E4X Meeting Notes 2003-03-27

Date	March 27, 2003 1:00 PM – 5:00 PM March 28, 2003 9:00 AM – 12:30 PM
Location	Netscape Communications Corp. Miami Vice Conference Room Building 21, 1st Floor 466 Ellis Street Mountain View, CA 94043
Convener	Rok Yu (Microsoft)
Editor	John Scheider (BEA/AgileDelta)
Participants	Jeff Dyer (Compiler Company) John Schneider (BEA/AgileDelta) Michael Shenfield (RIM) Rok Yu (Microsoft) Waldemar Horwat (Netscape)

Agenda

The agenda was adopted as written.

New Convener

The working group thanks Peter Torr for his outstanding service as convener of TG1 and welcomes Rok Yu into his new position.

Next Meetings

Date	Host	Comments
April 24, 2003	BEA/AgileDelta	Conference call 10:00 AM – 12:00 PM PDT. John to send out draft for discussion by April 17th.
May 9, 2003	Microsoft	
May 29, 2003		Conference call
June 13, 2003	Netscape	

Schedule

Schedule assumes 6 weeks between meetings which means that there is 4 meetings before spec must be complete. John wishes to add 1 or 2 additional face to face meetings. Working group agrees to this proposal.

We're behind by about one meeting. Working group agrees to have phone conferences between face to face meetings to make up time. In addition, we agree to try and resolve more issues via the email reflector.

Use Cases

Rok was going to review XQuery use cases and port to E4X. Working group agrees this is a nice to have, but not critical, and thus we will not plan to do this.

Onno Elzinga (ECMA Senior Technology Officer) has notified John of a possible synergy between TG1 and a specification on voice control occurring in another TC. Porting their examples to E4X may be good use case examples.

Jeff's use case is still in progress. He will present it next meeting.

Action Items

- John to find spec and example code from voice control specification and send to reflector.
- Rok to port voice control example to E4X solution
- Jeff to prepare use case for presentation at next meeting

Review of Spec Changes

We agreed that the XML type will also include text nodes (as opposed to strings being embedded directly inside XML trees). However, the implications of this have not been fully explored yet.

6.1.1.2 [[Put]] (P, V)

Based on discussions at the previous meeting, the algorithm was modified so that code does not throw exception when passed an invalid XML identifier. It was unclear what the rationale was as we did not have notes from previous meeting.

We reopened the issue and came to agreement that we will throw errors if the identifier to [[Put]] is not valid.

A method will be included to test if an identifier is a valid XML element or attribute name.

There was some thought that the Validate method was meant to be a method for ensuring well-formedness, but after agreeing to add the requirement that identifiers must have the proper characters, the working group could not come up with any examples to get ill-formed xml parsed into the data model. Hence, the need for method to ensure well-formedness no longer exists.

In addition, the working group believes that schema validation will be a commonly requested operation and we want to ensure a common API to do this. However, there are some environments such as on mobile devices where a trusted partner can do validation, and thus, validation functionality is not critical. We agree that support for the validation method will be optional.

Action Items

- John to update algorithm to throw errors for invalid property name.
- John to add method to validate identifier and do validation against a schema.

6.1.1.5 [[Replace]] (P, V)

Waldemar notes that for V values that are text, the value stored is a string. It needs to be converted into an XML node of type text.

Action Items

- John to review Replace algorithm and modify to make sure string values are stored as XML nodes of type text.

7.1.1 ToString Applied to the XML Type

Working group agrees that for the elements with simple content (i.e. where an element contains only child text nodes), ToString returns the the content of the element (i.e. the result of concatenating the strings stored in the text nodes). When applied to elements with simple content, ToString does not replace XML special characters with entities. ToXMLString does.

There is disagreement as to what ToString does when applied to other XML element instances withcomplex content (i.e. elements that contain other elements).

When ToString is applied to an XML element with complex content, John wants ToString to return a string representing the entire element, including its start tag, end tag, attributes, and children. He believes it is counter-intuitive to omit the start tag, attributes, and end tag for this case. i.e. ToString effectively returns innerXML for simple content, and outerXML for complext content. John believes this behavior satisfies the most common cases and yields the most intuitive results. In other cases, where the user always wants outerXML, the user may use ToXmlString(). In cases where the user always innerXML, they can call innerXML().ToXMLString() [or use "*" – the shorthand notation for innerXML()].

Michael, Jeff, and Waldemar believe consistency is more important and that ToString should always return innerXML. The alternative of modifying the print routine to call ToXMLString for XML values addresses some of John's concerns, but introduces a difference in behavior between using the print routine and other scenarios where XML is converted to string.

Rok leans towards Michael and Waldemar's point of view, but can see both sides.

There is agreement that attributes do not affect whether innerXML or outerXML is returned.

Examples:
x = <x>4</x>
y = <y id="number">5</y>
z = <z><a>3</z>

print(x)
→ 4
print(y)
→ 5
print(z)
→ <z><a>3</z>

No resolution was achieved on the desired behavior of ToString on XML elements with complex content. As a last resort, members agree we could keep ToString unspecified in spec for non-XML elements with complex content.

Action items

- Rok will get opinions from others at Microsoft on behavior preference and write up his position.

7.1.2 ToString Applied to the XMLList Type

Algorithm was modified to return a comma separated list. Working group agrees to this but notes that behavior is tied to 7.1.1 which is still being debated.

7.3 ToXML

Based on discussions at the previous meeting, the document was modified so ToXML returns undefined when an attempt to convert an XMLList fails. It was unclear why this was. The working group agreed to revert change to throw TypeException. This behavior is consistent with other type conversions that also fail with exception.

Action Items

- Rok will develop a proposal for ToXML for objects.

Review of New Sections

Sections 5 and 9 were added. Working group did not have enough time to review items 9.5 and later.

5.1 Identifiers

Waldemar question as to whether the Identifier token is the appropriate place to extend Edition 3 to add identifiers needed for XML lookup. He is concerned that grammar as it stands may be ambiguous. Waldemar thinks a better place for supporting the identifiers is off of the grammar section for the dot operator.

John's approach has been to use a more permissive grammar and to rely on semantics to disallow the new XML identifiers in inappropriate contexts. He thinks Identifier is an appropriate place to extend the grammar because he wishes to support unqualified references in filter predictates. i.e. foo.(@bar == mybar).

Waldemar and Rok indicate that unqualified references will have the same problem as the current with statement where the lexical scope captures identifiers from dynamic data. This is a situation both would like to avoid.

For now, we agree to leave grammar as is for now record an issue to revisit grammar once filtering predicate design has been agreed to.

Action Items

- Waldemar will investigate grammar and come back with proposal for a more specific grammar definition.

5.2 Puntuators

Waldemar notes .(is not needed as a punctuator and can be handled in grammar.

Action Items

- John to remove .(from punctuator list

9.1.1 XML Initializer

Waldemar notes that a lexing context is needed to resolve an ambiguity problem that exists in current XML initializer definition. A similar problem exists in the regular expression literal definition and the same technique can be used to resolve it.

```
x = a< b+c>d,"</y>"
y = a/b+c/d
```

Also noted is that XML literals will have the same funky interaction as regular expressions with semi-colon insertion.

```
a = b
/c+d/g.exec("Hello")
```

Several bugs also noted.

- Grammar does not allow literals with mixed content – i.e. <a><b/>hello
- PI, Comment, CDATA need to be rewritten to disallow terminating token sequence from content.

Action Items

- John to fix bugs and make changes so XML literals use technique similar to regular expression syntax to fix disambiguates.

9.1.2 XMLList Initializer

Rok notes that in almost all cases, an XMLList with one XML element behaves identically to an XML instance. He suggests that only XMLList initializers are necessary, and suggests that current delimeters <> and </> are too inconvenient for the common single XML element case.

Action Items

- Rok to work on proposal that unifies XML and XMLList initializeers

9.2.2 XML Descendent Accessor

Waldemar believes that in addition to MemberExpression, CallExpression needs to be modified to support ".." descendant operator.

Waldemar notes that there is currently no way to execute descendent ".." behavior for a runtime property name.

Action Items

- John will review and add necessary grammar to CallExpression.
- John to explore adding descendent lookup with runtime property name via a function.

9.2.3 XML Filtering Predicate Operator

Current proposal has same problem as the with statement. A race condition exists between members of the MemberExpression XML instance and the program's variables.

Rok suggests using "." to represent the context node. Using 'this' is undesirable because it is ambiguous. i.e.

```
x.(.price > balance)
x.(this.price > balance)
```

Waldemar notes that as with descendent operator, there needs to be some way to get at properties with computed names i.e. [].

Action Items

- Waldemar to check the grammar to see if "." is possible.
- John to explore options for getting at computed properties.

9.4 Additive Operators

Waldemar notes that since text is stored as XML nodes, concatenation will result in an XML list – not a string. Working group thinks behavior appears reasonable but has postponed finalizing what the behavior should be until the implications can be thought through.

Rok is concerned that plus operation on numeric values will be a very common case and people will be confused when plus concatenates the text nodes into a list. At minimum, we need to prescribe how to code up the intent to add XML values that are numeric.

Working group agrees that rules we come up with for the + operator should depend only on the types of the operands and not on their values.

Action Item

- John to add example for case of coercion to number and all text is stored as strings.
- All members to consider implications of having plus operator concatenate and produce a list when adding text nodes.

Discuss Outstanding Issues

Working group did not have enough time to handle this item.

Next Steps (Statements & Built-ins)

Working group did not have enough time to handle this item.