









## Implementation Loopholes In ECMAScript, 3rd Edition

ECMA-262 Section Number	Section Name	Section Text	Cross Ref to JScript Deviations	Notes	8/16/07 Meeting with Doug Crockford	Follow up
7.6	Identifiers	"implementations may allow additional legal identifier characters based on the category assignment from later versions of Unicode"		Need to update to reflect current state of Unicode	For ES 3.1 should adopt ES4 unicode proposal ( <a href="http://developer.mozilla.org/es4/proposals/update_unicode.html">http://developer.mozilla.org/es4/proposals/update_unicode.html</a> ) except for \uXXX\uXXX pairs	I think we need some further clarification of the pairs exclusion. On the surface the ES4 pairs rule seems reasonable but I believe the objection is that it may cause JSON incompatibilities. Doug, can you clarify??
7.8.5	Regular Expression Literals	 "An implementation may extend the regular expression constructor's grammar, but it should not extend the RegularExpressionBody and RegularExpressionFlags productions or the productions used by these productions."				
7.8.5	Regular Expression Literals / Semantics	"If the call to new RegExp generates an error, an implementation may, at its discretion, either report the error immediately while scanning the program, or it may defer the error until the regular expression literal is evaluated in the course of program execution"				
8.5	The Number Type	"external code might be able to detect a difference between various Non-a-Number values, but such behaviour is implementation-dependent"			Consider removing from ES3.1. Implementation level interaction with external code seems to be beyond the scope of this specification and hence doesn't need to be mentioned.	
8.6.2	Internal Properties and Methods	"Whether or not a native object can have a host object as its [[Prototype]] depends on the implementation."				

ECMA-262 Section Number	Section Name	Section Text	Cross Ref to JScript Deviations	Notes	8/16/07 Meeting with Doug Crockford	Follow up
10.1.1	Function Objects	"An implementation may also provide implementation-dependent internal functions that are not described in this specification. "		"Internal functions" are implementation artifacts. There isn't any particular reason they need to be documented.	Agreed 	
11.2.3	Function Calls	"Whether calling a host object can return a value of type Reference is implementation-dependent."			f() = v ??? Do any IE host objects do this? 	
11.4.3	The typeof Operator	typeof result for "host objects" is implementation dependent		Should be "implementation-defined" rather than "implementation-dependent"?	Agreed	
12.6.4	The for-in Statement	"The mechanics of enumerating the properties (step 5 in the first algorithm, step 6 in the second) is implementation dependent. The order of enumeration is defined by the object."	2.2			
13.2	Creating Function Objects	"If there is more than one object E satisfying these criteria, choose one at the implementation's discretion." ; "13. At the implementation's discretion, go to either step 2 or step 14." ; "Step 1 allows an implementation to optimise the common case of a function A that has a nested function B where B is not dependent on A. In this case the implementation is allowed to reuse the same object for B instead of creating a new one every time A is called. Step 13 makes this optimisation optional; an implementation that chooses not to implement it will go to step 2." ; plus additional vergabe about "joined" functions			We agreed that the concept of "joined" function objects should be eliminated from the 3.1 specification	

ECMA-262 Section Number	Section Name	Section Text	Cross Ref to JScript Deviations	Notes	8/16/07 Meeting with Doug Crockford	Follow up
15	Native ECMAScript Objects	"Unless otherwise specified in the description of a particular function, if a function or constructor described in this section is given more arguments than the function is specified to allow, the behaviour of the function or constructor is undefined. In particular, an implementation is permitted (but not required) to throw a TypeError exception in this case."		Rather than saying "is undefined" should say "is implementation defined" 	Agreed	
15.1	The Global Object	"The values of the [[Prototype]] and [[Class]] properties of the global object are implementation-dependent"				
15.1.2.2	parseInt (string , radix)	"When radix is 0 or undefined and the string's number begins with a 0 digit not followed by an x or X, then the implementation may, at its discretion, interpret the number either as being octal or as being decimal. Implementations are encouraged to interpret numbers in this case as being decimal." 	2.5			
15.1.2.2	parseInt (string , radix)	". (However, if R is 10 and Z contains more than 20 significant digits, every significant digit after the 20th may be replaced by a 0 digit, at the option of the implementation; and if R is not 2, 4, 8, 10, 16, or 32, then Result(16) may be an implementation-dependent approximation to the mathematical integer value that is represented by Z in radix-R notation.)"				
15.2.2.1	new Object ( [ value ] )	"4. If the value is a host object, then actions are taken and a result is returned in an implementation-dependent manner that may depend on the host object."		Should be "implementation-defined" rather than "implementation-dependent"?	Step 4 should be marked as deprecated. Implementations  should not define new semantics based upon passing a host object to the Object constructor	

ECMA-262 Section Number	Section Name	Section Text	Cross Ref to JScript Deviations	Notes	8/16/07 Meeting with Doug Crockford	Follow up
15.2.4.4	Object.prototype. valueOf ( )	"If the object is the result of calling the Object constructor with a host object (section 15.2.2.1), it is implementation-defined whether valueOf returns its this value or another value such as the host object originally passed to the constructor."			Should be deprecated along with step 4 of 15.2.2.1	
15.3.4.2	Function.prototype. toString ( )	"An implementation-dependent representation of the function is returned. This representation has the syntax of a FunctionDeclaration. Note in particular that the use and placement of white space, line terminators, and semicolons within the representation string is implementation-dependent."	2.6	Should be "implementation-defined" rather than "implementation-dependent"?	Agreed	
15.4.4.3	Array.prototype. toLocaleString ( )	"a separator string that has been derived in an implementation-defined locale-specific way"	2.7			
15.4.4.4- 15.4.4.13	Array.prototype.concat Array.prototype.join Array.prototype.pop Array.prototype.push Array.prototype.reverse Array.prototype.shift Array.prototype.slice Array.prototype.sort Array.prototype.splice Array.prototype.unshift	"Whether the XXX function can be applied successfully to a host object is implementation-dependent."		Should be "implementation-defined" rather than "implementation-dependent"?	Agreed	
15.4.4.11	Array.prototype.sort	"If comparefn is not undefined and is not a consistent comparison function for the elements of this array (see below), the behaviour of sort is implementation-defined. " ; "If there exist integers i and j and an object P such that all of the conditions below are satisfied then the behaviour of sort is implementation-defined:"			The default sort comparison function should be more clearly specified as a string comparison. Consider using ES4 verbage	 Doug, do you know what verbage we had in mind. I can't find any.

ECMA-262 Section Number	Section Name	Section Text	Cross Ref to JScript Deviations	Notes	8/16/07 Meeting with Doug Crockford	Follow up
15.5.4.9	String.prototype. localeCompare	"The two strings are compared in an implementation-defined fashion. " ; "The actual return values are left implementation-defined to permit implementers to encode additional information in the result value"				
15.5.4.11	String.prototype.replace	\$n: "If n>m, the result is implementation-defined." ; \$nn: "If nn>m, the result is implementation-defined"			Check what IE does, if it is reasonable, make it the spec. If not reasonable, consider Firefox,etc.	
15.7.4.2	Number.prototype.toString	"If radix is an integer from 2 to 36, but not 10, the result is a string, the choice of which is implementation-dependent."		Should be "implementation-defined" rather than "implementation-dependent"?	should define results for radices 2-36	the real issue seems to be relate to the value returned for non-integer value with radices other than 10. For example, IE and FF produce different results for (new Number(1.234)).toString(30). Do we really want to define this or is it better to maintain the status quo?
15.7.4.3	Number.prototype. toLocaleString	"This function is implementation-dependent, and it is permissible, but not encouraged, for it to return the same thing as toString."		Should be "implementation-defined" rather than "implementation-dependent"?	Agreed	
15.7.4.5-6	Number.prototype.toFixed Number.prototype. toExponential	"An implementation is permitted to extend the behaviour of XXX for values of fractionDigits less than 0 or greater than 20. In this case XXX would not necessarily throw RangeError for such values."			Need to determine if any of the major implementations actually do this and characterize any variation among implementations.	

ECMA-262 Section Number	Section Name	Section Text	Cross Ref to JScript Deviations	Notes	8/16/07 Meeting with Doug Crockford	Follow up
15.7.4.7	Number.prototype.toPrecision	"An implementation is permitted to extend the behaviour of toPrecision for values of precision less than 1 or greater than 21."			Need to determine if any of the major implementations actually do this and characterize any variation among implementations.	
15.8.2	Function properties of math object				URL of fdlibm is obsolete. Should be netlib.org/fdlibm	
15.9.1.8	Daylight Saving Time Adjustment	"An implementation of ECMAScript is expected to determine the daylight saving time algorithm"			This sentence doesn't add anything so it probably should be deleted.  All of sections 15.9.1.X which are attempting to specify the semantics of the time/date functions would benefit from a careful ready and potential cleanup.	
15.9.1.14	TimeClip (time)	"The point of step 3 is that an implementation is permitted a choice of internal representations of time values, for example as a 64-bit signed integer or as a 64-bit floating-point value. Depending on the implementation, this internal representation may or may not distinguish 0 and +0"			This is an internal function. Should probably force it to return +0.	
15.9.4.2	Date.parse (string)	"the value produced by Date.parse is implementation-dependent when given any string value that could not be produced in that implementation by the toString or toUTCString method."	2.1	Should be "implementation-defined" rather than "implementation-dependent"?	Specify the de facto IE data parsing syntax. Don't add ISO parsing to this function; instead add new ISO parsing/generation functions.	
15.9.4.3	Date.UTC	"When the UTC function is called with fewer than two arguments, the behaviour is implementation-dependent."		Should be "implementation-defined" rather than "implementation-dependent"?	Needs to specify ranges of arguments. Should allow a single argument (year)	

ECMA-262 Section Number	Section Name	Section Text	Cross Ref to JScript Deviations	Notes	8/16/07 Meeting with Doug Crockford	Follow up
15.9.5.2	Date.prototype.toString	"The contents of the string are implementation-dependent, but are intended to represent the Date in the current time zone in a convenient, human-readable form"	2.1	Should be "implementation-defined" rather than "implementation-dependent"?	All date method definitions need to have their argument ranges specified.	
15.9.5.3-7	Date.prototype.toString Date.prototype.toTimeString Date.prototype.toLocaleString Date.prototype.toLocaleDateString Date.prototype.toLocaleTimeString	"The contents of the string are implementation-dependent, but are intended to represent the "XXX" portion of the Date"		Should be "implementation-defined" rather than "implementation-dependent"?		
15.9.5.42	Date.prototype.toUTCString	"The contents of the string are implementation-dependent, but are intended to represent the Date in a convenient, human-readable form in UTC."		Should be "implementation-defined" rather than "implementation-dependent"?	Agreed	
15.10.4.1	new RegExp(pattern, flags)	"The source property of the newly constructed object is set to an implementation-defined string value in the form of a Pattern based on P."			Agreed	
15.11.4.3	Error.prototype.message	"The initial value of Error.prototype.message is an implementation-defined string."			should be define to be the empty string	
15.11.4.4	Error.prototype.toString	"Returns an implementation defined string."			change to follow mozilla	
15.11.7	NativeError Object Structure	"and in the implementation-defined message property of the prototype object."				
15.11.7.10	NativeError.prototype.message	"The initial value of the message property of the prototype for a given NativeError constructor is an implementation-defined string."				
16	Errors	Various allowances for implementation dependent error behavior (or lack there of) related to implementation dependent extensions				

ECMA-262 Section Number	Section Name	Section Text	Cross Ref to JScript Deviations	Notes	8/16/07 Meeting with Doug Crockford	Follow up
B.2	Additional Properties	"Some implementations of ECMAScript have included additional properties for some of the standard native objects. This non-normative annex suggests uniform semantics for such properties without making the properties or their semantics part of this standard."				