ECMA
Standardizing Information and Communication Systems

ECMA Working Group

Version 1 Comments

January 19, 1998

This document contains ECMA-262 comments received since the last working group meeting. The document is formatted as HTML mail.

There are three classes of comments:

1. lengthy tome from the Japanese National Standard Body
2. 2.Comments on unicode escape characters in URL's
3. 3.The NIST negative vote and its comments on two digit dates

The January 19 working group meeting should attack these as the first major agenda item. The time and place of the meeting will appear in mail tomorrow.

= Clayton =

I am sending this rather lengthy memo to you in order to make a better communication with you on what we have found last Friday meeting in Tokyo about the Fast Track voting on the subject standard.

This is intended to be personal and informal information, not the official arguments of Japan National Standard Body.

Again, this is lengthy, because this reflects collective wisdom and observation of our experts, so please accept this for a good token to improve both the technical and editorial contents of the proposed standard documents.

I pointed out 4 major problems: Conformance, Data representation, Characters, and Date. Also pointed out are minor technical and editorial problems.

Best Regards,

Toshi Kurokawa

Major technical comments

1 Conformance

The conformance section should be rewritten in order to clarify what are included in the requirements to conform to the standard and what are excluded. The following items 1.1 through 1.3 are problems, which I found so far.

1.1 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 number of characters and number 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. I believe that the requirement would be too tough for any implementation.

Therefore, I 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 of 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.

1.2 Direct reference to documents outside of ISO/IEC standard

The ECMA-262 directly refers to technical contents of documents/specifications outside of dejour 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 dejour 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.

1.3 "Discussion" clause

ECMA-262 has clauses named "discussions." According to the ISO 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.

2 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 datatype, then should have a definition of character type as that the repertoire of the character data type shall be whole/entire repertoire of ISO/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 datatype 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, eg., -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 specification 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 datatypes. ISO/IEC 10967-1:1994 Information Technology - Language Independent Arithmetic - part 1: Integer and floating point arithmetic.

I am 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 is already a NaN. ECMAScript could permit an implementation which has representation of Infinities but of NaN.

I am also skeptical if ECMAScript need such a high precision as of current specification on floating point computation. This standard requires some specific real values, such as E, PI, LN10, be available as closest possible floating numbers in 53 bits 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.

Dejour 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, I don't think that ECMAScript has such binary portability requirement.

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

3 Character related issues

3.1 Repertoire of charter

In the clause of 15.5.3.2, 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 1/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, I do 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 has, I 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.

3.2 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 is 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.

3.3 Character literal outside of ISO/IEC 646 repertoire

The ECMA-262 allows literal representation method such as \uXXXX. Having the format is fine, but the description should be specified, such as "Character short identifiers headed by \u are defined as follows. "

For your information, 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 ISO 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/IEC 10646.

I think, the change of the definition of \uXXXX 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, I would suggest to specify \uXXXXXXXX form in addition to \uXXXX, so that the character included in ISO/IEC 10646 but allocated outside of BMP becomes able to be represented.

4 Date Object

4.1 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 these 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 methods 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.

4.2 Local time and daylight saving time

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 1st, the local time will be back to 1:00. In the case, 1:30 of September 1st 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 time support, at that time.

Minor Technical and Editorial comments:

With ECMA-262, I found a lot of minor technical and editorial errors. Therefore, I would strongly suggest you should prepare a technical corrigenda of this standard, and republish it 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 in stead of the final version by accident, since there are so many spell errors.

Minor technical errors:

M1 15.8.2.11 & 15.8.2.12

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

M2 From the syntactic rules of 11.4 and 11.5, we can produce not well defined arithmetic expressions such as "-3*-2."

Minor Editorial errors:

E1 3 Reference

Only dejour 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 dejour 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.

E2 4 Overview

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

E3 1 Scope

We need more than one liner to define the scope of the standard.

E4 2 Conformance

There is no "section 0" for "future reserved words."

E5 4.2 In the last sentence of the last paragraph, "anddefined" should be "and defined."

E6 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.

E7 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."

E8 5.2 In the last sentence of the third paragraph, "mustbe" should be "must be."

I am sending another finding on if statement.

Regards,
--Toshi

5 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 if() and if()else() for conditional branching. If you prefer, you may consult with, say, Pascal, which put it as:

The token 'else' shall not occur next to any if-statement which has no 'else' corresponding to its 'if'.

Subject: Re: ecma-262 and %uxxxx notation in URLs

Date: Wed, 19 Nov 1997 09:24:52 PST

From: Martin J. Dürst <mduerst@ifi.unizh.ch>

To: <masinter@parc.xerox.com>

CC: url-i18n@unicode.org

Many thanks to Chris for discovering the %u in the ECMAscript documents.

On Sun, 16 Nov 1997, Glenn Adams wrote:

> At 06:37 PM 11/15/97 PST, Larry Masinter wrote:
> >I like %uxxxx/UCS2 better than %xx%yy%zz%ww (or whatever) using
> >UTF8, too. The only problem is that while UTF8 actually will
> >work in, say, FTP URLs, UCS2 won't.
> >
> >That is, you can go ahead and send
> >
> >   http://whatever.cn/%Ua0b4
> >
> >in the HTTP protocol, because HTTP doesn't require you to decode.

But some clients may try to do it, and most servers will try to do it, and it's not clear what the result will be. I have just made a try with Netscape 3.x on a SUNOS, and with our appache server.

In the directory at http://www.ifi.unizh.ch/mml/mduerst/ I have created a file with the name "%ua0b4.html". When I type the combined URL into the browser, I get back a

    400 Bad Request

with the explanation:

    Your browser sent a request that this server could not understand.

You can try it, too. When I change the file name to

    "%25ua0b4.html"

I of course get the right file back. So with the %u... proposal, we get a lot of backwards compatibility problems. Maybe good for people who want to sell new software, and for webmasters that can control all of their server installation, but very bad for a single user that wants to create new resource names maybe just by adding a link.

> >But what does

> >

> > ftp://whatever.cn/%ua0b4

> >

> >actually mean, vis a vis the FTP protocol? Or gopher? Or whatever?

>

> There's no ambiguity in its meaning. The issue is how to communicate to an FTP

> or Gopher service agent which employs a restricted character protocol.

There is no ambiguity in its meaning in terms of characters. But the FTP protocol either deals with file names in terms of octets, or it uses UTF-8. Using UTF-8 %xx%yy%zz%ww has the big advantage that it will work very well with future implementations of FTP without any changes to a browser. "%u..." will break many browsers and other tools.

The fact that the %HH (+UTF-8) notation works well with pure octet-based protocols, with protocols based on characters (e.g. IMAP folder names), and in particular with protocols that are in the transition between both (ftp, hopefully http), or where there is only a private convention to use UTF-8, is a big advantage.

That UTF-8 can not always uniquely be identified may look like a disadvantage, and in some cases it may indeed be slightly disadvantageous, but on the other hand, because of the ambiguous/ transitory state of many protocols, it has great advantages, because it more closely reflects the protocols and therefore creates less interoperability problems.

As for "I like better", I could agree. But we are doing the whole work not to see some "%..." stuff, but to see the real characters. "%..." is only a an escape if there is nothing else.

Also, HTML 4.0, in the form of a proposed recommendation, already describes how to convert URLs with "illegal" characters in them to "legal" URLs (i.e. ASCII only, with the usual additional restrictions) by using UTF-8 and %HH-encoding.

In summary, I think that although the ECMAscript definition is a nice idea, definitely so if we could start all over, it creates more problems than solutions. I therefore propose that we do the following:

– Add a backwards compatibility note to our emerging document
  that says that somewhere between the script engine and
  the central URL pivot point in the browser, %u....
  should be converted to UTF-8 + %HH (which can be done
  unambiguously).

– File a request with ECMA for a technical correction changing
  the spec so that it produces UTF-8 with %HH.

As for the "emerging document" mentionned above, I worked quite a bit on it last week, but I won't be able to finish it until the cutoff deadline for internet drafts on this Friday, as I have to spend most of my time for organizing my move.

Regards, Martin.

The context of %uXXXX is within the converted-to-string argument to unescape, and the result string of escape -- this is not a lexical convention in all strings, it is peculiar to those methods. And % was already reserved by those methods according to RFC 1738.

So the problem here was ECMA design-by-committee (I'm partly to blame). We wanted to continue to support %xx per 1738, but thought that UCS-2 would be used to spell characters beyond ASCII. At this point, I think removing %uXXXX from the ECMA spec is fine, because we don't yet ship any implementation that scans or produces it -- but MS may, and ECMA-262 is on its way to ISO, so is pretty well frozen.

A note deprecating %u and saying it will be removed from a future version of ECMA-262 (ISO-16262) might be enough.

/be

ECMAScript

Letter Ballot

Subject: Establishing the U.S. Position w.r.t the ECMAScript DIS
Reference: www.ecma.ch, ECMA Standard 262.
Question: I support the adoption of ECMAScript DIS as an ISO standard

Ballot period: 12 noon EST 11/12/97 through 12 noon EST 1/12/98.
Return ballot to: rex@aussie.com

Please check one of the following choices:

___ Yes

___ Yes with editorial comments as stated below

_X_ No for the technical reasons stated below

___ Abstain

Name: Gary E. Fisher
Affiliation: National Institute of Standards and Technology
_X_ Primary __ Alternate (please check one only)
Date: 12/17/1997

NIST is in favor of the ECMAScript specification becoming an international standard, but the ECMAScript DIS contains provisions in section 15.9, Date Objects, that cause NIST to cast its no vote for the following reasons:

BACKGROUND

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 agengies 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 implementors 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 implementors 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. Notwithstanding 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.