ECMA/TC39/98/5

ECMA comments ISO/IEC DIS 16262, ECMAScript

under ballot until 1998-04-09

1)The second paragraph of clause 2 on conformance need to be improved.

A proposal for such an improvement is the following:

"A conforming implementation of this International standard shall interpret characters in conformance with The Unicode Standard, Version 2.0, and ISO/IEC 10646-1 with UCS-2 as the adopted encoding form, implementation level 3. If the adopted ISO/IEC 10646-1 subset is not otherwise specified, it is presumed to be the BMP subset, collection 300."

Background information

The above proposed text *does talk* about implementation level 3 as described in 10646, but it does not mean that ECMAScript would have to process a character and a combining sequence as yet another uniquely identified character. ECMAScript can treat a combining sequence as just another 16-bit value. Combining sequences are encoded for use with some base characters especially for Indic, Thai, Arabic and Hebrew scripts. Platforms and application software decides how to process a base character and a combining sequence.

Note that combining sequences can be used for Latin as well but the vast majority of Latin script characters are already encoded in 10646.

For conformance with The Unicode Standard it is important that combining sequences are not damaged as they go through a process, such as ECMAScript. Unicode supports combining sequences and provides an equivalence algorithm that facilitates comparing precomposed characters with a base character followed by a combining sequence. All that ECMAScript, and other programming languages, need to do is to let a data stream of 16-bit values pass through the process cleanly and with no damage to the data.

2) 7.4.3 Reserved words

The following keywords are reserved in at least one implementation and
should be included as future reserved words:

abstract boolean byte char double final float goto implements
instanceof int interface long native package private protected public
short static synchronized throws transient volatile

3) 7.7.3 Numeric literals

This section does not define precisely what is meant by hexadecimal and octal literals that result in Mathematical Values that are larger than can be accommodated in a IEEE double. Since octal and hexadecimal literals only make sense when used in conjunction with bitwise operators, which operate on unsigned 32 bit integers, it would make sense to limit octal literals and hexadecimal literals to specifying unsigned 32 bit integers.

4) 9.3.1 StrWhiteSpaceChr

This does not handle Unicode strings that use white space characters other than ASCII. The definition of this should be changed to correspond to the definition of isWhiteSpace in the Java class library.

5) 10.1.6 Activation Object, and 15.3 Function Objects

The arguments property of Function instances should be deleted instead of discouraged. The way it is defined now is over-specified and thread-unsafe. It neither makes sense for future implementations (which may implement threads) nor describes existing practice.

6) 11.2 Left-Hand-Side expressions

The grammar should not allow nonsensical expressions such as new new foo(), and should not accept Function calls and new expressions on the left-hand-side of an assignment.

7) 11.4.1 The delete operator

Step 2 will generate a runtime error if Result(1) is not a reference. This does not appear to be the intent, as in Step 4 GetBase always returns an object (if it returns at all), hence this test is redundant and in Step 5, Objects are required to implement the [[Delete]] method, hence this test is redundant.

8) 12.2 Variable statement

Globals explicitly declared with var should be marked DontDelete.

9) 12.2 Variable statement, evaluation of Identifier Initializer

Step 1 is incorrect: Identifier is not a syntax rule, hence it does not make sense to evaluate it. The intent seems to be "Evaluate Identifier as if it appeared in a PrimaryExpression : Identifier production, see section 11.1.2." However, this could lead to strange behavior when variable declarations appear inside with statements. A better formulation is "Construct a value of type Reference whose base object is the next activation object on the scope chain and whose property name is the Identifier." Step 5 should then be: "Return Result(1)."

10) 12.6.3 The for..in statement, evaluation of for(var ...)

Step 7 is not needed, provided that the definition of VariableDeclaration is fixed appropriately, and similarly Step 8 can then use "Result(1)".

11) 15.1 The Global Object

The standard should outlaw calling the global object's eval method indirectly. In particular, programs should not extract the global object's eval property except when calling it directly, or assign to the global object's eval property.

12) 15.2.4.2 Object.prototype.toString()

This conversion could result in a runtime error, which does not seem to be the intent here. If [[class]] is required to be a string, this conversion is not needed.

13) 15.4.2.2 new Array(len)

This constructor constructs a very large array if given an argument such as -1. The language of the new Array(len) constructor should be changed to require that ToInteger(len) be between 0 and 2^31-1, inclusive; if len is a number outside this range, new Array(len) should signal an error. The same applies to setting the "length" property of an array via [[Put]].

14) 15.4.4.5 Array.prototype.sort(comparefn)

Step 3, last paragraph: the phrase "and the result of applying ToNumber to this value is not NaN" is not necessary and creates the erroneous impression that the compare function need not return a number. See earlier "If comparefn is provided, it should be a function that accepts two arguments x and y and returns a negative value if x < y, zero if x

= y, or a positive value if x > y." While "value" is not a specific as
"number", it seems unlikely that the intent was that "negative value"
be shorthand for "a value which can be converted to a negative number".

15) 15.4.5.1 [[Put]](P, V)

Array instances always have a length property, hence the first test in
Step 9 is not needed.

16) 15.5.4 Properties of the String Prototype Object, final paragraph

The phrase "it is a runtime error if this does not refer to an object
for which the value of the internal [[Class]] property is 'String'."
should be deleted. It contradicts the note at the end of section
17) 15.5.4.4 and as well as the implicit intent.

18) 15.9.5.34 setMonth and 15.9.5.34 setUTCMonth

Step 2 of the algorithm should refer to 'mon', not 'date'.