ECMA/TC39/98/10

Disposition of Comments Report for DIS-16262
15 June, 1998

The following report describes the actions that have resulted from comments received for the ISO/IEC Letter ballot for DIS-16262. All comments received before the deadline of 9 April 1998 are included in this report. This document was approved at the ISO/IEC DIS-16262 ballot resolution meeting held 15 June 1998 at Sun Microsystems in Menlo Park, California

ECMA TC-39 appreciates all of the comments that have been received. These comments and the resulting improvements to the draft standard provided great benefit to future users of ECMAScript.

Comments have been received from the following countries:

Denmark (D)

France (F)

Japan (J)

Netherlands (N)

USA (U)

ECMA (E)

All comments received have been merged into a single document. The sequence of the comments has been arranged to match the paragraphs in the DIS. To identify the original comments tags referencing country and paragraph (e.g. D1, D2, D3, etc., U1.1, U1.2, etc.) have been added. Many of the comments that apply to more then one specific section have been grouped together at the beginning of the report in a section titled General Comments.

After each comment the action of the committee is provided. The committee actions are highlighted by the notation: >>>>>> **Action: [committee action] --- [comment on the action]** <<<<<

This report is divided into two major sections:

Comments that apply to across multiple sections -- General Comments

Comments that apply to specific sections

General Comments are divided into four sections: Editorial; Name of the Standard: Year 2000; and Character Related issues. In some cases comments on specific sections are also covered by the comments in the General Comments section. In some of these cases a reference to more general response has been provided.

Comments on specific sections have been arranged by the sections and subsection used in the draft ECMAScript standard. In some cases similar comments have been combined. In these cases the comment tag includes a reference to each of the original comments.

General Comments

Editorial Issues

J-general) --- Minor Technical and Editorial comments:

With ECMA-262, JNB found a lot of minor technical and editorial errors. Therefore, JNB would strongly suggest that a technical corrigenda should be prepared of this standard, and be republished after spell checking of the standard document. The followings are just examples and not the complete list of errors. I'm afraid if a work document was published instead of the final version by accident, since there are so many spell errors.

>>>>> Action: Accepted --- The committee will endeavor to improve future documents. <<<<<

N1) General comment:

It is disappointing that this document contains quite a large number of typos and some misplaced sections. We think that the fast-track procedure has not been intended for such textually immature documents. Careful inspection by the editor would have uncovered many problems. Below some of these problems have been indicated: we are unsure we caught all.

>>>>> Action: Accepted --- The committee will endeavor to improve future documents. <<<<<

U1.50) General comment. Some level-2 and level-3 headings have each word with a leading capital letter and some don't. Make them consistent.

>>>>>> Action: Accepted --- Consistent headings will be used. <<<<<

U1.51) General comment. It would probably be an improvement to set all reserved words, function names, and operators in headings in a constant-width typeface. Having the level-2 heads 12.7 to 12.10 be all caps whereas those in 12.6.1 to 12.6.3 are not, looks strange.

>>>>>> Action: Accepted --- Document corrected to be consistent. <<<<<

U2.1) The term "runtime error", "compile-time error" and "error" are all used, but not defined. This document should define how a "compile time error" is recognized, for example by an implementation defined diagnostic message or return code Same with "runtime error" and error. Without these definitions measuring conformance will be impossible.

>>>>>> Action: Accepted --- The term compile-time error was removed; error alone should be runtime error (there were only 1 or two cases). (Definition of how the error is surfaced is an environment issue, that is, implemenetation-defined) <<<<<

U6.4) Look for all copies of the word "we" and replace with standards wording.

>>>>>> Action: Accepted --- Use of "we" will be removed from the document. <<<<<

Name of the Standard

U5) During the standardization process for ECMA 262, the naming of the standard was dealt with in an unsatisfactory manner with a less than optimal result. The original name put forward to the committee, LiveScript, was agreed on by all members but was withdrawn at the last moment with no alternative proposed. This resulted in a standard named ECMAScript. Because this is not a copyrighted or trademarked label it is unlikely that any implementation of the language will be called ECMAScript. This has and will result in confusion for users as to what the standard means and what language engines support the standard.

We believe that in the case of standards applicable to the mass market (where both the contributors to the standardization effort and the public at large have an expectation of interoperability) names are very important, both as an indicator of the openness of the process where the parties cooperating expect to begin to compete from a common footing at the end of the process, and as an assurance to the public that their expectations are forthrightly met (e.g. codewords and numbers are unacceptable when assuring the public that they can connect an appliance into a wall outlet). Where the community of customers are small or well informed, this is less an issue than in the case of ECMA262.

Also, the lack of a marketable name and ownership (or at least unencumbered use) of a popular name by ECMA (or any standards body that is unable to assert ownership of their efforts) for a highly visible interoperability standard diminishes that body's ability to generate revenues from the publication and ongoing maintenance of the definitive standard. This can be seen in the commercial publication of other "standards" in the same discipline without regard for the hosting organization's need for a sustaining revenue stream. Lack of ownership will bias future efforts towards less formal consensus efforts not dependent on publication revenues and further undercut the traditional standards bodies.

>>>>>> Action: Not accepted --- The committee has not obtained use of the name LiveScript. <<<<<

Year 2000 Issues

J4.1) Two digit representation of year

As discussed in the ECMA-262, Date.prototype.getYear() and setYear() has "year 2000 problem". If the rationale of inclusion of the functionality is only backward compatibility, those methods should be removed form this standard, because

(1) this is the first edition of ECMAScript therefore there is no previous version nor edition from the standard view point.

(2) this standard allows enhancement of properties and methods, therefore implementations of this standard can provide these method as enhancement.

Also, in some methods when the year value between 0 and 99 is specified. the 1900 will be added to the value as a base. This specification may not appropriate to the standard published in 1998 or 1999. The default base should be removed or amended. For example add 1990 if the value is somewhat between 70 and

99, and add 2000 if the value is between 0 and 69.

>>>>>> Action: Partial acceptance --- Please see the Action note at the end of this section on year 2000 issues. <<<<<

U3) Paragraphs 15.9.5.5 Date.prototype.getYear() and 15.9.5.38

Date.prototype.setYear() should be removed because, as noted in ECMA-262 they may contribute to the "Year 2000" problem. The rationale for their inclusion, backwards compatibility (with what, as there are no prior standards), is not sufficient for such a strongly deprecated programming practice.

>>>>>> Action: Partial acceptance --- Please see the Action note at the end of this section on year 2000 issues. <<<<<

U7) The year 2000 date rollover problem is having profound effects on government and industry software systems worldwide. Systems are already failing due to the erroneous processing of dates that include the year 2000. Most software systems are programmed to recognize 2-digit years in dates as occurring within the 20th century, with the implication that the first two digits of the year are 19. When the year 2000 comes into play, these software systems will more than likely recognize 00 as 1900 instead of 2000, causing consternation among users, system managers, and database administrators. Billions of dollars are being spent on fixing the problem.

The U.S. Congress is holding frequent hearings on the progress that Federal agencies have made in repairing systems that support a myriad of government programs. A meeting among representatives of the U.S. Office of Management and Budget (OMB), 27 Federal agencies, and 41 states in October 1997 affirmed the action to require 4-digit years in all dates used in data interchange between the Federal and state governments, and that the Federal government would act as lead in further actions as necessary. Further, the U.S. Securities and Exchange Commission (SEC) is examining requirements for publicly held companies to disclose the extent of date processing problems and plans for correcting these problems within their mandatory filings. Other Federal agencies, most notably the Nuclear Regulatory Commission (NRC), the Federal Aviation Administration (FAA), the Environmental Protection Agency (EPA), the Food and Drug Administration (FDA) and others are developing implementation plans for overseeing the correction of date processing problems within regulated industries. NIST spearheaded the government's position when it issued Change Notice 1 to Federal Information Processing Standard (FIPS) 4-1 in March 1996 which highly recommended the use of 4-digit years and deprecated the use of optional 2-digit years.

The National Committee on Information Technology Standards (NCITS -- formerly X3) Technical Committee L8 has recommended a new date format interchange standard that provides for only a 4-digit year format (NCITS L8 3.30). The recommended standard has been forwarded to ANSI which has placed it on its list of standards to be published. The 2-digit year format has been excluded. The international standard, ISO 8601:1988 (under the auspices of ISO TC154), has not been changed.

REQUIRED CHANGES

The ECMAScript specification provides functions for processing 2-digit and 4-digit years directly (see sections 15.9.5.5, 15.9.5.6, 15.9.5.7, 15.9.5.10, 15.9.5.11, 15.9.5.36, and 15.9.5.38) and other functions that

use 2-digit or 4-digit years based on the prototype date arguments.

The 2-digit year option should be left out of the specification entirely for the following reasons:

1. The Year 2000 problem is based on this option and will provide no end of frustration for implementers who have to justify why this option is still part of the ECMAScript specification, in view of the attention the problem has received already.

2. The liability of resellers and implementers will come more and more into focus as users rely on the court system to determine who is responsible for correcting problems based on this option.

3. The specification allows for implementations with extended capabilities, as long as the extended capabilities do not cause erroneous operation of the standard requirements. The ECMAScript specification is clear in providing two sets of functions that treat dates differently. The lack of 2-digit date processing functions should, ostensibly, not interfere with 4-digit year processing. Two-digit year processing may be implemented within the realm of extensions without causing undue influence on the standard use of the specification.

4. Not withstanding the need to provide functionality of de facto implementations, notes in the specification to the effect that these functions are provided for backward compatibility have no meaning with respect to this standard since there were no previous nationally or internationally recognized standards.

>>>>>> Action: Partial acceptance --- Sections 15.9.5.5 and 15.9.5.38 have been removed from the normative section of the ECMAScript standard. These sections will remain as informative text describing current practice to allow for a transition to more appropriate programing practice. Related changes will be made in other sections including 15.9.3 and 15.9.4. The committee while discouraging its use has chosen to retain the special function to dates in the range 0 to 99 to be off set by 1900. <<<<<

Character Related Issues

D3. All references to "unicode" string of characters should be changed to "UCS" strings or characters. This relates at least to clauses 4.3.16, 4.3.17 5.1.4 6 7 7.7.4 8.4 11.8.5 11.9.3 15.1.2.4 15.5.3.2 15.5.4.5 It is necessary to specify that this means UCS-4, or possibly UTF-8

>>>>>> Action: Partial acceptance --- Please see the Action note at the end of this section on Character issues. <<<<<

J3.1 Repertoire of charter

In the clause of 15.3.2, the String.fromCharCode is specified. This method is specified based on an assumption that BMP of ISO/IEC 10646 can contains every character in the world, since the method has 16bit dependency. Per request from users, right now ISO/IEC JTC l/SC2 is working to specify additional plains of ISO/IEC 10646 with the understanding of 16bit space is not sufficient to encode characters required for some applications. Therefore, this standard also should not have the 16bit dependency, then the return value of the method should be a integer value regardless of 16bit or 32bit. In addition to the above particular problem, JNB does not understand why the method need to be standardized, since I/O

functionality is outside of this standard, and this standard allows addition of methods and properties to conforming implementations. If this standard include I/O function as JavaScript or JScript have, JNB can understand the requirement to convert character to character code supported by its platform. but it is not the case of ECMAScript. Also, it is quite bad programming manner to check an attribute of character from its code point. If there is any requirement to check an attribute of character, a future revision of this standard should provide such functionality in the manner being "locale" sensitive. One suggestion might be simply remove the method from the standard, and make it enhancement by the specific implementations.

>>>>>> Action: Partial acceptance --- Please see the Action note at the end of this section on Character issues. <<<<<

J3.2) Upper-/lower.casing It is widely understood that upper and lower-casing rules are language and/or culture dependent. Therefore, if such language sensitive case conversion in the requirement. the functionality should be provided in the manner of "locale" sensitive in a future revision of this standard. If not, this standard should specify minimum requirement that is common to every language, such as case conversion rule for the Latin characters specified in ISO/IEC 646, then behavior of outside of ISO/IEC 646 should be specified as implementation defined.

>>>>>> Action: Partial acceptance --- Please see the Action note at the end of this section on Character issues . <<<<<

J3.3) Character literal outside of ISO/IEC 646 repertoire

The ECMA-262 allows literal representation method such as YuXXXX. Having the format is fine, but the description should be specified, such as "Character short identifiers headed by Yu are defined as follows. " Per request from the ISO/IEC JTC 1/SC22. the ISO/IEC JTC 1/SC2 proposed a short hand representation of character name that has one to one correspondence with character long name used throughout of IS0 character set standard, i.e. character code point of ISO/IEC 10646. then the second edition of the ISO/IEC TR 10176 that is approved recently recommended the support of the form of literal representation in order to specify character literal in an character code independent manner. The character short identifier looks very similar to the code point of ISO/IEC 10646, but the big difference would be the correspondence between the character short identifier and a character will be maintained even if code point assignment of the character is changed by a corrigenda or an amendment of ISO/lEC 10646. JNB thinks, the change of the definition of YuXXXX may not impact to the actual technical contents of this standard, but contribute to make the standard independent from any encoding system. Also, if the comment 3.1 is accepted, JNB would suggest to specify YuXXXXXXXX form in addition to YuXXXX, so that the character included in ISO/IEC 10646 but allocated outside of BMP become able to be represented.

>>>>>> Action: Partial acceptance --- reference the following text

E1)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.

>>>>>> Action: Accepted --- New wording will be added to the conformance section. Also reference the following text.

Resolution to Comments on Character Related Issues

To address the Character Related issues changes were made in many sections of the ECMAScript standard. Described below is the wording to added to Sections 2 and 3 to the majority of the comments . Additional changes have been made that clarify when ECMAScript operations refer to codepoints (cc data elements) and not characters as described in the original draft ISO/IEC standard.

Text to be added to Section 2 - Conformance.

"A conforming implementation of this edition of ECMAScript 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."

Please note that the text below is included here to explain the rationale of the proposed text and does not need to be included in the normative part of ECMAScript.

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.

From a Unicode conformance point of view, what is important is 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 ECMAScript, and other programming languages, need to do is let a data stream of 16-bit values pass through the process cleanly and with no damage to the data.

Text to be added to Section 3 - References.

Unicode: "Unicode Inc. (1996), The Unicode Standard TM, Version 2.0. ISBN: 0-201-48345-9,

Addison-Wesley Publishing Co.: Menlo Park, California"

ISO 10646: "ISO/IEC 10646-1:1993 Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane"

The additional information provided below is only for reference and not for inclusion in the ECMAScript normative part of the standard. The information highlights the contents of the conformance clauses of Unicode 2.0 and ISO 10646:

Additional Information from Unicode 2.0 Conformance:

This is spelled out in detail in Chapter 3, Conformance, of The Unicode Standard 2.0. Here are the bulleted items:

- Byte Ordering:

- C1 A process shall interpret Unicode code values as 16-bit quantities.

- C2 The Unicode Standard does not specify any order of bytes inside a Unicode value.

- C3 A process shall interpret a Unicode value that has been serialized into a sequence of bytes, by most significant byte first, in the absence of higher-level protocols.

- Invalid Code Values

- C4 A process shall not interpret an unpaired high- or low-surrogate as an abstract character.

- C5 A process shall not interpret either U+FFFE or U+FFFF as an abstract character.

- C6 A process shall not interpret any unassigned code value as an abstract character.

- Interpretation

- C7 A process shall interpret a coded character representation according to the character semantics established by this standard, if that process does interpret that coded character representation.

- C8 A process shall not assume that it is required to interpret any particular coded character representation.

- C9 A process shall not assume that the interpretations of two canonical-equivalent character sequences are distinct.

- Modification

- C10 A process shall make no change in a valid coded character representation other than the possible replacement of character sequences by their canonical-equivalent sequences, if that process purports not to modify the interpretation of that coded character representation.

Additional Information from ISO 10646 Conformance clause:

This is spelled out in detail in clause 2, Conformance, of ISO/IEC 10646-1:1993. Here is the essence of that clause.

2 Conformance

2.1 General

Whenever private use characters are used as specified in ISO/IEC 10646, the characters themselves shall not be covered by these conformance requirements.

2.2 Conformance of information interchange

A coded-character-data-element (CC-data-element) within coded information for interchange is in conformance with ISO/IEC 10646 if

a) all the coded representations of graphic characters within that CC-data-element conform to clauses 6 and 7, to an identified form chosen from clause 14 or Annex Q or Annex R, and to an identified implementation level chosen from clause 15;

b) all the graphic characters represented within that CC-data-element are taken from those within an identified subset (clause 13);

c) all the coded representations of control functions within that CC-data-element conform to clause 16. A claim of conformance shall identify the adopted form, the adopted implementation level and the adopted subset by means of a list of collections and/or characters.

<<<<<

Comments by Section

French Title

F 1) Qualifier: major editorial

Rationale: French title inaccuracy

Proposed change

Change the French title for the following :

ECMAScript : un language de programmation multiplate-forme à usage général

>>>>>> Action: Accepted --- French title will be changed. <<<<<

Table of Contents

N2) Contents: (ed)

It is requested that annexes containing the collected syntaxes will be provided in the final document.

>>>>>> Action: Not accepted --- The committee feels such a section would be useful but will leave this task to others. <<<<<

U6.5) Page vi: typo on top of page in PDF version.

>>>>>> Action: Accepted --- Document corrected <<<<<

Section 1 - Scope

J-E3) We need more than one liner to define the scope of the standard.

>>>>>> Action: Not accepted --- The committee feels this scope is clear in conjunction with the other information provided. <<<<<

Section 2 - Conformance

J1.1) Implementation limits and implementation defined matters

The ECMA-262 does not specify implementation limits nor implementation defined items, therefore it is considered that conforming implementation of the ECMAScript must meet with everything described in the ECMA-262.

Besides, the clause 7.5 says "An identifier is a character sequence of unlimited length" without specifying any implementation limits of l number of characters and numbers of identifiers. It implies that every conforming implementation must support identifiers that consists of millions of characters and accept millions of identifiers in a program. Japanese National Body (JNB) believe that the requirement would be too tough for any implementation.

Therefore, JNB would suggest that the ECMA- 262 should have "implementation limits" clause and specifies minimum requirements for program portability in the clause, then provide a clause, "implementation defined matter" and list the items that a conforming implementation can define.

For example, specify minimum requirements of length of an identifier as 256 in the implementation limits clause and specify the number of character allowed for identifiers as implementation defined matter, so that such an implementation becomes conformance to this standard that takes first 1024 characters of the identifier as meaningful and ignore the remaining characters.

The length of identifiers, the number of identifiers in a program, and the length of a line should be implementation defined.

The clause 7.2 introduces the concept Line terminators. But the means of line termination is file system dependent, e.g. FIXED type dataset of IBM System 390 does not have any line terminator character. So, the means of line termination should also be implementation defined, as far as the scope of this standard is general purpose.

>>>>>> Action: Not accepted -- Environments must provide line terminator characters. ECMAScript follows Java precedent in this respect. <<<<<

J1.2) Direct reference to documents outside of ISO/IEC standard

The ECMA-262 directly refers to technical contents of document/specifications outside of de jure standards. The bad examples are of reference to Unicode book and a RFC.

Since those documents are out of control of standard body, once the documents are revised, a once-conforming implementation of this standard may suddenly become non-conforming.

Therefore, only international de jure standards, i.e: ISO/IEC, can be referred to by normative part of this standard. For example, reference to Unicode book should be replaced with the reference to the ISO/IEC 10646, If there is no existing ISO/IEC standard that is equivalent with the technical contents of what referred to by this standard, a clause or a normative annex should be provided in this standard, then specify the technical contents in the clause or annex.

>>>>>> Action: Partial acceptance --- The Document has been reviewed to remove references to documents outside of ISO/IEC standards were appropriate. Please see additional information in the Action note at the end of the above section on Character Related issues. <<<<<

J1.3) "Discussion" clause

ECMA-262 has clauses named "discussion" According to the IS0 directives part 3. these clauses should be normative portion of this standard and the contents in the clauses are included in the requirements for conformity.

However, my impression on these is different. They sounds more like private notes or memoranda.

If the contents of the discussion clauses does not have requirements for conformity, the clause should be NOTES or it should be clarified that the discussion clauses are informative portion of this standard in the conformance clause.

>>>>>> Action: Accepted --- Use of "discussion" will be changed to "notes" to make a clear indication that the text is informative. <<<<<

J2) Data representation in a datatype

Programming language standard that does not have binary/object level portability as its objectives should not specify data representation of a datatype. In order not to restrict freedom of implementation. In order programming language standards to be independent from any encoding technology, the datatype should be specified by repertoire of data that the datatype can contains.

In this sense, the String type should be specified as the set of all finite ordered sequence of zero or more character data type, then should have a definition of character type as that the repertoire of the character data type shall be whole/entire repertoire of 1SO/IEC 10646,

Note that the ISO/IEC 10646 has the concept of subset, so if this standard allows an implementation that support a subset of ISO/IEC 10646, the minimal subset should be specified by this standard and actual repertoire of character should becomes implementation defined.

The same thing can be applied for the Number type. Usually, a data type for numeric data is specified by limits of the value, e.g., -128 through 127.

If this standard need to have a wide range of exact integer values, e.g., -2^40 through +2^40 to assure the exact calculation of Date values in milliseconds, this standard should specify so, instead of referring to IEEE 754 and concluding the integer value range.

Also. if this standard need some severe requirement on the precision of real (floating) values, this standard should specify so by giving necessary minimum requirements. Many programming languages have tried to make their specifications as, "cross-platform" as possible from the users' point of view. especially for the purpose that numerical algorithms can be programmed in "cross-platform" way. They specify minimum requirements (for all implementation) and introduce constant names such as MAX_VALUE, MIN_VALUE etc. to make platform defined values available to users. Otherwise, an implementation that uses a representation which precision is more accurate/large than IEEE 754 becomes not conformity to this standard. For those point it might help to consult

ISO/IEC 11404:1995 Information Technology - Language Independent data types.

ISO/lEC 10967:1994 Information Technology . Language Independent Arithmetic

- part 1: Integer and floating point arithmetic.

JNB is skeptical if ECMAScript need special values such as NaN, Infinity, etc. for itself. Infinities are returned when the computation yields "overflow"; ECMAScript has no "Notification" mechanism to handle "overflow": and it might be a way to continue the computation without interruption that ECMAScript requires Infinities as continuation values in those cases. But NaN is yielded only when some of arguments ie already a NaN. ECMAScript could permit an implementation which has representation of Infinities but of NaN.

JNB is also skeptical if ECMAScript need such a high precision as of current specification on floating point computation. Thin standard requires some specific real values, such as E, PI, LNlO, be available as closest possible floating numbers in 53 byte accuracy. Nevertheless all the defined functions are left unspecified about their returning values accuracy. If ever high precision computation were mandatory to ECMAScript, those functions should have been specified with severe accuracy requirement on their results.

De jure standard should not hinder the future improvement of technology as far as possible. Note that some programming language that has requirements on binary portability, such as BYTE CODE of Java, may need to specify internal representation of data in a datatype. But, JNB does not think that ECMAScript has such binary portability requirement.

For improvement of this standard, JNB would suggest that this standard should align with recently approved ISO/IEC TR 10176 and IS0 standard regarding language independent data types.

>>>>>> Action: Partial acceptance --- The standard will make it clear that it deals with type-byte encodings of characters. Please see the Action note at the end of the above section on Character Related issues for additional information on ISO/IEC 10646. The committee as chosen to leave the math functions unchanged. <<<<<

N3) Section 2 (ma)

The conformance clauses in this section, and in particular the last one leave too much room for non-standard extensions to the language. Such extensions will lead to portability problems. It is unclear how conformity of implementations will be checked against conformance clauses as have been given here.

>>>>>> Action: Not accepted --- Though portability will be an issue the current wording reflects the committee's intention. The goal of this first version of ECMAScript was to create a base standard. It is expected that the use of non-standard extensions will be reduced through the development of future versions of ECMAScript. <<<<<

U6.6) Clause 2, page 13:

If a conforming implementation can support *any* syntax, then how are conforming implementations tested? The conformance clause should be worded differently to identify non-conforming programs.

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E4 & N4 & U1.1) 2 Conformance

"section 0" should be "section 7.4.3"

>>>>>> Action: Accepted --- Document corrected <<<<<

Section 3 - References

D1) The standard needs to be aligned with ISO and IEC standards in the area. These include:

on page 1, clause 3, references:

ANSI X3.159 programming language C, should read ISO/IEC 9899:1996

including AM1 and TCOR1.

ANSI/IEEE 1754 should maybe be "754".

Unicode consortium unicode standard 2.0 should be replaced by :

ISO/IEC 10646-1:1998 including TCOR 1 and AM 1-9, plus

ISO/IEC DIS 14651 International sorting order, and

ISO/IEC DIS 14652 Specifications for cultural conventions

Then there is no need to refer the non-de-jure Unicode specification.

The Java specification is not used normatively, and can be moved to a bibliography section.

The specific statements needed of RFC1738 can be incorporated directly in the standard, it is about encoding in characters of control characters.

We could not ascertain the normative usefulness of the Ungar and Smith reference, it can most likely be moved to a bibliography section.

Then the normative references is only de jure standards.

>>>>>> Action: Accepted --- Some of this comment will also be moved to section 4. <<<<<

D2) The following references should be added:

ISO/IEC 646 (instead of ASCII)

ISO/IEC 6429 - for control characters.

ISO/IEC DIS 15897 - for reference to locales/fdcc-sets.

>>>>>> Action: Partial acceptance --- Reference to ISO/IEC 646 has been added, others are not mentioned. <<<<<

J-E1) 3 Reference

Only de jure standard referred to be from normative part of the standard that is related with conformance of this standard should be listed in the reference clause. Every reference documents outside of de jure standards and the ones just help to understand about the technical contents this standard should be removed from the clause or move to an informative annex.

>>>>>> Action: Accepted --- Document corrected <<<<<

N5) Section 3 (ed)

It is requested that, where possible, references to IS0 standards will be provided. The following standards have been referenced in the document, but are not mentioned here: ASCII, HTML.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.2) pp. 1, 3 References, line 1.

ANSI X3.159-19989 is the original C standard. That was withdrawn and replaced by the ANSI/ISO C standard ANSI/ISO 9899:1990, adopted in 1990. (An addendum was added in 1996, but I think the 1990 version reference will be sufficient.)

>>>>>> Action: Partial acceptance --- Document corrected to reference ISO/IEC 9899:1996 <<<<<

U2.2 & U6.7) The reference to the C standard should be ISO/IEC 9899:1993, not the ANSI document.

>>>>>> Action: Partial acceptance --- Document corrected to reference ISO/IEC 9899:1996 <<<<<

U4.1) On p. 1, Section 3 (references), the references should be to the ISO/IEC versions of the standards mentioned.

>>>>>> Action: Accepted --- Document corrected <<<<<

U6.3 & U6.9) Unicode should be replaced with ISO 10646-1 BMP (universal character) throughout the document.

>>>>>> Action: Partial acceptance --- Please see the Action note at the end of the above section on Character Related issues. <<<<<

U6.8) Add URL for RFC 1738.

>>>>>> Action: Not accepted --- The reference was removed. The description of the mapping was already present. <<<<<

Section 4 - Overview

J-E2) 4 Overview

The first word of the clause 4, i.e. "EMCAScript" should be replaced with "ECMAScript"

>>>>>> Action: Accepted --- Document corrected <<<<<

N6) Section 4 (ed)

A scripting language is intended for use by both professional and non-professional programmers and therefore there may be

-- The implication shown by the use of 'therefore' is unclear.

-- The sentence seems incomplete.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.3) pp. 1, 4 Overview, line -3

Re `informalities and build', either strike `and' or add the missing noun that should follow it.

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E5) 4.1 There are "server-side" and "server side." They should be one representation either with or without hyphen.

>>>>>> Action: Accepted --- Document corrected <<<<<

N7) Section 4.1 led)

A web browser provides en .,... (isn't this prescribing too much? 4 times)

--> A conforming web browser can/may provide en .

Typo: error: and abort

Typo: clients, and files. and

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E6 & U1.5 & U4.2) 4.2 In the last sentence of the last paragraph, "anddefined" should be "and defined."

>>>>>> Action: Accepted --- Document corrected <<<<<

N8) Section 4.2 (ed)

It is unclear how a syntax can be `relaxed'. A syntax is simply a description.

Typo: anddefined

Typo: missing full stop

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.4) pp. 2, 4.2 Language Overview, line -4.

Should Java be indicated as a trademark here (or possibly in the front matter)?

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E7 & N9 & U1.6 & U4.3) 4.2.1 In the second sentence of the second paragraph, "aprototype" should be "a prototype." In the figure. "Cfp" should be "CFp" in order to be consistent with the description below.

>>>>>> Action: Accepted --- Document corrected <<<<<

N9) Section 4.2.1 (ed)

Comma: contains. share

Figure: Cfp -> CFp

Figure; There is no meaning given for the normal arrow used form CF to CFp.

>>>>>> Action: Accepted --- Document corrected to show legend for solid link. <<<<<

U1.7) pp. 3, 4.2.1 Objects, line 8.

`The following diagram may illustrate this discussion:' That doesn't sound like it definitely does. Either make the diagram illustrate it or improve the wording.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.8) pp. 3, 4.2.1 Objects, line 13.

Strike the colon.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.9) pp. 3, 4.2.1 Objects, line 15.

`on the fly' sounds a bit colloquial to me. How about `dynamically' or `at run time'?

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.10) pp. 3, 4.2.1 Objects, line 16.

Re `any of its properties.' What is the subject referred to by `its'? I guess its refers to an object, but `its' is singular and `objects' is plural.

>>>>>> Action: Accepted --- Document corrected <<<<<

U6.10) Subclause 4.3:

This should be broken out as a separate clause. The beginning of clause 4 implies that the clause is informative; it appears that subclause 4.3 should be normative.

>>>>>> Action: Accepted --- Document corrected <<<<<

N10) Section 4.3.1 (mi)

A type is a set of data values.

-- Are non-homogeneous sets allowed?

In general, the correct functioning of a program is not affected if different data values of the same type are substituted for others. Well , it depends upon what is meant by `in general' and `correct', but as a general statement this seems to be incorrect for any programming language.'

>>>>>> Action: Accepted --- Wording has been removed. <<<<<

U4.4) On p. 3, Section 4.3.1 (Type), "A type is a set of data values. In general, the correct functioning of a program is not affected if different data values of the same type are substituted for others." This sentence is unclear because the correct functioning of a program does depend on the proper sequencing of values.

>>>>>> Action: Accepted --- Wording has been removed. <<<<<

U1.11) pp. 3, 4.3.3 Object, line 1.

Change `properties which contain' to `properties each of which contains'.

>>>>>> Action: Accepted --- Document corrected <<<<<

N11) Section 4.3.9 (mi)

The notion of a 'variable' has not been defined in this section.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.12) pp. 4, 4.3.9 Undefined, heading.

Add `value' to the heading as in 4.3.13 and 4.3.16.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.13) pp. 4, 4.3.11 Null, heading.

Add `value' to the heading as in 4.3.13 and 4.3.16.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.14) pp. 4, 4.3.13 Boolean value, line 1.

Strike `either'.

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E9) 4.3.15 In some Cases, "Boolean object" is used, but in some other cases, "boolean object" is used. Whether a word like "boolean' should start with a capital letter or not, should be consistent throughout the document. The same thing applies to 4.3.21 of "Number object" and "number object,"

>>>>>> Action: Accepted --- Document corrected will be made consistent <<<<<

N12) Section 4.3.15 (ed)

This is an example of . .

-- This sections seems to be misplaced.

>>>>>> Action: Not accepted --- The committee feels it is correctly placed. <<<<<

U1.15) pp. 4, 4.3.15 Boolean object, line 5.

Replace `in this case it is' with `the ability'.

>>>>>> Action: Accepted --- Document corrected <<<<<

N13) Section 4.3.16 (ed)

of the type String and is the set of .

-- the latter part of that sentence seems misplaced (see also 4.3.17)

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.16) pp. 4, 4.3.16 String value, line 1.

It seems to me that a string value is one of the set of all finite ordered sequences not the whole set.

>>>>>> Action: Accepted --- Document corrected <<<<<

N14 & U1.17) Section 4.3.19 (ed)

. . . a number value _is_

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.18) pp. 5, 4.3.20 Number type, line 1.

``In ECMAScript the set of values represent the double-precision 64-bit format IEEE 754 value ..." sounds like there is only 1 64-bit format value. Perhaps it should say ``In ECMAScript the set of values represent all the double-precision 64-bit format IEEE 754 values including the special "Not-a-Number" (NaN) value, positive infinity, and negative infinity."

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.19) pp. 5, 4.3.22 Infinity, heading.

Add `value' to the heading as in 4.3.13 and 4.3.16.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.20) pp. 5, 4.3.22 Infinity, line.

Is `Infinity' set in the correct typeface? The values true and false are set differently in 4.3.13.

>>>>>> Action: Accepted --- Document corrected <<<<<

U6.1) It appears that "undefined behavior" is not defined anywhere.

>>>>>> Action: Not accepted --- The committee feels that these terms are well defined outside of this standard. <<<<<

U6.2) Missing definitions: client-server architecture and client-side

>>>>>> Action: Not accepted --- The committee feels that these terms are well defined outside of this standard. <<<<<

Section 5 - Notation Conventions

N15) Section 5.1.2 (ma)

. . It defines a set of productions starting from the goal symbol

Input. that . . .

-- This symbol cannot be found in section 7.

Common programming languages do not need full parsers for analyzing the lexical structure of a program text. The set-up of this parser cannot be determined because the structure of the grammar is unclear.

>>>>>> Action: Accepted --- Document corrected <<<<<

N16) Section 5.1.2 (mi)

A multi-line comment is likewise simply discarded if it contains no line terminator: but . . .

-- it is unclear how a _multi-line_ comment cannot contain a line terminator (but see also a later comment)

>>>>>> Action: Accepted --- Document corrected <<<<<

N17) section 5.1.4 (mi)

. if an end-of-line character ___ -- is an end-of-line character equivalent to a line terminator?

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.22) pp. 8, 5.2 Algorithm conventions, heading.

Upcase C in `conventions' to match all other level-2 heads.

>>>>>> Action: Accepted --- Document corrected <<<<<

U6.11) Subclause 5.2:

Remove paragraph 1, "We often use ..."

Replace "X is Y" with wording that uses "shall".

>>>>>> Action: Accepted --- Wording has been changed to remove use of "we". Use of 'shall' is not accepted due to the scope of the changes required. <<<<<

J-E9 & U1.23) 5.2 In the last sentence of the third paragraph, "mustbe" should be "must be."

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.21) pp. 6, 5.1.5 Grammar Notation, line -7.

Change `recursive, that is to say' to `recursive; that is'.

>>>>>> Action: Accepted --- Document corrected <<<<<

Section 6 - Source Text

D4) clause 6: change ASCII to ISO/IEC 6464 IRV. Four hexadecimal digits are too little to represent UCS characters of ISO/IEC 10646-2 (planes outside BMP).

>>>>>> Action: Partial acceptance --- Reference changed to ISO/IEC 646 IRV <<<<<

U4.5) On p. 9, Section 6, they describe the use of Unicode in comments and string literals. We program in English and seldom use Unicode, I would want to be sure that others feel that the approach here is consistent with other programming languages and the intended use of Unicode.

>>>>>> Action: Partial acceptance --- Please see section on Character Related Issues above. <<<<<

U6.12) Clause 6:

ASCII is mentioned but no reference in the References clause.

An ISO standard should be referenced rather than the ASCII standard.

>>>>>> Action: Accepted --- Document corrected <<<<<

Section 7 - Lexical Conventions

D5) Clause 7: strictly speaking control characters are defined in ISO/IEC 6429.

>>>>>> Action: Not accepted --- Please see section on Character Related Issues above. <<<<<

U1.24) pp. 9, 7 Lexical Conventions, line 1.

The source text of a[n] ECMAScript program ...

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.25) pp. 9, 7 Lexical Conventions, line 2.

A token is a sequence that comprise[s] ...

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.26) pp. 9, 7.1 White Space, line 2.

... each other[,] but ...

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.27) pp. 10, 7.2 Line Terminators, line 1.

Replace

``Line terminator characters, like white space characters, are used to improve source text readability and to separate tokens (indivisible lexical units) from each other. Unlike white space characters, ..."

with

``Like white space characters, line terminator characters are used to improve source text readability and to separate tokens (indivisible lexical units) from each other. However, unlike white space characters, ..."

>>>>>> Action: Accepted --- Document corrected <<<<<

U6.13) Subclause 7.2:

Line terminators should include the Next Line character (I think is it 0x84 or 0x85).

>>>>>> Action: Not accepted -- ECMAScript follows Java precedent. <<<<<

N18) Section 7.3 (mi)

The description is unclear about Line terminators in Multi-line comments

>>>>>> Action: Accepted --- Document corrected <<<<<

N19) Section 7.3 (mi)

The production for MultiLineNotForwardSlashOrAsteriskChar ::

SourceCharacter but not forward-slash / or asterisk *

seems to be better written as:

SourceCharacter but not (forward-slash / or asterisk *)

This case occurs more often.

>>>>>> Action: Partial acceptance --- Would require substantial change to notation; existing production has been clarified. <<<<<

U4.6) On p. 10-11, Section 7.3, the syntax for comments seems more complicated than necessary, particularly when things like special Unicode characters are described earlier. Is this the way the similar syntax is done in C or C++?

>>>>>> Action: Not accepted --- The committee agreed that the current syntax is correct. <<<<<

U6.14) Subclause 7.3.3:

Bullets on page 26: exponents appear in font too small.

>>>>>> Action: Accepted --- Document corrected <<<<<

U4.7) On pp. 11-12, Sections 7.4.2 and 7.4.3, keywords and future reserved words, it is not clear whether the case of the characters is significant. I thought ECMAScript was case sensitive, but I didn't see that mentioned.

>>>>>> Action: Not accepted -- The standard makes it clear elsewhere that ECMAScript is case-sensistive. <<<<<

E2) 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

>>>>>> Action: Accepted --- Requested text has been added <<<<<

D6) 7.5: DOLLAR SIGN should not be in the identifier list, according to recommendations in TR 10176. 7.5 should refer to the "i18n" specification of ISO/IEC 14652 for definitions of letters and digits.

>>>>>> Action: Partial acceptance --- ECMAScript follows Java precedent. A comment will add that $ should only be used for mechanically-generated code. <<<<<

U1.28) pp. 13, 7.7.3 Numeric Literals, line -4.

``... ideally using IEEE 754 round-to-nearest ..." The word `ideally' doesn't sound like good standard's language. What's the implication if the implementer doesn't use this?

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.29) pp 15, 7.7.3 Numeric Literals, line 7.

The use of nested parentheses is rather unusual. How about replacing

``A digit is significant if it is not part of an ExponentPart and (either it is not 0 or (there is a nonzero digit to its left and there is a nonzero digit, not in the ExponentPart, to its right))."

with

``A digit is significant if it is not part of an ExponentPart and

-- either it is not 0 or,

-- there is a nonzero digit to its left and there is a nonzero digit,

not in the ExponentPart, to its right."

>>>>>> Action: Accepted --- Document corrected <<<<<

E3) 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.

>>>>>> Action: Not accepted --- No limit will be used <<<<<

D7) in 7.7.4 UnicodeEscapeSequence should be renamed UcsEscapeSequence. Care should be taken that all UCS characters (31-bit) can be handled, e.g. in UcsEscapeSequencee, HexEscapeSequence, and OctalEscapeSequence.

>>>>>> Action: Not accepted --- Current text will be used for compatibility. <<<<<

J-E1O) 7.7.8 In the last paragraph, last part parenthesis of the second sentence, "(in the sense defined in section 8.4)", the referred section number should be "8.5." Also, in OctalIntegerLiteral ::

0 OctalDigit

OctalLiteral OctalDigit

OctalLiteral" should be 'OctalIntegerLiteral."

>>>>>> Action: Accepted --- Document corrected <<<<<

N20) Section 7.8.1 (ed)

The notion of 'the header of a for statement' has not been defined.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.30) pp 18, 7.8.1 Rules of automatic semicolon insertion, line 14.

Replace ``These are all the restricted ..." to ``These are the only restricted ..."

>>>>>> Action: Accepted --- Document corrected <<<<<

Section 8 - Types

U1.31) pp. 19, 8 Types, line 2.

Strike `called' and put the list ``Reference, List, and Completion" in parens as for the six standard types in the line above.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.32) pp 19, 8.3 The Boolean Type, line 1.

Replace

``The Boolean type represents a logical entity and consists of exactly two unique values. One is called true and the other is called false."

with

``The Boolean type represents a logical entity having two unique values, called true and false."

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.33) pp. 20, 8.5 The Number type, line 7.

Instead of ``... all NaN values are the same." might it be better to say that ``... all NaN values compare equal"?

>>>>>> Action: Accepted --- Indistinguishable will be used in new wording. <<<<<

U4.8) On p. 20, Section 8.5 (and then other pages and sections), the number type is described in terms of IEEE 754. (Isn't there an ISO/IEC number for this standard?) In Section 11.5.3, they have some differences in the % operator. There has been some discussion about how this standard is used in Java. I would hope that things are done appropriately here. The use in this proposed standard (and in Java and other languages) might motivate a review of IEEE 754.

>>>>>> Action: Partial acceptance --- Reference to IEEE 754 will be changed to IEC 559:1993. Use of the % operator is as intended. <<<<<

N22) Section 8.6 (mi)

Each property consists of a name. a Value and a set of attributes. This seems inconsistent with section 4.2

>>>>>> Action: Not accepted --- 4.2 is a non-normative overview and therefore simplifies. <<<<<

U1.35) pp. 21, 8.6.2 Internal Properties and Methods, line -2.

... implement[ation]-dependent ...

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.34) pp. 21, 8.6.2 Internal Properties and Methods, line 4.

Add ``, respectively" to the end.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.36) pp. 22, 8.6.2 Internal Properties and Methods, line -7.

Re `it is used internally ...', is the subject `it' referring to the value of a [[Class]] property? It's probably worth naming the subject explicitly.

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E11) 8.6.2.3 The item7., "Return Result(4)"should be "Return Result(6)."

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E12) 8.7 in 4 the paragraph, "A Reference consist of two parts, the base object and the property name" should be clarified such as "A Reference consists of two components, the base object and the property name" so that the "part" is actually the "component."

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E13) 8.7.4 In the item 6, there should be a forward reference to section 10.1.5 for the newly appeared undefined term, 'global object."

>>>>>> Action: Accepted --- Accepted as a helpful clarification. <<<<<

J-E14) Section reference. There are three types of section reference in parentheses (see section x.y.z), (section x.y.z), and (x.y.z). It would be consistent and much better to use the one style instead of mixing various formats.

>>>>>> Action: Accepted --- Document corrected <<<<<

N21) Section 8.8 (ed)

There are six standard types . . In section 4.2 these are called built-in types.

>>>>>> Action: Accepted --- Wording will be changed. <<<<<

J-E15) 8.9 In "abrupt completion," the word "completion" should also be italicized.

>>>>>> Action: Accepted --- Document corrected <<<<<

Section 9 - Type Conversion

J-E16 & N23) 9.1 In the column of Object, "(see section 8.6.2.5)" should be "(see section 8.6.2.6)."

>>>>>> Action: Accepted --- Document corrected <<<<<

E4) 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.

>>>>>> Action: Not accepted --- This will be considered for a future version of ECMAScript. <<<<<

J-E17) 9.5 In the step 6, "Result(5)" should be "Result(4)." 918 10.1.1 and many other sections. There are two formats for "implementation dependent"; with and without a hyphen. It would be better to define the technical term "implementation-dependent" and use it throughout the document.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.37) pp. 30, 9.8 ToString, line 1.

Re `attempts', what happens if it cannot convert its argument? I see no provision for the generation of a runtime error (like 9.9 provides on conversion failures).

>>>>>> Action: Accepted --- Use of 'attempts' will be removed. <<<<<

D8. clause 9.8.1 the Gay 1990 algorithm needs to be spelled out completely, for portability.

>>>>>> Action: Partial acceptance --- The algorithm will be included to avoid the need for a reference. <<<<<

Section 10 - Execution Contexts

N24) Section 10 (ed)

When control is transferred to ECMAScript executable code, we . . The use of `we' is not common in standardization documents.

>>>>>> Action: Accepted --- Document corrected <<<<<

N25) Section 10.1.1 third bullet (ed)

. ..The use of these attributes are described . . .is described, is probably intended here.

>>>>>> Action: Accepted --- Document corrected <<<<<

U6.15) Subclause 10.1.1

Change to standards wording "which we refer to ..."

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E19) 10.2.4 In the 4th bullet, "object object" should be "object."

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.38) pp. 33, 10.1.3 Variable instantiation, lines 1-2.

Remove extra vertical space between these lines.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.39) pp. 33, 10.1.3 Variable instantiation, lines 6-7.

Remove extra vertical space between these lines.

>>>>>> Action: Accepted --- Document corrected <<<<<

U6.16) Subclause 10.1.3

Vertical white space after first sentence -- remove it

>>>>>> Action: Accepted --- Document corrected <<<<<

E5) 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.

>>>>>> Action: Accepted --- Document corrected <<<<<

Section 11 - Expressions

E6) 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.

>>>>>> Action: Not accepted --- There are valid reasons for the rules. <<<<<

J-E20) 11.2.3 In the item 2, "section 0" should be "section 8.8."

>>>>>> Action: Accepted --- Document corrected <<<<<

J-M2) .-From the syntactic rules of 11.4 and 11.5, we can produce not well defined

arithmetic expressions such as "-3*-2."

>>>>>> Action: The semantics for these expressions is defined by operator precedence rules. <<<<<

E7) 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.

>>>>>> Action: Accepted --- Will be changed to implementation dependent. <<<<<

U1.40) pp. 44, 11.7.1 The left shift operator (<<), line 1.

Replace both occurrences of `argument' with `operand'.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.41) pp. 44, 11.7.2 The signed right shift operator (>>), line 1.

Replace both occurrences of `argument' with `operand'.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.42) pp. 45, 11.7.3 The unsigned right shift operator (>>>), line 1.

Replace both occurrences of `argument' with `operand'.

>>>>>> Action: Accepted --- Document corrected <<<<<

D9) clause 11.9.3 should refer to ISO/IEC 14651 for the complex sorting. We propose that ECMAScript

does include a more complex string comparison conforming to ISO/IEC 14651.

>>>>>> Action: Not accepted --- This can be considered in a future version. <<<<<

N26) Section 11.11 (ed)

6 Call GetValue((Result(5))

Bracket mismatch

>>>>>> Action: Accepted --- Document corrected <<<<<

Section 12 - Statements

N27) Section 12.2 (ed)

The reference to section 0 seems incorrect.

>>>>>> Action: Accepted --- Document corrected <<<<<

E8) 12.2 Variable statement

Globals explicitly declared with var should be marked DontDelete.

>>>>>> Action: Accepted --- Document corrected <<<<<

E9) 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 betterformulation is "Construct a value of type Reference whose base object is the next activation object on the scope chain and whose propertyname is the Identifier." Step 5 should then be: "Return Result(1)."

>>>>>> Action: Partial acceptance --- The concept of this comment is accepted. Exact wording will provided by the committee. <<<<<

J5) Ambiguous Syntactic Rules 12.5 The IF statement

The following sentence is mandatory just below the syntax rules:

'else' shall associate with the nearest 'if among 'if's in the same block (excluding those 'if's which are contained in the inner blocks) that precedes the 'else' and has no corresponding' else'.

(The sentence is borrowed from the C language standard.)

Without such a sentence the syntax remains ambiguous, since one cannot tell whether

if(a==b) if(c==d) x= 1; else x=2;

means

(1) if(a==b) {if(c==d) x= 1- else x= 2;}

or

(2) if(a==b) {if (c==d) x= 1.) else x= 2;

This phenomena has been well known for languages which have both forms of ifO and if0elseO for conditional

branching. Pascal may be consulted with this. It goes: The token 'else' shall not occur next to any if-statement which has no 'else' corresponding to its 'if'.

>>>>>> Action: Accepted --- Document corrected by adding additional wording. <<<<<

N28) Section 12.5 (mi)

The syntax given here allows for so-called dangling else problems. These seem not to be resolved.

>>>>>> Action: Accepted --- Document corrected by adding additional wording. <<<<<

E10) 12.6.3 The for..in statement, evaluation of for(var ...)

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

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.43) pp. 55, 12.7 The CONTINUE Statement, line 4.

Re ` ... may not be executed ...' The use of `may' in a formal specification can be troublesome, especially when used with the negative. Specifically, does `may not' mean `might not' or does it mean `shall not'? One imposes conformance requirements while the other doesn't.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.44) pp. 55, 12.7 The CONTINUE Statement, line 5.

Replace `at least one' with `a'. It seems to me that the number of nested while or for statements is irrelevant.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.45) pp. 55, 12.8 The BREAK Statement, lines 4 and 5.

See the comments for CONTINUE above.

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.46) pp. 55, 12.9 The RETURN Statement, lines 4.

See the first comment for CONTINUE above.

>>>>>> Action: Accepted --- Document corrected <<<<<

U6.17) Subclauses 12.8, 12.9, 12.10:

These subclauses refer to a program as "syntactically incorrect", but the corresponding behavior is not clear: what happens when the program fails. Furthermore, this response to bad syntax should be defined early in the document, say, in the Conformance clause.

>>>>>> Action: Not accepted --- Error behaviors are described in section 16. <<<<<

Section 13 - Function Definition

J-E21) 13 In the first paragraph, "the Identifier" is ambiguous in the sense whether it is in the FunctionDeclaration, or in the FormalParameterList. It should be clarified such as "the function Identifier."

>>>>>> Action: Accepted --- Section will be changed to clarify meaning. <<<<<

Section 14 - Program

N29) Section 14 (ed)

. 1. Process SourceElements for function declarations ..

From the description given, it can't be determined whether this needs to be done left to right or right to left.

1. Evaluate SourceElements.

....

. 4. Return Result(1)

It is unclear whether the result of step 4 is the result of the first term or of the last term of (1).

On two occasions 'is' is printed in italics.

On one occasion `is' is written without preceding space.

>>>>>> Action: Partial acceptance --- Type style and spacing changes will be made. The committee feels the current wording is clear. . <<<<<

Section 15 - Native ECMAScript Objects

E11) 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.

>>>>>> Action: Accepted --- Document corrected <<<<<

D10) clause 15.1.2.4: RFC1738 should be spelled out, it is not very complicated and thus a non-de jure reference can be removed. "ASCII" should be replaced by ISO/IEC 646 IRV. escape() and unescape() should be applicable to 31-bit UCS values.

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E22) 15.1.2.4 In the item 7, "nonblank ASCII characters" should be "nonblank characters. " It should be irrelevant whether they are ASCII or not.

>>>>>> Action: Accepted --- Document corrected <<<<<

U4.9) On p. 60, Section 15.1.2.4 (escape string), it's not clear why a different format is being used. There also seems to be over specification of some of the rules in this and surrounding sections.

>>>>>> Action: Accepted --- Document corrected <<<<<

N30) Section 15.1.3.5 (ed.) (ed))

. 15,6,2.

Commas?

>>>>>> Action: Accepted --- Document corrected <<<<<

E12) 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.

>>>>>> Action: Accepted --- Change will be made in section 8.6.2. <<<<<

E13) 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]].

>>>>>> Action: Partial acceptance --- Document changed to reflect a limit of 2^32-1 <<<<<

U1.47) pp. 68, 15.4.4 Array.prototype.sort(comparefn), line -6.

Replace `compared' with `compares'.

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E23) 15.4.4.4 In the first paragraph and in the item l, the term, "this object," appears which causes some confusion whether "this" is a special "this" or not. Use "the object" or "the array object" to avoid the misleading.

>>>>>> Action: Accepted --- Document corrected <<<<<

D12) clause 15.4.4.5. String.prototype.charCodeAt() shall return a integer less than 2**31.

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E24) 15.4.4.5 In the first paragraph, the second sentence, "The sort is not necessarily stable." may cause misunderstanding to whom those do not have expertise in the sorting. The precise meaning of the word "stable" should be added or explained.

>>>>>> Action: Accepted --- Document corrected <<<<<

E14) 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".

>>>>>> Action: Accepted --- Concept of this change is accepted. <<<<<

E15) 15.4.5.1 [[Put]](P, V)

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

>>>>>> Action: Accepted --- Document corrected <<<<<

D11) clause 15.5.3.2: UCS characters are 31 bit, not 16 bit. The right function to call is ToUint32().

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E25) 15.5.3.2 In the first sentence, "asthe" should be "as the."

>>>>>> Action: Accepted --- Document corrected <<<<<

E16) 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 15.5.4.4 and as well as the implicit intent.

>>>>>> Action: Accepted --- Document corrected <<<<<

E17) 15.5.4.4 and as well as the implicit intent.

>>>>>> Action: Not accepted --- Numbering error in ECMA comments. This text was a part of the above comment. <<<<<

D13) clause 15.5.4.11 and 15.5.4.12 Needs to refer to ISO/IEC 14652 specifications in the "i18n" fdcc-set for upper and lowercase equivalences, instead of Unicode values.

>>>>>> Action: Accepted --- Document corrected <<<<<

U4.10) On p. 76, Section 15.7.3.2 (Number.MAX_VALUE) starts "The value of Number.MIN_VALUE ..."

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.48) pp. 77, 15.8.1.5 LOG10E, line 2.

Re ``(Note that the value of Math.LOG2E is approximately the reciprocal of the value of Math.LN2.)"

I'm no mathematician, but I'm guessing that this section was cloned from 15.8.1.4, and that it should read as

follows instead:

``(Note that the value of Math.LOG10E is approximately the reciprocal of the value of Math.LN10.)"

>>>>>> Action: Accepted --- Document corrected <<<<<

U1.49) pp. 77, 15.8.2 Function Properties of the Math Object, line -2.

Re `[XXXREF]', is a cross-reference missing here?

>>>>>> Action: Accepted --- Document corrected <<<<<

J-M1) 15.8.2.11 & 15.8.2.12

The behavior in the case that argument x is equal to y is not specified.

>>>>>> Action: Accepted --- Document corrected <<<<<

J-E26) 15.9.1.1 In the 4th sentence, constants "iMin" and "iMax" appear as if they are ECMAScript constants. However, it looks just a convention in this place, and not used anywhere else. Avoid these constants.

>>>>>> Action: Accepted --- Document corrected <<<<<

U6.18) Subclause 15.9.1.1:

Reword "This span easily covers all of recorded human history ..." as specific year numbers in the common era (A.D.) and before the common era (B.C.).

>>>>>> Action: Accepted --- Document corrected to remove "iMin" and "iMax". <<<<<

U6.19) Subclause 15.9.1.3, para 2:

Remove/reword "Of course"

>>>>>> Action: Accepted --- Document corrected <<<<<

D14) clause 15.9.1.4: month numbers should be numbered from 1 to 12. This is analogeous to date number in 15.9.1.5 and conforming to ISO 8601.

>>>>>> Action: Not accepted --- The committee has chosen to the maintain the current text in order to maintain compatibility with current practice. A new function could be considered for a future

version. <<<<<

D15) clause 15.9.1.6: week day numbers should be numbered from 1 to 7 as argued in comment 14.

>>>>>> Action: Not accepted --- The committee has chosen to the maintain the current text in order to maintain compatibility with current practice. A new function could be considered for a future version. <<<<<

J4.2 Local time and daylight saving time {15.9.1.7 and 15.9.1.8}

The ECMA-262 have a concept of daylight saving time and functionality to convert local time with daylight saving time to UTC. However, without having any good mechanism the one to one correspondence between local time and UTC can not be guaranteed. Let's assume that at 2:00 of September lst, the local time will be back to l:OO. In the case, 1:30 of September lst should be converted to what value in UTC? Until no good mechanism is provided, this standard should not support local time and daylight saving time. Again, there is no problem from the view point of backward compatibility and conformance. Implementation can provide those functions as extensions. UTC support would be good enough for this standard. If further revision of this standard introduce the concept of locale, the local time support should be specified with a mechanism of daylight saving support, at that time.

>>>>>> Action: Not accepted -- Locale awareness is out of scope of the current standard. Also see section on Year 2000 issues above. <<<<<

D16) clause 15.9.1.8: refer to ISO/IEC 14652 for a possible reference to information on daylight savings.

>>>>>> Action: Not accepted -- Out of scope of the current standard. This will be implementation-defined. <<<<<

N31) Section 15.9.1.8 (ed)

. . /t.... ??

>>>>>> Action: Accepted --- Document corrected <<<<<

U6.20) Subclause 15.9.1.8:

Reword "by whatever means available" as standards words ("implementation-defined"? -- but this would require defining implementation defined).

Paragraphs 2 and 3: It is not clear what the required behavior is for a conforming implementation.

>>>>>> Action: Accepted --- Corrected wording has been added. <<<<<

D17) clause 15.9.1.12 and 15.9.2: please note that year is currently a four-digit integer.

>>>>>> Action: Not accepted -- The definitions do not require a 4-digit year. <<<<<

D18) clause 15.9.3.1-7: the result rules for year<99 makes it hard to talk about things at the time of the birth of Jesus Christ - it should be removed. The easiness it gives for dates in this century are not so useful just a couple of years from now. This is not a foolproof rule.

>>>>>> Action: Not accepted -- Would break many thousands of existing programs. <<<<<

E18) 15.9.5.34 setMonth and 15.9.5.34 setUTCMonth

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

>>>>>> Action: Accepted --- Document corrected <<<<<

D19) clause 15.9.5.39: refer to ISO/IEC 14652 for specifications of date formatting, and ISO/IEC 15897 for references to different locales.

>>>>>> Action: Not accepted --- Complete solution to this problem has been determined to be outside the scope of this committee. Results will become implementation dependent. <<<<<

Section 16 - Errors

U6.21) Clause 16:

This clause should be moved to the beginning and, possibly, incorporated in the Conformance clause.

>>>>>> Action: Not accepted -- This is not a conformance item. <<<<<