ECMA International

ECMA/TC39-TG1/2003/15
ECMA/TC39/2003/26

Standardizing Information and Communication Systems

TC39-TG1 E4 Meeting Notes 2003-06-05

Date	June 5, 2003 10:00 AM – 5:00 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	Waldemar Horwat (Netscape)
Participants	Jeff Dyer (Macromedia) John Schneider (BEA/AgileDelta) Rok Yu (Microsoft) Waldemar Horwat (Netscape) Markus Scherer (IBM)

1 Agenda

The agenda was adopted as written.

2 Document Status

Rok will begin tracking all issues and action items on E4 issues spreadsheet. For items tracked by the spreadsheet, notes will include corresponding issue number in parentheses.

The following is the list of work items that remain to complete algorithms.

- (12) Lexer support for supplementary character/supplementary character classes
- (14) Global object methods
- (15) Method GeneralNumber.toExponential
- (16) Method GeneralNumber.toPrecision
- (17) Math object methods
- (18) RegExp methods on characters and strings
- (19) Array methods
- (20) Attribute methods
- (21) Date methods
- (22) RegExp class
- (23) Class methods
- (24) Function methods
- (25) Cleanup package definitions
- (26) Error classes

3 Design Review of Modifications

- Waldemar has completed the following updates to the document.
- (5) Added \U escapes to the lexical grammar (but not yet the semantics).
- (10) Put back a very simple package definition/import mechanism. Moved the extended import directive with the include/exclude selection mechanism to the rationale.
- (11) Removed include and exclude keywords.
- Split the super semantic field into super for classes and archetype for prototype-based objects.
- Implemented simple prototypes on most objects, including class objects themselves, for compatibility with JavaScript 1.5.
- Changed semantics so that the enumerable attribute now inherits when overriding an instance property. This was the original intent, although it was ambiguous in the written description of the attribute.
- Added semantics for the library classes Object, Void, Null, Boolean, GeneralNumber (except for number formatting), Number, float, sbyte, byte, short, ushort, int, uint, long, ulong, Character, String (except for regular expressions), and Namespace.
- Added numerous semantic utility functions to support the above classes.
- Removed some obsolete text from the description of Types; however, that page still needs a little work as it's not as up-to-date as the semantics.
- Namespace implementation completed

Rok asks how short, ushort, int, and uint relate to Number and how built in methods show up on these instances of these types.

Waldemar states that these exist strictly as type constraints the semantics are identical to Number. Coercions from double to these types is checked. The methods sit on GeneralNumber – the prototype of all these values is the Number.prototype object.

Waldemar states that he added an archetype member to non-primitives which is the equivalent to the [[prototype]] property in Edition 3. He introduced a function called archetype which returns the appropriate prototype object for primitives, and the value of the archetype member for non-primitives.

Waldemar asks what String(null) should return. Is String a primitive or a reference type? As currently specified, variables of type String cannot store null.

Rok mentions that the String type in languages like Java and C# behave as a reference type and allow null. Allowing the null value is required to smoothly interoperate with object model defined in other languages

Waldemar asks whether functions should be callable in constant expressions. He is making a case by case decision. Functions are callable only if they clearly have no side effects. Members agree this is okay.

Waldemar asks whether error class methods should be prototype based, or class based. No strong opinion either way. He will do what ever is easiest.

Action Items

(27) How should implicit and explicit coersions of the null value to the String type work?

4 Review Progress and Check on Schedule

June 5th meeting was checkpoint to be algorithm complete. We're still a bit away from there. Given this information, Rok asks how algorithm reviews are going and whether it makes sense to officially slip schedule now.

John states that algorithm reviews are going slow. Currently he is the only one at BEA that is looking at the algorithms. He believes it is most efficient for his resources if he waits until prose is ready before passing algorithms to others for review.

Jeff is starting to ramp up on the reviews.

Rok is currently the only one at Microsoft reviewing algorithms. The rate of reviews has started off slow, but is improving as understanding improves.

Everyone agrees that it is likely we won't hit the September deadline. However, the working group will continue with the current plan of record and make the final decision on which specifications to submit at the July 25th meeting.

Waldemar asks whether completing algorithms is still the priority given that general review of algorithms by a wider audience is blocked by lack of prose. He wants people to start reviewing algorithms now because he believes that once he will be able to complete prose quicker than people will be able to review prose and algorithms.

John states that running the reference implementation and doing black box testing on it will be much more effective at finding issues than reviewing the code. He continues to believe that getting algorithms completed first is the correct prioritization.

Rok agrees and states that it is important for Microsoft to run test cases against reference implementation to get a good understanding of behavior changes from E3. He adds that a reasonable plan is for members to review algorithms in core semantics and that a good use of secondary resources is to review the object model algorithms.

Working group agrees to continue with current plan of completing algorithms first.

5 Final Review of 11.2 Identifiers through 11.5 Function Expressions

11.2 Identifiers

Result: Working group accepts section.

Rok asks (28) where semantics for Name[Identifier] is defined for Identifier token. Section 7.5 Keywords and Identifiers which introduces the tokens uses terminology like "return an Identifier with the name id", but doesn't connect the concepts.

Waldemar has added section 11.1 Terminal Actions and will define the behaviors in the prose for this section.

Rok asks if (29) Setup[SimpleQualifiedIdentifiers]() could be nuked. Propagation of call could be pruned at the QualifiedIdentifier => SimpleQualifiedIdentifier level.

Waldemar states it could be, but this is a minor local issue and semantically correct, so we won't change it.

Action Items

- (30) Make sure prose for 11.1 Terminal Actions defines what Name[Identifier] means for Identifier token.

11.3 Qualified Identifiers

Result: Working group accepts section minus minor change for action item (32).

Rok asks if (31) Eval[QualifiedIdentifier] should use the "propagates call to every nonterminal expression" short form. The current text appears to do this explicitly and is harder to read because it requires reader to carefully look at algorithms to verify that they indeed to just propagate the call.

Waldemar states that the current processor does not use the propagate phrase if there is exactly one production, or if the semantics return a value. He states it will be easy to change the former to start using the propagate phrase. He thinks it will be harder for the case where semantic actions returns a value, but he believes there are very few cases where this occurs.

Working group agrees that using the short form for the former case and punt on the latter case. Change will be made throughout algorithms.

Action Items

- (32) Modify algorithms to use "propagates call to every nonterminal expression" where possible.

11.4 Primary Expressions

Result: Working group accepts section allowing for additional semantics to be added when RegExp is defined.

Action Items

- (33) Re-review 11.4 Primary Expressions when RegExp is added.

11.5 Function Expressions

Result: Working group accepts section.

Rok asks if (34) Function expressions w/ Identifier should introduce a new scope. He thinks that in Edition 3, the identifier modifies the variable object.

Waldemar states that in Edition 3, Function expressions do introduce a new scope for the Identifier. Function declarations modify the variable object. Review of Edition 3 documentation confirms this.

6 Review of Unicode Open Issues

(1) Ensure conformance section states Unicode 3.0 or later

This is prose and will be done after algorithms are complete.

(2) Add 0x0085 to be added to "7.3 Line Breaks"

Not yet completed. Waldemar will add to draft for the July 1st meeting.

(3) Rewrite based on Unicode White_Space property.

This is prose and will be done after algorithms are complete.

(4) Rewrite based on ID_Start, Other_ID_Start character classes.

This is prose and will be done after algorithms are complete.

(5) Add \U syntax in strings and identifiers

The lexical grammar has been update to allow these, but the semantics haven't been written. Semantics are blocked on completion of "(12) Modify lexer to support supplementary code points" which Markus will do for July 1st meeting.

(6) Update terminology to match Unicode

This is prose and will be done after algorithms are complete.

(7) stringFromCharCode should allow supplementary code points

Algorithm isn't written yet.

(8) localeCompare should be clear that behavior is implementation dependent. Same apples to toLocaleUpper, toLocaleLower

Completed.

(9) Date/Time formatting use cases

Markus will get information from colleague for June 19th conference call.

(12) Modify lexer to support supplementary code points

Waldemar states that changing the lexer to process code points and having a prepass to convert code units to code points has complications because escape sequences are significant. Given '[' represents the high Unicode surrogate pair code unit \uD800, ']' represents the low surrogate pair code unit \uDF30, there are five combinations to consider for the Gothic letter AHSA \U00010330 in source text (character prefixed by ab, and suffixed by cd).

1. ab\uD800\uDF30cd
2. ab\uD800]cd
3. ab[\uDF30cd
4. ab[]cd
5. ab\U00010330cd

Cases 2 and 3 are not possible because UTF16 disallows unpaired surrogate pair code units.

Markus will work with Waldemar to investigate the changes needed to the lexer offline and will present the proposal at July 1st meeting.

7 Walkthrough of Sections 11.6 Object Literals through 11.14 Bitwise Shift Operators

11.6 Object Literals

No issues.

11.7 Array Literals

Rok asks what private::length is and whether (35) it should be a float64 instead of a long.

Waldemar states private::length is an the private backing store for the public length property. He agrees semantics will be strange if length is a long and will evaluate changing it to float64.

11.8 Super Expressions

Rok states that the super is used in Microsoft's implementation solely for calling the super constructor. He asks if (36) the behavior of super(<expr>) is inconsistent with this.

Waldemar states that calls to super class constructor are handled by 12.3 Super Statement. The two cases are distinguished in the grammar because the super expression must be followed by a Property operator.

11.9 Postfix Expressions

Rok asks if (35) Postfix operations should be applicable to post fix expressions. In Edition 3, expressions such as x++-- were disallowed by the grammar. Current syntax allows this.

Waldemar states this was unintentional. However, the additional allowed syntactic constructs either result in generation of an error in the semantics, or a reasonable definition.

Working group agrees that the benefits of attempting to refactor the grammar at this point is not worth the risk.

Rok asks if the intent of AttributeExpressions is to capture what is possible as part of an attribute.

Waldemar states that they are and define syntax for all attributes minus keywords.

11.10 Property Operators

No issues.

11.11 Unary Operators

Waldemar notes the special treatment for NegatedMinLong which allows -2^63 to be represented as a long.

11.12 Multiplicative Operators

On review of rationalToLong defined in section 10.1 Numeric Utilities, Rok asks where 0.5 comes from in the lower bound range check q < -2^63 – 0.5 to decide if the result should be a double as there might be a closer long value.

Waldemar states that this is because the else clause which returns the long value relies on this as it picks the integer closest to q which must be in the range of the long.

11.13 Additive Operators

No issues.

11.14 Bitwise Shift Operators

No issues.