3APPENDIX D: RESOLUTION HISTORY

18 D.1 JANUARY 15, 1997

4D.1.1 White Space

Updated the White Space section to include form feed and vertical tab as white space.

5D.1.2 Keywords

Updated the Keywords section to exclude those keywords related to proposed extensions. Also updated this section to include the `delete` keyword which was missing.

6D.1.3 Future Reserved Words

Update the Future Reserved Words to only include keywords related to proposed extensions. We decided to remove words that had been only included as future reserved for Java compatibility purposes.

7D.1.4 Octal And Hex Escape Sequence Issue

Decided to support octal and hex notation. Since only two hex digits are used with hex notation, many unicode characters cannot be represented this way. Furthermore, we were not sure if the high 128 characters match up with unicode. (Removed open issue at bottom of section Once the exact MV for a numeric literal has been determined, it is then rounded to a value of the Number type. If the MV is 0, then the rounded value is +0; otherwise, the rounded value must be the number value for the MV (in the sense defined in section 8.4), unless the literal is a DecimalLiteral and the literal has more than 20 significant digits, in which case the number value may be any implementation-dependent approximation to the MV. A digit is significant if it is not part of an ExponentPart and (either it is not 0 or it is an important zero or there is no decimal point Ô.Õ in the literal). A digit 0 is an important zero if there is at least one important item to its left and at least one important item to its right within the literal. Any digit that is not 0 and is not part of an ExponentPart is an important item; a decimal point Ô.Õ is also an important item.)

The argument against was that these notations are redundant since any character can be represented using the unicode escape sequence. The arguments for were that hex and octal notation are convenient and simple and also that there is a language tradition to be upheld.

8D.1.5 ToPrimitive

Removed the erroneous note stating that errors are never generated as a result of calling ToPrimitive in the ToPrimitive section.

9D.1.6 Hex in ToNumber

We decided to allow hex in ToNumber but not octal. Looking at it from the user input source point of view, we decided that it was reasonable to use hex but not octal since it might be common to include leading zeros in a user input field. Furthermore we did not believe that the ability to use octal in data entry was desirable. (Removed open issue at the bottom of 5.3.1 ToNumber Applied to the String Type)

10D.1.7 Attributes of Declared Functions and Built-in Objects

We decided that built-in objects will have attributes { DontEnum } and that variables declared in global code will have empty attributes. (Updated the 6.1.1 Global Object section)

11D.1.8 The Grouping Operator

We decided that the grouping operator would return the result of GetValue() so that the result is never of type reference. (Updated the The Grouping Operator and removed the open issue at the bottom of this section)

12D.1.9 Prefix Increment and Decrement Operators

We decided to not to perform GetValue to the return value and thus leave the algorithm as is. (removed the open issue at the bottom of the Prefix Increment Operator)

13D.1.10 Unary Plus

We decided to leave the algorithm for unary plus alone and continue to call GetValue() and ToNumber() after evaluating the unary expression which guarantees a numeric result as opposed to only evaluating the unary expression which would not guarantee a numeric result. (Updated the Prefix Decrement Operator section)

14D.1.11 Multiplicative Operators

Updated step nine in the Multiplicative Operators section to refer to three new sections 7.41, 7.42 and 7.43 which define the behavior of `*`, `/` and `%`.

15D.1.12 Additive Operators

Updated step 11 in 7.5.1 and step 10 in 7.5.2 to refer to a new section 7.5.3 which define the behavior of `+` and `-`.

16D.1.13 Left Shift Operator

We decided to leave the algorithm for left shift as is, which converts the left operand using ToInt32 rather than ToUint32. Although an unsigned conversion might be arguably preferred, we decided to continue to convert to signed, as we can always add a new operator (<<<) to accomplish an unsigned shift. (Removed the open issue at the bottom of The Left Shift Operator (<<))

17D.1.14 Binary Bitwise Operators

We decided to leave the algorithm for the binary bitwise operators as is, which uses signed conversion on the GetValue of its operands. (Removed the open issue at the bottom of Binary Bitwise Operators)

18D.1.15 Conditional Operator (`? :`)

We decided to leave the algorithm for the conditional operator as is, which performs a GetValue on the result before returning. Current implementations do not do this. (Removed the open issue at the bottom of Conditional Operator (?:))

19D.1.16 Simple Assignment

We decided to leave the algorithm for simple assignment as is. (Removed the open issue at the bottom of Simple Assignment (=))

20D.1.17 The `for..in` Statement

We decided to impose no restrictions on Expression1. (Removed the first open issue at the bottom of ISSUE: Finish the necessary changes for the other three forms of iteration statement for value returns.)

21D.1.18 The `return` Statement

We decided to not generate an error if one return statement in a function returns a value and another return in the same function does not return a value. (Removed the first open issue at the bottom of the The return Statement The second issue at the bottom of this section has been moved to)

22D.1.19 New Proposed Extensions

Sections B.10 Preprocessor, B.11 The do..while Statement and B.12 Binary Object were added.

19D.2 JANUARY 24, 1997

23D.2.1 End Of Source

Updated Error: Reference source not found section to describe the end of source token as logical rather than physical \u0000 since strings may contain embedded \u0000 characters.

24D.2.2 Future Reserved Words

Updated Future Reserved Words section to include the word **do** and removed the footnotes indicating the origin of the proposed keywords.

25D.2.3 White Space

Updated White Space section. Updated the lexical production for SimpleWhiteSpace to include <VT> and <FF> (already mentioned in the white table above).

26D.2.4 Comments

Added new issue to 3.2 regarding nested comments.

27D.2.5 Identifiers

Updated section 3.3.2 to correctly state what is an allowable first character in an identifier.

28D.2.6 Numeric Literals

Updated section 3.3.4.3 Numeric Literals to disallow leading zeros in floating point literals.

29D.2.7 String Literals

Updated the table describing the set of character escape characters in section Once the exact MV for a numeric literal has been determined, it is then rounded to a value of the Number type. If the MV is 0, then the rounded value is +0; otherwise, the rounded value must be the number value for the MV (in the sense defined in section 8.4), unless the literal is a DecimalLiteral and the literal has more than 20 significant digits, in which case the number value may be any implementation-dependent approximation to the MV. A digit is significant if it is not part of an ExponentPart and (either it is not 0 or it is an important zero or there is no decimal point Ô.Õ in the literal). A digit 0 is an important zero if there is at least one important item to its left and at least one important item to its right within the literal. Any digit that is not 0 and is not part of an ExponentPart is an important item; a decimal point Ô.Õ is also an important item., to include a new column indicating the unicode value. Also added a new issue to the end of this section.

30D.2.8 Automatic Semicolon Insertion

Added two new issues to the end of .

31D.2.9 Property Attributes

Renamed *Permanent* to *DontDelete* in the property attributes table in the Property Attributes section.

32D.2.10 ToPrimitive

Reworded section ToPrimitive to better describe the optional hint *PreferredType*.

33D.2.11 ToNumber

Updated section ToNumber. Added Hint Number in call to ToPrimitive. Also added new issue to the end of this section.

34D.2.12 White Space

Updated section ToNumber Applied to the String Type Updated the lexical production for SimpleWhiteSpace to include <VT> and <FF>.

35D.2.13 ToNumber Applied to the String Type

Updated section 5.3.1, ToNumber Applied to the String Type. Reworked lexical productions to be similar to those used in section, . The difference between string numeric literals and numeric literals is that string numeric literals do not allow octal notation and do allow leading zeros.

36D.2.14 ToString

Updated section Note that ToUint32 maps -0 to +0.. Added Hint String in call to ToPrimitive.

37D.2.15 Postfix Increment and Decrement Operators

Updated section Error: Reference source not found. Updated the algorithm to return Result(3) (the result of converting ToNumber), rather than (Result(2).

38D.2.16 The `typeof` operator

Added a new issue at the end of section The typeof Operator.

39D.2.17 Prefix Increment and Decrement Operators

Removed extraneous calls to ToPrimitive from the algorithm in section Prefix Increment Operator.

40D.2.18 Multiplicative Operators

Remove step 7 in the algorithm in section 7.4 (either operand NaN) and added a new rule to 7.4.1 and 7.4.2 to reiterate what was in the old step.

41D.2.19 The Subtraction Operator

Removed extraneous calls to ToPrimitive from the algorithm in section 7.5.2.

42D.2.20 The Subtraction Operator

Remove the old step 9 in the algorithm in section 7.5.2 (either operand NaN) and added a new rule to section 7.5.3 to reiterate what was in the old step.

43D.2.21 Applying the Additive Operators (+, -)

Update the last rule in section 7.5.3 to clearly state that operands mentioned in the final sentence must be numeric.

44D.2.22 Equality Operators

Moved the Semantic discussion at the beginning of 7.8 to the discussion section at the end of 7.8

45D.2.23 ToPrimitive Usage

Added issue at the end of sections 7.5.1 and 7,7.

46D.2.24 Binary Logical Operators

Added issue at the end of 7.10.

20D.3 January 31, 1997

47D.3.1 MultiLineComment

Updated the lexical production *MultiLineComment* in section Comments, to allow empty multi-line comments. Also removed the issue at the end of this section regarding nested mutli-line comments. The *MultiLineComment* production continues to disallow multi-line comments.

48D.3.2 String Literals

Removed open issue at the end of section Once the exact MV for a numeric literal has been determined, it is then rounded to a value of the Number type. If the MV is 0, then the rounded value is +0; otherwise, the rounded value must be the number value for the MV (in the sense defined in section 8.4), unless the literal is a DecimalLiteral and the literal has more than 20 significant digits, in which case the number value may be any implementation-dependent approximation to the MV. A digit is significant if it is not part of an ExponentPart and (either it is not 0 or it is an important zero or there is no decimal point Ô.Õ in the literal). A digit 0 is an important zero if there is at least one important item to its left and at least one important item to its right within the literal. Any digit that is not 0 and is not part of an ExponentPart is an important item; a decimal point Ô.Õ is also an important item. which stated that the maximum string constant supported must be at least 32000 characters long.

49D.3.3 Automatic Semicolon Insertion

Updated section , to include rules governing parsing the **`for`** statement and dealing with postfix **`++`** and postfix **`--`** tokens.

50D.3.4 The Number Type

Updated the description in section The String Type.

51D.3.5 Put with Explicit Access Mode

Update section 4.5.2.3, Put with Explicit Access Mode to include looking in the prototype object for access violations.

52D.3.6 Put with Implicit Access Mode

Update section 4.5.2.4, Put with Implicit Access Mode to include looking in the prototype object for access violations.

53D.3.7 The String type

Updated the description in section 4.6, The String Type.

54D.3.8 ToNumber

Updated section 5.3, ToNumber to return a **`NaN`** for an input type of **`Null`**.

55D.3.9 ToNumber Applied to the String Type

Updated the lexical production for SimpleWhiteSpace in section 5.3.1 to include <CR> and <LF>. Also updated the lexical productions StrFloatingPointLiteral and StrIntegerLiteral to allow signs.

56D.3.10 ToInt32

Updated description in section 5.5, ToInt32: (signed 32 bit integer) to tentatively use Guy’s Conversion modulo 2^32 algorithm.

57D.3.11 ToUint32

Updated description in section ToUint32: (unsigned 32 bit integer) to tentatively use Guy’s Conversion modulo 2^32 algorithm.

58D.3.12 Execution Contexts (Variables)

Section 6 (Variables) replaced by new section (Execution Contexts).

59D.3.13 Function Calls

Swapped steps 2 and 3 in section 7.2.4, Function Calls.

60D.3.14 The `typeof` Operator

Updated the table in section The typeof Operator to specify the result when the input type is an external object. Removed related open issue at the end of this section.

61D.3.15 Applying the `%` Operator

Removed step 7 in the algorithm in section 7.4.(either operand NaN) and added a new rule to 7.4.3 to reiterate what was in the old step.

62D.3.16 The Addition Operator (+)

Added the hint Number in the calls to ToPrimitive in section 7.5.1, The Addition Operator (+). Removed related open issue at the end of this section.

63D.3.17 Relational Operators

Added the hint Number in the calls to ToPrimitive in section 7.7, Relational Operators. Removed related open issue at the end of this section.

64D.3.18 Conditional Operator (`?:`)

Updated the syntactic production, ConditionalExpression, in section Conditional Operator (?:)

65D.3.19 Compound Assignment (`op=`)

Swapped steps 2 and 3 in section 7.12.2, Compound Assignment (**`op=`**)

21D.4 February 21, 1997

66D.4.1 Unicode Escape Sequences

Rewrote section Error: Reference source not found to reflect the restriction that non-ASCII Unicode characters may appear only within comments and string literals. Moved the description of Unicode escape sequences to Once the exact MV for a numeric literal has been determined, it is then rounded to a value of the Number type. If the MV is 0, then the rounded value is +0; otherwise, the rounded value must be the number value for the MV (in the sense defined in section 8.4), unless the literal is a DecimalLiteral and the literal has more than 20 significant digits, in which case the number value may be any implementation-dependent approximation to the MV. A digit is significant if it is not part of an ExponentPart and (either it is not 0 or it is an important zero or there is no decimal point Ô.Õ in the literal). A digit 0 is an important zero if there is at least one important item to its left and at least one important item to its right within the literal. Any digit that is not 0 and is not part of an ExponentPart is an important item; a decimal point Ô.Õ is also an important item..

67D.4.2 Future Reserved Words

Added **`import`** and **`super`** to table in Future Reserved Words.

68D.4.3 Automatic Semicolon Insertion

Rewrote the rules for semicolon insertion in section to incorporate the rule that a semicolon is not inserted if it would be treated as an empty statement. Also, broke out the empty statement as a separate kind of statement for expository purposes in section The production Initializer : = AssignmentExpression is evaluated as follows:.

69D.4.4 The Number Type

Corrected formatting of formulae in section The String Type.

70D.4.5 NotImplicit and NotExplicit Property Attributes Deleted

The NotImplicit and NotExplicit property attributes were deleted from the table in section Property Attributes. Many changes throughout the rest of chapter 4 to reflect this deletion. Also, the [[TestPutExplicit]] helper method was renamed [[CanPut]].

71D.4.6 ToInt32 and ToUint32

Corrected formatting of formulae in sectionToInt32: (signed 32 bit integer) and section ToUint32: (unsigned 32 bit integer). Also, change the discarding of the fractional part to truncate toward zero rather than using a simple floor operation.

Correct an error in the descriptions by adding a new step 4 to each one, which makes sure that if the input is negative zero, the output is positive zero.

72D.4.7 Grouping Operator

Delete step 2 from section The Grouping Operator. Parentheses no longer force dereferencing.

73D.4.8 Shift Expressions

Correct the grammar for *ShiftExpression* by adding *AdditiveExpression* as an alternative in section Bitwise Shift Operators.

74D.4.9 Conversion Rules for Relational Operators

Updated description in section Relational Operators so that lexicographic string ordering is used only if both operands become strings when converted to primitive type; if one is a string and one is a number, then numeric ordering is used. Thus relational operators differ from the + operator, which, if one operand is a string and one is a number, performs string concatenation rather than addition.

75D.4.10 && and || Semantics

Updated description in section Binary Logical Operators so that `&&` and `||` have PERL-like semantics; that is, the result of `1||2` is `1`, not true, and the result of `0||ÓHelloÓ` is `ÒHelloÓ`.

76D.4.11 Conditional Operator

Updated section Conditional Operator (?:) to reflect the change that the second and third subexpressions should each be *AssignmentExpression*.

77D.4.12 Assignment Operators

Updated section Assignment Operators to reflect the change that the left-hand side of an assignment should be a *PostfixExpression*. Also change two occurrences in subsections of SetVal to PutValue.

78D.4.13 Syntax of Class Statement

Updated section B.1 The Class Statement1 to allow the parentheses in a class declaration to be optional.

79D.4.14 Syntax of Try Statement

Updated section B.2.1 The try Statement1 to require the body of a `catch` or `finally` clause to be a *Block*.

22D.5 February 27, 1997

80D.5.1 Grammar Notation

Big rewrite of section Syntactic and Lexical Grammars to make the description of grammar notation more detailed and rigorous. Is this okay? (Much of the text was borrowed, in form at least, from the

Java Language Specification.) The notation is still a bit inconsistent throughout the document (example: ÒexceptÓ versus Òbut notÓ), and should be made consistent within itself and with section Syntactic and Lexical Grammars.

Also decided to call out the grammar in Chapter 5 as a separate grammar and use triple colons on its productions.

Restructured some of the grammar in Chapter 3 to make it a bit more readable. Is this okay?

81D.5.2 End of Medium Character Is No Longer WhiteSpace

Deleted character \u0019 (End of Medium) from the table in section White Space, and deleted <EOM> as an alternative for SimpleWhiteSpace in that same section. Also deleted <EOM> as an alternative for StrWhiteSpaceChar in section ToNumber Applied to the String Type. These changes reflect the decision that neither \u0019 (End of Medium, mistakenly also referred to in previous drafts of this document as ^Z) nor \u001A (Substitute, which really is ^Z) shall be considered whitespace in an ECMAScript program. It is expected that host environments will filter any ^Z character that might occur at the end of the host environmentÕs representation of an ECMASCript program.

82D.5.3 Meaning of Null Literal

Added to section Null Literals a discussion of the meaning of a null literal.

83D.5.4 Meaning of Boolean Literals

Added to section Semantics a discussion of the meaning of a boolean literal.

84D.5.5 Meaning of Numeric Literals

Added to section a discussion of the meaning of a numeric literal. It does not yet address the restriction to 19 significant digits. Is this the style of description we want?

85D.5.6 Automatic Semicolon Insertion

Updated description of automatic semicolon insertion in section . Systematically replaced the word ÒinjectedÓ with ÒinsertedÓ. Invented a new theory of Òrestricted productionsÓ to explain in a general way why the parser inserts semicolons in places where there would otherwise be a valid parse without a semicolon. Added more examples and advice. Also modified productions in sections Left-Hand-Side Expressions and The return Statement to indicate the restrictions explicitly.

86D.5.7 The Number Type

Updated section The String Type to provide explanations of those large numbers as sums and differences of powers of two.

87D.5.8 ToString on Numbers

Updated section ToString Applied to the Number Type have a draft specification of how this conversion ought to be done. This needs to be reviewed. This version requires that, when the number has a nonzero fractional part, the output must be correctly rounded and produce no more digits than necessary for the fractional part. Added a bibliographic reference to the paper and code of David M. Gay on this subject.

88D.5.9 New Operator

Updated description in section The production CallExpression : MemberExpression [Expression] is evaluated in exactly the same manner, except that the contained CallExpression is evaluated in step 1. to describe the case where no argument list is provided. This needs to be reviewed.

89D.5.10 Delete Operator

Updated description in section The delete Operator to reflect decision that this operator shall return a boolean value; the value **`true`** indicates that, after the operation, the object is guaranteed not to have the specified property.

90D.5.11 == Semantics

Updated section If Result(2) is a prefix of Result (1), return false. (A string value p is a prefix of string value q if q can be the result of concatenating p and some other string r. Note that any string is a prefix of itself, because r may be the empty string.) so that (a) **`null`** and **`undefined`** are considered equal, and (b) when a number meets a string, the number is converted to a string and then string equality is used.

91D.5.12 && and || Semantics

Updated description in section Binary Logical Operators to delete step 7 for eachoperator (the result of this step was no longer used).

92D.5.13 Separate Productions for Continue, Break, Return

To make certain kinds of cross-reference in the document simpler, I broke out the continue, break, and return statements into separate grammatical productions, eliminating the production for *ControlFlowStatement* (which was something of a misnomer anyway, and other statements also result in (structured) control flow.

93D.5.14 Dead Code Is Not Protected from Compile-Time Analysis

Added text to chapter 12 (Errors).

23D.6 MARCH 6, 1997

94D.6.1 Reformatted the Entire Document

I order to make future revisions easier and to take better advantage of the desktop-publishing capabilities of Word, the entire document was reformatted using some newly defined Word styles. Heading numbering was turned on to facilitate automatic numbering of headings in the main text (sections of the appendices are still numbered manually, using new styles Appendix Heading 1, Appendix Heading 2, and Appendix Heading 3). A new style Algorithm is used for algorithmic steps; in some cases, the last step should be styled with AlgorithmLast to provide extra vertical space after the last step.

Added a style called MathSpecialCase (generates bullet lists for now).

The title page now uses styles Title and Subtitle, which were modified to use apropriate fonts and paragraph spacing.

Extraneous tab characters and multiple spaces were deleted from all headings.

The paragraph spacing of Normal, the various headings, Algorithm, AlgorithmLast, SyntaxRule, and SyntaxDefinition were adjusted so that the correct vertical space is inserted automatically. All blank paragraphs in the document were deleted.

The index and all index entries were deleted. Sorry, but they were somehow interfering with other formatting, and the index entries were terribly incomplete anyway. If we have time to do a good index, entries can be added semi-systematically.

The document was divided into three of what Word calls ÒsectionsÓ so that the pages of the Table of Contents could be numbered with the customary roman numerals, with the main text starting on page 1.

All the revisions listed in this item were accepted and the change bars reset before the following items were entered, so that all the changes of this item would not clutter the manuscript.

95D.6.2 Designed a Section Outline for Chapter 11

Filled in nearly all necessary section headings for Chapter 1 for describing Object, Function, Array, String, Boolean, Number, and Math and all their properties and methods. Added a fair amount of boilerplate text.

96D.6.3 Defined Math Functions

Added complete definitions for all properties in the Math object, following the example of C9X for the treatment of IEEE 754 special cases.

24D.7 March 10, 1997

97D.7.1 Added Definition of ÒThe Number Value for xÓ

In section 8.4, the phrase Òthe number value for *x*Ó is now defined. It encapsulates the entire IEEE 754 process for converting any nonzero mathematical value to a representable value by using round-to-nearest mode. This phrase is of great use in Chapter 15 and elsewhere.

Also corrected two typos in this section: −1073 replaced by −1074, and 2^{53} replaced by 2^{52}.

98D.7.2 atan and atan2 May Use Implementation-Dependent Values for π, etc.

It was decided at the phone meeting that when **`Math.atan`**, for example, is supposed to return π/2, it need not return exactly one-half the initial value of **`Math.pi`**, but may produce an approximation. The motivation is to allow implementors the use of whatever C math library is present on the hardware platform at hand, whether or not it conforms to the high quality standards of, for example, the C9X proposal.

99D.7.3 Improved Discussion of Input Stream for Syntactic Grammar

Text added to section 5.1 to better explain the handling of whitespace, comments, and line terminators, and the fact that line terminators become part of the input stream for the syntatic grammar. Also corrected a type in section 5.1.5 where the phrase Ò[no *LineTerminator* here]Ó had been inadvertently omitted.

100D.7.4 Improved Treatment of LineTerminator in Lexical Grammar

Eliminated the mythical <EOS> character. As a result, *LineEnd* is not needed either. The trick is not to include LineEnd (or LineTerminator) as part of the grammar of a single-line comment. This works out better, because a single-line comment still runs to the end of the line (as dictated by the longest-token-possible rule), but it doesnÕt swallow the *LineTerminator*, so it doesnÕt affect automatic semicolon insertion. (That the previous production did swallow the *LineTerminator* was thus a bug.)

The section on whitespace has been divided into two sections, one on *WhiteSpace* (formerly called *SimpleWhiteSpace*) and one on Line Terminators.

THIS CHANGE REQUIRES REVIEW.

101D.7.5 Clarify Behavior of Unicode Escape Sequences

In Chapter 6, clarify that a Unicode escape sequence such as \u000D does not produce a carriage return that could end a single-line comment, for example.

102D.7.6 Add Careful Description of the String Value of a String Literal

In imitation of the text already present describing the value of a numeric literal, text was added to section 7.7.4 to describe carefully the exact sequence of characters represented by a string literal. In the process, missing productions for *DoubleStringCharacters* and *SingleStringCharacters* were added, and the redundant defintions of *HexDigit* and *OctalDigit* were removed. Also dealt with an open issue by emphasizing that a *LineTerminator* may not appear within a string literal.

103D.7.7 Description of Identifiers Reworded

Improvements to the wording in section 7.5. Also repaired a typo (capital **`I`** replaced by lowercase **`I`**).

104D.7.8 Table of Punctuators Corrected

Underscore replaced by + operator in table in section 7.6.

105D.7.9 Improved Descriptions of ToInt32 and ToUint32

Step 5 of the algorithms in sections 9.5 and 9.6 have been clarified to use a mathematical description rather than fragments of code .

106D.7.10 Changes to ToString Applied to the Number Type

See section 9.8.1. Negative zero now produces **"0"**., not **"-0"**.. Integers less than 10^{20} shall print without decimal points. Values less than 1 but not less than 10^{-6} will not require scientific notation.

107D.7.11 Revised Syntax for NewExpression and MemberExpression

Made the changes to section 11.2 as suggested by Shon, eliminating *NewCallExpression* and providing a pleasing symmetry in which the number of **new** operators can exceed or fall short of the number of argument lists.

108D.7.12 Clarify Multiplicative and Additive Operators

In section 11.5.1, describe the multiplication of infinity by infinity.

In section 11.5.2, describe the division of infinity by zero.

In section 11.5.3, better describe the remainder of a zero by a finite number.

In section 11.6, better describe the sum of two zeros and the sum of finite numbers of same magnitude and opposite sign.

109D.7.13 Addition Operator No Longer Gives Hint Number

When the addition operator **+** calls ToPrimitive, it no longer gives hint Number. Note that all built-in objects respond to ToPrimitive without a hint as if hint Number were given, so thius change affects only external objects.

110D.7.14 Correct Description of Relational Operators

Miscellaneous small corrections.

111D.7.15 Assignment Operator LHS Must Be PostfixExpression

Change four occurrences of *UnaryExpression* to *PostfixExpression* in section 11.13.

112D.7.16 Changes to For-in Loops

Without **var**, the expression before **in** must be a *PostfixExpression* (as for an assignment),

With **var**, an optional *Initializer* is permitted after the *Identifier*.

A For-In loop enumerates not only properties of the given object itself, but also properties of its prototype, and so on, recursively.

ISSUE: Are shadowed properties of the prototype enumerated?

113D.7.17 Break and Continue Must Occur within While or For Loop

Added text to sections 12.6 and 12.7 to require **break** and **continue** to appear within loop statements.

25D.8 MARCH 12, 1997

114D.8.1 Added Overview Chapter

Added a chapter at the beginning as a placeholder for introductory exposition.

115D.8.2 More Exposition about Internal Properties

Renamed section 8.6.2 from ÒProperty AccessÓ to the more general ÒInternal Propeties and MethodsÓ.

Added properties [[Class]], [[Value]], [[CanPut]], and [[DefaultValue]] to the table so as to complete the list.

Added some discussion of these internal properties.

116D.8.3 Date Object

Added the Date object to chapter 15 and method descriptions, etc.

117D.8.4 Array, String, Boolean, Number Objects

Tons of work in chapter 15 to add method descriptions, etc.

118D.8.5 Math Object

Corrections to atan2 and floor.

26D.9 MARCH 24, 1997

119D.9.1 Numeric Literals

Revamped the grammar for numeric literals to simplify it (per ShonÕs suggestion) and added prose indicating that the number value for a decimal literal need only be an implementation-dependent approximation is there are more than 20 significant digits. In the process, the words Òfloating-pointÓ disappear from the grammar; floating-point literals are merely one kind of decimal literal.

120D.9.2 String Numeric Literals

Revamped the grammar for string numeric literals and added prose indicating that the number value for a decimal literal need only be an implementation-dependent approximation is there are more than 20 significant digits. In the process, the words Òfloating-pointÓ disappear from the grammar; floating-point literals are merely one kind of decimal literal.

Also added the text describing how to calculate a mathematical value for a string numeric literal. The details are a bit different from those for ordinary numeric literals.

27D.9.3 Prefix and Postfix Increment and Decrement Operators

Revise the grammar (for expository reasons), restructure the sections, and revise the algorithms for the `++` and `--` operators to be more precise.

28D.9.4 Left-Hand-Side Expressions

Revise the grammar for *PostfixExpression* so that uses of the postfix **++** and **--** operators cannot occur to the left of a **.** or **=**, for example. Now only a *LeftHandSideExpression* may occur on the left-hand side of an assignment or **`in`** keyword. Also updated the list of restricted productions accordingly in the description of automatic semicolon insertion (section 7.8).

29D.9.5 Reference Type

Revise description of the internal reference type (section 8.7).

30D.9.6 Infinities and Zeros

Decided to use the forms **NaN**, **positive zero**, **negative zero**, **positive infinity**, and **negative infinity** (Times bold, no caps except for **NaN**) consistently throughout the document to refer to those quantities. (Overridden by D.10.2.)

31D.9.7 Miscellaneous Small Corrections

Among the small corrections is the deletion of step 10, which was redundant, in the algorithm for the addition operator (section 11.6.1).

32D.10 MARCH 27, 1997

121D.10.1 Corrections to [[CanPut]] and [[HasProperty]]

Allow for the possibility that the [[Prototype]] is not implemented or has an undefined or primitive value.

122D.10.2 Discussion of Number Type

Explicitly introduce +∞, **−∞**, **+0**, and **−0** as symbols used for expository purposes in this specification, and briefly point out the program expressions `NaN`, `+Infinity`, `-Infinity`, `+0`, and `-0`.

Then override D.9.6 to use the symbols in preference to **positive zero**, **negative zero**, **positive infinity**, and **negative infinity** in most places, as they have turned out to be visually clumsy..

123D.10.3 Infinity and NaN

Add properties `NaN` (initial value is NaN) and `Infinity` (initial value is) to the global object.

Specify that $Sign_{opt}$ `Infinity` be recognized when ToNumber is applied to a string.

WhileI am at it, clarify the process fo converting a mathematical value (MV) to a rounded value of Number type, both for numeric literals and for string numeric literals.

124D.10.4 charCodeAt and String.fromCharCode

Add `charCodeAt` method for String objects and `String.fromCharCode` function to the String object.. In support of the description of `String.fromCharCode`, add the ToUint16 abstract operator.

125D.10.5 Last fraction digit from ToString applied to a number

Added discussion of the rule that if x is a number, ToNumber(ToString(x)) must be the same as x.

126D.10.6 Multi-line comment containing line terminator treated as line terminator

In section 5.1.2, added text to say that a multi-line comment is simply discarded if it contains no line terminator; but if a multi-line comment contains one or more line terminators, then it is replaced by a single line terminator, which becomes part of the stream of input elements for the syntactic grammar.

127D.10.7 Automatic semicolon insertion at end of source

Reworked the description of automatic semicolon insertion yet again. Handle end of course as a special case in section 7.8; as a consequence, it is not necessary to specially append a line terminator to the input stream in section 5.1.4.

128D.10.8 Added proposed extension for labelled break and continue

New section B.13 proposes the use of labels as in Java to allow transfer to other than the innermost containing loop.

129D.10.9 Lowercase ÒeÓ for scientific notation in ToString of a number

A lowercase ÒeÓ shall be used, not uppercase ÒEÓ (section 9.8.1).

130D.10.10 Evaluation of argument lists

Added algorithmic explanation of the evaluation of argument lists (section 11.2.4). To this end, invented yet another fictitious expository data type, List (section 8.8).

131D.10.11 For ToPrimitive of native objects, no hint is same as hint Number

Added a helpful note to section 11.6.1.

132D.10.12 Major overhaul of equality and relational operators

Revised descriptions to make them more precise, especially about **NaN**.

133D.10.13 String type

Moved the section on the String type and augmented its description (section 8.4).

33D.11 April 9, 1997

134D.11.1 Added mathematical operators to notation section

Added definitions of sign, abs, floor, modulo to section 5.2.

135D.11.2 Added overview text

Added overview text from Richard Gabriel. Renumbered sections so that Scope, Conformance, and Normative References come first (ECMA format).

136D.11.3 Added Date stuff

Added material provided by Shon to section 15.9 and did a lot of reformatting.

137D.11.4 Lots of work on native objects chapter

In particular, corrected lots of typos per Shon and added material about Object and Function.

34D.12 April 14, 1997

138D.12.1 Lots of corrections

Finished all my outstandiong corrections to the native objects chapter.

(Still to come: work on the functions and program chapters and on function invocation, including dynamic binding of the **`arguments`** property.)

139D.12.2 Reworked Date type

Provided a more mathematical treatment, relying less on invoking other methods.