TC39-TG1 E4 Meeting Notes 2003-07-01

Date	July 1, 2003 10:00 AM – 12:00 PM
Location	Microsoft Corporation
	Building 113/3001
	One Microsoft Way
	Redmond, WA 98052
	Conference call
	866-500-6738 access code 795181
Convener	Rok Yu (Microsoft)
Editor	Waldemar Horwat (Netscape)
Participants	Jeff Dyer (Macromedia)
	John Schneider (BEA/AgileDelta)
	Rok Yu (Microsoft)
	Waldemar Horwat (Netscape)
	Markus Scherer (IBM)

Meeting Logistics

The working group agrees to start the July 2nd meeting at 9:00 AM.

Rok states that the March TC39 meeting is in Melbourne and asks if working group should schedule an adjacent TG1 meeting.

The working members would like to keep the option of calling into TC39 meeting rather than physically attending. The working group understands that members should physically attend if the group is plans to present a specification for ratification, and postpones making a final decision until the July 25th conference call where the decision will be made on the set of specifications that will present at the September TC39 meeting.

Prioritizing Between E4 and E4X

Rok states that it is clear that E4 will not make the September deadline and that E4X is at significant risk to meet the deadline. He proposes that the working group prioritize E4X work over E4 work until the September meeting.

Waldemar would like to continue to make progress on the E4 work.

The working group agrees to that future meetings will start with the E4X agenda and be goal driven rather than time driven. The E4 agenda will not start until E4X agenda is completed. Furthermore, until the reference implementation is completed, rather than sequentially reviewing algorithms, the E4 agenda will consist of:

- Resolving open design issues
- Reviewing sections thought to be exceptionally risky or that overlap with E4X
- Address questions and reported bugs

The working group agrees to end the E4 meeting early and start on the E4X agenda in the afternoon.

Design Issues and Review Items

Waldemar list the following items for addition to the issues list.

(42) How should IndexRead and IndexWrite be defined?

(43) Review use of HasProperty to make a string behave as an array of characters.

(44) Which implicit coercions should be supported?

(45) What precision should string to number conversions have? Which exceptions should be thrown?

(46) How should ambiguities and inconsistencies found in E3 object model be dealt with?

(47) How should object model handle unexpected input?

(48) What are the semantics for characters?

(49) What are the consequences of introducing numeric types (float, int, etc) to the object model?

(50) Review variable declarations, function definitions, and signature matching

(51) Review property look up algorithm

Reference Implementation and Test Harness

Rok states that in order for Microsoft to run test cases, the reference implementation needs:

- (52) A mechanism to load and run ECMAScript programs from disk
- (53) A built-in object model to log results from test case runs

Rok thinks it will take about three weeks of iteration to get test harness and major blocking issues resolved. He would like to start this before algorithms are complete so any obvious problems can be understood early.

Waldemar agrees to add the necessary support for the next meeting.

Rok asks if there are any tips for debugging the reference implementation.

Jeff mentions that commercially available Allegro Lisp has a trial version and will probably have good debugging facilities.

Waldemar states that there are methods to pretty-print the internal data structures.

Issues List

Rok has taken a pass through the issues list and updated it to match current state of document. He did not attempt to assess current state of work items 14 through 26 which track the implementation of the object model.

(5) Add \U syntax in strings and identifiers

<<uXXXX>> syntax is used for BMP characters. Waldemar added <<UXXXXXXXX>> syntax to represent supplementary characters.

Waldemar believes he can implement the changes in the lexer to support supplementary characters. He asks that Markus help him use the correct Unicode terminology to describe character classes.

Markus agrees to help Waldemar.

(9) Date/Time formatting use cases

Markus states that this work item isn't a high priority and can be postponed to a future version.

(42) How should IndexRead be defined?

Waldemar states that E3 array algorithms are inconsistent as to whether the prototype chain is looked at when looking up array indices. As an example, shift looks at the prototype chain for index 0, but none of the others. He notes that many of these methods are defined to be generic and are applicable to non-arrays.

Waldemar believes that the cases where array elements are stored in the prototype chain are rare and recommends the standard allow implementations to choose whether they walk the prototype chain when resolving indicies.

The working group agrees that this is an edge case and that leaving the behavior as implementation defined is acceptable.

John notes that the way the algorithms are currently specified, the standard implies that the implementations will consistently choose whether index lookup will look at the prototype chain. The desired semantics is that the behavior is consistent for any specific call site.

Waldemar will (54) update the prose to make this clearer.

Waldemar notes the following about the implementation:

- Calls to get array elements and the length property are vectored through indexRead and readLength so they will work on non-array objects.

- In E4, the Get method will return the value for a property or the token none if the property doesn't exist. In E3, a separate method [[HasProperty]] exists to check if a property doesn't exist.

Rok notes that in the writeArray method, (55) the range check on the index needs to happen before the call to ordinaryWrite.

(43) Review use of HasProperty to make a string behave as an array of characters.

The working group has no issues with this implementation choice.

(44) Which implicit coercions should be supported?

Waldemar states that implicit coercions are used in four cases:

- assigning to a typed variable
- passing in a parameter to a typed argument
- returning from a function with typed return value
- the "as" operator

Waldemar has chosen to be conservative with introducing implicit coercions. So far, the only implicit coercions arefrom Number to the integer types.

Waldemar asks if implicit coercions should strings of length 1 and characters.

The working group agrees (56) that these should be added.

Jeff asks whether (57) user defined implicit coercions should be supported on classes.

The working group agrees that this is not critical for E4 and chooses postpone this feature until a future version.

Jeff asks whether the standard should define implicit coercions from various types to string to impove the "scriptness" of the language.

The working group decides against standardizing this because the working group believes that users of type annotations will want stricter type checking. In addition, leaving coercions out does not prevent implementations from adding addition coercions.

(45) Which exceptions should be thrown during number conversions? What precision should be preserved?

Rok asks why float32 be distinct from float64 and if it should be removed?

Waldemar states the main reason was because there was original float32 arithmetic that was distinct from float64 arithmetic. This is no longer the case and Waldemar agrees that (58) float32 should modeled in the same way as the integral types.

The current algorithm to convert from String to Number considers all digits of the string as significant. Edition 3 allows implementations to round the last 20th digit. Rok believes this wording was specifically added so implementations didn't have to resort to arbitrary precision arithmetic. (58) Rok will confirm this is the case.

Waldemar has chosen to use RangeError rather than TypeError when coercions between various number types fail.

The working group is concerned that this will confuse users as coercing 64.5 to int would return a range error even though it falls between the minimum and maximum values of int. The working group accepts this choice with the rationale that RangeError's are acceptable here because all the number types are subtypes of GeneralNumber, and coercion doesn't change the type of the value but instead applies constraints on it.

Markus asks (60) whether coercion routines that take a locale should be included.

The working group agrees that the work to define a locale is significant and decides to postpone this for consideration in a future version.

(46) How should ambiguities and inconsistencies found in E3 object model be dealt with?

Waldemar proposes that when ambiguities exist in E3 behavior, he will raise the issue and either E4 will behave compatible to E3 or for questionable semantics, make the cases errors.

The working group accepts this proposal.

(47) How should object model handle unexpected input?

To keep things simple, Waldemar proposes to make missing arguments behave the same as if an undefined value is passed in. It will be an error to supply too many arguments.

The working group accepts this proposal.