foo>”.

John states that semantics don’t currently address this distinction.

Waldemar suggests that as a general rule, setting the value of a text node to the value read out of it should work.

John notes that the [[Put]] algorithm should not be doing a parsing ToXML coercsion, as in general, both parsing and non-parsing behaviors need to be supported.

#### Action Items

- John to rework algorithms to address distinction between parsing ToXML and non-parsing ToXML cases.

### 11.2,.4.27 validate() and 11.2.4.28 xpath ()

John asks whether schema and XPath related sections should be moved to appendix where W3C DOM node sections have been moved.

The working group agrees to this proposal.

John asks whether we can have normative references to W3C recommendations. No one is certain.

Jeff asks how big XPath 1.0 is and if it is possible to embed the semantics into E4X.

John states that it is too big – especially considering the updated version of XPath are absorbing a lot of the functionality found in XQuery.

Waldemar asks what values XPath can return, what global state XPath has, if XPath can have side effects, and if closures can be passed into XPath expressions.

John's  not absolutely sure but believes the answers are as follows: XPath returns W3C DOM nodes; XPath has no global state, XPath has no side effects, and there is no provision for passing closures into XPath.

**Action Items**

- Rok will find out if it is acceptable for ECMA/ISO standards to have normative references to W3C documents.
- John to confirm answers to Waldemar's questions regarding XPath – what can it return, does it have global state, can it have side effects, is there a provision for passing in a closure?

## Review of Sections 11.3

- [60 mins] Review section 11.3 XMLList Objects

John notes that many of the methods on XMLList exist so that XML lists of length 1 behave just like an XML.

The working group completed reviewing all sections of 11.3.

### 11.3.1 The XMLList

Waldemar states that as a general rule, in E4 when a constructor is called as a function and passed an argument, it preserves the identity of the argument if possible – i.e. it doesn't make a copy. The current specification states the function call behavior is identical to the construct behavior and always returns a new instance. He notes that E3 isn't consistent this way, but suggests that E4X should.

The working group agrees to this proposed behavior.

Waldemar asks what happens when null is passed to XMLList. Does it return a list with a single element that has the null value, or does it return the empty list.

John states that the current semantics has XMLList(null) return a list of with the single element null.

Rok notes that if we want to make list == null true for the case where list is an empty list, then we probably want to make new XMLList(null) return empty list.

Waldemar states that the general issue is whether XMLList should be considered a primitive type or a reference type.

John records this issue with the issue that captures if an empty XMLList == null and/or an empty XMLList == undefined.

Waldemar asks if there is a mechanism to convert from arrays to XMLList and what happens when an array object is passed into the XMLList constructor. A more general question is whether an XMLList can store non-XML data.

John thinks the conversions from Array to XMLList can be punted from this version. He adds that XMLList can contain primitives or data of type XML.

Rok notes that with easy conversion functions between Array and XMLList will reduce the need to duplicate methods found on array on XMLList.

John asks if the proposal is for toXMlList and to toArray.

Rok and Waldemar state that using Array and XMLList as the conversion functions seems most straightforward.

John and Waldemar not that the overview for [[Append]] doesn't match what algorithm does.

**Action Items**

- John to add algorithm for XMLList called as a function which preserves identity when passed an XMLList.
- Working group to discuss XMLList for value == null case.
- Working group to consider and decide on coercion functions between Array and XMLList.
- John to review [[Append]] function and reconcile prose with algorithm.

### 11.3.2 The XMLList Constructor

Waldemar asks if the XMLList constructor should support a variable number of arguments like the Array constructor.

John doesn't think it's critical to add this and believes we should punt this due to lack of time.

Waldemar asks how user would create an XMLList of primitives.

John states that if we support between Array and XMLList, this is relatively easy.

Waldemar agrees that if we support the coercion, we can punt vararg.

### 11.3.3.1 appendChild ( )

Rok asks if there is a general rule for deciding what an XMLList method does when it is called on a list of length 0 or greater than 1. i.e. Does it throw an exception or does it iterate and call function on all elements?

John states that the general rule is that anything that does updates will only work on single item. Otherwise, it does the map behavior.

### 11.3.3.5 childIndex ( )

Rok notes that the resulting list will only have elements for items in the source list that are of type XML. He states that with this behavior, it is hard for the scripter to map from the source list to the destination list. He suggests that the list should be the same length as source list with the special

value NaN the result for primitive values. He proposes that as a general rule if method returns an list of primitives, the target list should be the same length as source list and have a one to one mapping.

The working group agrees with this guideline.

**Action Items**

- John will review the algorithms to see apply the guideline that the target maps one to one with the source list.

### 11.3.3.14 isComment ( ), 11.3.3.15 isProcessingInstruction ( [ name ] ), etc.

Rok asks why these methods don't iterate over the elements and return a list like childIndex().

Waldemar states that if the methods always returned an XMLList, they wouldn't be useful for the common case of a single element since an XMLList is always true when coerced to a Boolean. Waldemar suggests that it may make sense to have an XMLList that only stores objects of type XML, and that methods that return primitives always return a single primitive value.

John says he can see the logic to this, but needs to think more carefully about the ramifications.

**Action Items**

- Working group members to consider design pattern for calling methods on type XML list that return primitives.

### 11.3.3.17 name ( )

Waldemar notes that this has the same problem as chilIndex().

**Action Items**

- Working group will revisit once general design pattern is determined.

### 11.3.3.20 parent()

Waldemar notes that the common use case if(a.parent() == null) doesn't work. This is another instance of the same problem.

**Action Items**

- Working group will revisit once general design pattern is determined.

## Issue Resolution

- [30 mins] Discuss the following items from the issues list

06 | Need for a generalized List type

Given the discussion regarding XMLList and the set of possible values that may be stored in them, it is agreed that there is no need to for a generalized list type.

10 | Using "+" for XMLList concatenation

When there is no XML, strings
Building up XML lists piece by piece is a big one.

Waldemar: TODO '==' and other relation operator behavior. It effects what '+' should do.
John: Current expectation, will define behavior when both sides are XML, convert to string.
Waldemar: No specific concern, but want to do this.

Issue closed: We will go for this, if issue comes up , we'll deal with it.

Fullly replace E3 algorithm for Additive operator – current specification isn't clear where intersection with existing algorithm.

05 | Do XMLLists store lvalues or rvalues?
14 | Identifier grammar (status check)

These issues were not discussed due to lack of time.