Errata, Issues, and Proposed Changes
ECMAScript 5 Candidate Specification

July 27, 2009 Revision

Revised/added sections July 27: 10.2.1, 10.2.1.1.2, 10.2.1.1.5-8, 10.2.1.2.2, 10.2.1.2.5-6, 10.5, 11.4.1, 15.1.3, 15.10, 15.10.7, A1, A7

Revised/added sections July 16: 7.3, 7.5, 7.5.1-7.5.3->7.6.1-7.6.1.2 10.5(really 10.6), 12.14, 13.2, 15.3.3, 15.3.4.3, 15.3.4.4, 15.3.4.5, 15.3.5, 15.3.4.5, 15.5.4.14, 15.9.1.5, A1, C

Revised/added sections July 9: 5.1.2-5, 7.2, 7.8.5, 8.6.2, 8.7.1-2 8.10, 10.5/10.6, 10.6/10.5, 11.1.1, 11.1.4, 11.2.4, 11.3.1, 11.3.2, 11.4.1, 11.4.3, , 11.4.4, 11.4.5, 11.13.1, 11.13.2, 12.6, 12.10, 13, 13.2.1, 14, 15, 15.3.2.1, 15.3.4.5, 15.4, 15.4.3.2, 15.4.2.1, 15.4.2.2, 15.4.4.14, 15.4.4.15, 15.4.4.16-15.4.4.22, 15.4.5.1, 15.9.5.44, 15.10.7.5, 15.12.3, C

Revised/added sections June 17: 5.1.5, 6, 7.1, 7.2, 7.5, 7.5, 8.6.2, 8.7.2, 8.12.8, 11.1.5, 11.4.1, 12.2.1, 12.14.1, 13, 13.1, 14, 14.1, 15, 15.1.3, 15.4, 15.4.4.4, 15.4.4.14, 15.4.4.15, 15.4.4.11, 15.4.5.1, 15.9.5.43, 15.11.6.1, 15.12, 15.12.1.1, 15.12.3, 16, A.1, D

Revised/added sections May 21: 4.3.27, 5.1.5, 7.5.3, 7.6, 8.6, 8.6.1, 8.7.2, 12.14.1, 14.1

4.2 Language Overview.

Issue: The explanation of a method in the second paragraph is circular. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: this is ancient text but can be easily fix as:

> Properties are containers that hold other objects, ***primitive values***, or ***functions***. A primitive value is a member of one of the following built-in types: **Undefined**, **Null**, **Boolean**, **Number**, and **String**; an object is a member of the remaining built-in type **Object**; and a function is a callable object. A function that is associated with an object via a property is a ***method***.

4.2.1 Objects

Issue: In "… constructors which create objects by executing code that allocates storage for the objects and initializes …", Javascript code does not express explicit allocation, so I suggest deleting "allocates storage for the objects and". https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html , Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Proposed resolution: Revise paragraph:

> ECMAScript does not use classes such as those in C++, Smalltalk, or Java. Instead objects may be created in various ways including via a literal notation or via ***constructors*** which create objects

and then execute code that nitializes all or part of them by assigning initial values to their properties. Each constructor is a function that has a property named "`prototype`" …

4.3 Definitions

Issue: The definitions in the subsections that follow would be enhanced by cross references to the sections that provide normative details. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Agree with point, will consider on a time available basis. Contrary to what the introductory paragraph says, some of these definitions are in fact the normative definition of the term.

4.3.6 Native Object

Issue: The definition of a native object confuses me. It requires such an object to be "supplied by an implementation" but allows it to be "constructed during the course of execution of an ECMAScript program". Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Revise first sentence as:

> A ***native object*** is any object in an ECMAScript implementation whose semantics are fully defined by this specification rather than by the host environment. …

4.3.9-12 Primitive value/type Definitions

Issue: it is not explicitly stated that the undefined value of 4.3.9 is the one member of type Undefined referred to in 4.3.10. Ditto for 4.3.11 and 4.3.12. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Agree with point, will consider on a time available basis.

4.3.11 Null Value

Issue: this paragraph uses "reference", which is not consistent with the "properties are containers" metaphor of 4.2. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Replace definition with:

> The ***null value*** is a primitive value that represents the intentional absence of any other value.

4.3.15 Boolean Object

Issue: this is the first time that "instance of" appears. I think it would be useful to define it somewhere in 4.3. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Agree with point, will consider on a time available basis. Also see item for 4.3.18.

4.3.18 String Object

Issue: 4.3.18: Confusing because it basically says “a string object is an instance of the string object”. Perhaps when using “instance of” you could use the word “constructor” instead of object. This would be a change throughout. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: replace “object” with “constructor in this context and similar ones in 4.3.15 and 4.3.21

4.3.26 Property

Issue: last line implies that functions are not objects. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: It add “object” after first occurrence of “function”.

4.3.27 Method.

Issue: What does it mean to be “called as a method”, asked by Waldemar about section 8.6.1

Resolution: Add sentence to definition of Method:

> When a function is called as a method of an object, the object is passed to the function as its **this** value.

5.1.2-5.1.5 xxx Grammars

Issue: the uses of “Unicode character” in this section are inconsistent with the conventions established in section 6.

Resolution: drop “Unicode” and reference section 6.

5.1.5 The Use Strict Directive Grammar

Issue: Waldemar things that the productions of this grammar should use “:::” instead of “:” https://mail.mozilla.org/pipermail/es5-discuss/2009-May/002572.html Allen thinks it is correct the way it is written https://mail.mozilla.org/pipermail/es5-discuss/2009-May/002590.html Also see items for 14.1

Resolution: May F2f: : A grammar will not be used to describe directive. Delete section, renumber 5.1.6, 5.1.7 Also see changes to 14.1

5.1.7 Grammar Notation

Issue: middle of the 1st paragraph: “All nonterminal characters specified in this way…” Shouldn’t that be “All *terminal* characters”? Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Replace “all nonterminal characters” with “All symbol characters”

Issue: near the bottom of page 9: “immediately following input terminal”. Wouldn’t “token” be a clearer word than “terminal” here? Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: agreed, change “terminal” to “token”

Issue with: “For convenience, the set can also be written as a nonterminal, in which case it represents the set of all terminals to which that nonterminal could expand.” Since a nonterminal can expand into a sequence of terminals, this can be misread as including any terminal in any possible expansion, even if it can’t be an initial terminal. In practice, I believe this shorthand is only used for one terminal deep nonterminals. Not sure what is the best way to clarify. Perhaps simply “… set of all initial terminals to …”. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002430.html

Proposed resolution reject: As such a misreading cannot occur in the only use of this convention in the specification there is little value in making changes to it at this point.

6 Source Text

Issue: 1st paragraph: the requirement for UTF-16 puzzles me. I don’t understand what purpose it serves in the specification, and it might lead readers to think that they have to transcode their HTML files to UTF-16! I would think that it would be sufficient to say something like “a conforming interpreter must treat string literals as if they were encoded in UTF-16”. Everything is probably correct as it stands here, but it is probably confusing to all but real Unicode wizards. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: At May F2F - Strings are raw 16-bit code units, source text is Unicode. Trouble occurs when a supplemental character occurs in a string literal -- the spec is not well-defined. Agreed to change wording of chapter 6 to state that source text is first converted to 16-bit code units if it's not already in that form. If it's in one of the Unicode formats then convert it to UTF-16 and treat the resulting 16-bit code units as inputs to the lexer grammar. If it's already in 16-bit code units (as would happen when eval'ing a string), pass those through verbatim. Note that they might contain unmatched surrogates which would not have been allowed under UTF-16.

This is accomplish via the following:

> ECMAScript source text is represented as a sequence of characters in the Unicode character encoding, version 3.0 or later ~~using the UTF-16 transformation format~~. The text is expected to have been normalised to Unicode Normalised Form C (canonical composition), as described in Unicode

Technical Report #15. Conforming ECMAScript implementations are not required to perform any normalisation of text, or behave as though they were performing normalisation of text, themselves. ECMAScript source text is assumed to be a sequence of 16-bit code units for the purposes of this specification. Such a source text may include sequences of code units that are not valid UTF-16 character encodings. If an actual source text is encoded in a form other than 16-bit code units it must be processed as if it was first convert to UTF-16.

SourceCharacter **::**
any Unicode ~~character~~ code unit

Throughout the rest of this document, the phrase "code unit" and the word "character" will be used to refer to a 16-bit unsigned value used to represent a single 16-bit unit of ~~UTF-16~~ text.). The phrase "code point" refers to such a Unicode scalar value. "Unicode character" ~~This~~ only refers to entities represented by single Unicode scalar values: the components of a combining character sequence are still individual "Unicode characters," even though a user might think of the whole sequence as a single character.

7 Lexical Conventions

Issue: It is unfortunate that one of the non-terminals (aka tokens) of this lexical grammar is named "Token". This is confusing. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: True, but probably too much work to change for this edition. Emphasis and capitalization distinguish the two meanings.

7.1 Cf characters in identifiers

Issue: Second paragraph says that Cf characters can be used in identifiers but identifier grammar (7.6) does not allow for them.
jgraham@opera.com https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002430.html and much other discussion in es5-discuss
Resolution: Replace 7.1 with the following and fix up whitespace and IdentifierPart

The Unicode format-control characters (i.e., the characters in category "Cf" in the Unicode Character Database such as LEFT-TO-RIGHT MARK or RIGHT-TO-LEFT MARK) are control codes used to control the formatting of a range of text in the absence of higher-level protocols for this (such as mark-up languages).

It is useful to allow format-control characters in source text to facilitate editing and display. All format control characters may be used in within comments, and within string literals and regular expression literals.

<ZWNJ> and <ZWJ> are format control characters that are used to make necessary distinctions when forming words or phrases in certain languages. In ECMAScript source text, <ZWNJ> and <ZWJ> may also be used in an identifier after the first character.

<BOM> is a format-control character used primarily at the start of a text to mark it as Unicode and to allow detection of the text's encoding and byte order. <BOM> characters intended for this purpose can sometimes also appear after the start of a text, for example as a result of concatenating files. <ZWSP> is a format-control character used for line break control when justifying text. In ECMAScript source text, <ZWSP> and <BOM> characters are treated as white space characters (7.2).

The following format-control characters have special treatment outside of coments, string literals, and regular expression literals:

Code Unit Value	*Name*	*Formal Name*	*Usage*
\u200B	Zero width space	<ZWSP>	*Whitespace*
\u200C	Zero width non-joiner	<ZWNJ>	*IdentifierPart*
\u200C	Zero width joiner	<ZWJ>	*IdentifierPart*
\uFEFF	Byte Order Mark	<BOM>	*Whitespace*

7.2 White Space

Issue: says that whitespace can appear in string literals but not other tokens. This refers to the Token production, but it is confusing because comments and regexps can contain whitespace as well… The distinction between regexps and other "tokens" is an artificial one anyway, so I think it would be clearer to list all non-terminals that can contain whitespace… Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Revised second sentence as follows:

> White space characters may occur between any two tokens, and may occur within a *StringLiteral* or a *RegularExpressionLiteral* (where they are considered significant characters forming part of the literal value) or within a *Comment*,…

Issue: NEF should not be a whitespace character

Resolution: May F2F, yes that is correct. Remove NEL from table in 7.2 and in the WhiteSpace grammar production in 7.2 and A.1. Also from Annex E section 7.2 item.

Issue: <ZWSP> should not be whitespace, both for consistency with Unicode spec. and for backwards compatibility with ES3. https://mail.mozilla.org/pipermail/es5-discuss/2009-June/002867.html

Resolution: remove <ZWSP> from whitespace productiuons.

7.3 Line Terminator

Issue: same comment as above. Line terminators can occur in comments and it is confusing that comments aren't considered "tokens". Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Replace sentencing beginning “A line terminator cannot occur…” with:

> ~~A line terminator cannot occur within any token, except a *MultiLineComment* or a *StringLiteral*. Any line terminators within a *StringLiteral* token must be preceded by an escape sequence.~~

David-Sarah ad issues with the above because comments are not tokens so did some more restructuring and moved text about comments to a separate paragraph.

7.5 Tokens

Issues: The organization of the grammar gets strange here. It seems to me that 7.6 should come before 7.5. 7.5 begins with a production for identifiers, but then the body of the section defines reserved words, with no indication that they are kinds of identifiers. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution. ~~Leave as is~~. The real problem is that ReservedWord is not listed as a Token ~~nor does it have the get its own section like the other kinds of Token and giving them one would change the section number for the remainder of section 7.~~ It isn’t clear why ReservedWord isn’t an alternative for Token.

Added note suggest by David-Sarah that explains that DivPunctuator and RegularExpressionLiteral defines tokens but not *Tokens*.

Bit the bullet and moved sections related to reserved words (7.5.1-7.5.3) to be subsections of 7.6

7.5.3 Future Reserved Words

Issue: const enum export extends import super are the only FutureReservedWords that are actually reserved on IE (or any other browser). (May F2F)

Resolution: We'll unreserve the rest except for the following, which would behave just like let and yield: implements interface package private protected public static

7.5.1, 7.5.2 , 10.2.1.2 Identifier usage

Issue: “In Reserved words cannot be used as identifiers.” And similar context it isn’t clear whether or not “identifiers” is referring to the like named non-terminal. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed fix. In 7.5.1 and 7.5.2 capitalize and italicize "Identifiers" in these contexts. In 10.2.1.2 replace the first “identifiers” with “identifier names” and the second “identifiers” with “an *IdentifierName*”

7.5.3 Let and Yield reserved in strict mode

Issue: The spec. doesn’t say this. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html Waldemar https://mail.mozilla.org/pipermail/es5-discuss/2009-May/002572.html . Note that Walder

also propose a number of sections in 11, 12, 13 where he thinks we need to explicitly state that it is a syntax error for let or yield to show up. I think this is overkill for what we are trying to accomplish here and that the proposed fixed below accomplishes the same thing with a much smaller change. allenwb@micrsoft.com

Proposed Fix: Replace *NOTE* with:

> The tokens **`let`**, and **`yield`** are also considered to be *FutureReservedWords* when they occur within strict mode code (10.1.1). The occurrence of either of these tokens within strict mode code in a context where the occurrence of a *FutureReservedWords* would produce an error must also produce an equivalent error.

Note that Waldemar proposed a different wording for the first sentence but I don't think his extra words add anything essential.

7.6 Identifiers

Issue: Continuing sloppy use of "Identifier" and "Identifier Name" allenwb@microsoft.com and https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html.

Proposed Fix: Replace most occurrences of "identifier" in this section's propose with "*IdentifierName*". See marked-up draft for details.

Issue: Change sentence in one of the middle paragraphs to: A UnicodeEscapeSequence cannot be used to put a character into an identifier that would otherwise be illegal, whether due to grammar rules or due to rules in sections 11, 12, and 13 that cause some uses of identifiers to be syntax errors or EvalErrors. Waldemar https://mail.mozilla.org/pipermail/es5-discuss/2009-May/002572.html This fix seems a little to specific to the "eval" restrictions. I think a more general statement can be make. allenwb@micrsoft.com

Proposed Fix: Add the following sentence to the end of the 3d paragraph:

> All interpretations of identifiers within this specification are based upon the actual characters of the identifier regardless of whether or not an escape sequence was used to contribute any particular characters.

Issue: bottom of page 17: the HexDigit production is not needed here. It belongs in 7.8.4 Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Fix: Actually, 7.8.3 seems like where it belongs as that is the first explicit reference to it. Move it after the *HexIntegerLiteral* production. Also fix it in Annex A.1

Issue: <ZWNJ> and <ZWJ> should be allow as IdentiferPart characters

Resolution: add them.

7.8.5 Regular Expression Literals

Issue: in the sentence "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." the word "should" should be replaced with "must". https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html.

Issue: there seems to be a typo in

RegularExpressionBackslashSequence **::**
\ *NonTerminator* ← Should this not be *RegularExpressionNonTerminator?*

Resolution: fixed

Typos: Fixed various grammar formatting errors identified by Waldemar.

7.9 Automatic Semicolon Insertion

Issue: this is a really awkward paragraph--statements must be terminated with semicolons, but the semicolons may be omitted. . Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Proposed Resolution: Fix if time available. It's a difficult concept to express, however, I don't think the current text is actually as self contradictory as this comment suggests.

8 Types

Issue: . 2nd paragraph: would it simplify things throughout the specification if you formally defined Function as a subtype of Object? Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution : Leave as it. I don't think it would simplify things as for consistency other built-in object "types" would also need to be specified and we would end up with a more complex hierarchical model of ECMAScript types that included subtyping relationships that are not really relevant to specifying the semantics of the language.

8.6.2,8.12.5, 8.12.6, etc. [[Put]] and [[ThrowingPut]]

Issue: All calls to [[Put]] should be replaced by calls to [[ThrowingPut]] and then [[ThrowingPut]] should be renamed to [[Put]].

In the Table 4 of 8.6.2 replace the Domain and Descriptions cells of [[Put]] with the corresponding [[ThrowingPut]] cells and delete the ThrowingPut row of the table.

Eliminate redundant mentions of [[ThrowingPut]] in 8.6.2

Rename section 8.12.5 from [[ThrowingPut]] to [[Put]] and delete section 8.12.6. Renumber 8.12.7-8.12.10 by -1.

Add extra final **false** argument to [[Put]] calls: 10.2.1.2.2 step 4; 11.1.4 Step 3 first algorithm and step 4 second algorithm;

Add extra final **true** argument to [[Put]] calls: 15.4.4.6 step 4.a, 15.4.4.9 step 4.a

Replaces all algorithm references to [[ThrowingPut]] with [[Put]] in: 8.7.2, , 15.4.4.6, 15.4.4.7, 15.4.4.8, 15.4.4.9, 15.4.4.11, 15.4.4.12, 15.4.4.13,

Delete first sentence in NOTE at end of 11.1.4.

8.6 The Object Type

Issue: 1st bullet point: either say "value and a set of boolean attributes" or just a "set of attributes" since [[Value]] is defined as an attribute on the next page. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Issue: 2nd bullet: method or function? These are called as methods, but they're described as functions in the table on the next page. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

~~Resolution: Rephrase first two bullet items as:~~

- ~~A *named data property* associates a name with a value attribute and a set of Boolean attributes.~~
- ~~A *named accessor property* associates a name with a get function attribute, a set function attribute, and a set of Boolean attributes.~~

Issue: Waldemar found the revised phrasing ambiguous

Resolution: Rephrase the first two bullet items as:

- A *named data property* associates a name with an ECMAScript language value and a set of Boolean attributes.
- A *named accessor property* associates a name with one or two accessor functions, and a set of Boolean attributes. The accessor functions are used to store or retrieve an ECMAScript language value that is associated with the property.

Issue: 3rd bullet: cut "by the language specification". Also, since we've just talked about an "accessor property", it is really confusing to have the phrase "property accessor" appear here. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Delete last sentence of bullet. Replace “via the property accessor operators” with “via ECMAScript language operators.

8.6.1 Property Attribute Table

Issue: 1st table: change header of 2nd column to match the tables that follow. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Fix: change second table heading to match first table. Use “Value Domain” for both tables.

Issue: For [[Writable]] “attempts by ECMAScript code to change the property’s [[Value]] attribute” isn’t specific enough. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Fix: replace text with “attempts by ECMAScript code to change the property’s [[Value]] attribute using [[Put]]”

Issue: For [[Configurable]] Statement about what cannot change if [[Configurable]] is true is not quite accurate because of allowable change of [[Writable]] from true to false.

Resolution: Leave as it, this non-normative statement is accurate enough. [[DefineOwnProperty]] actually defines the allowable state transitions.

Issue: 2nd table: the descriptions of [[Get]] and [[Set]] should make it clear that these are called as methods, even if they are described as "functions". Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Fix: add ”as a method” immediately after “called” in the description of both [[Get]] and [[Set]].

Issue: What does “called as a method mean? Waldemar

Fix: see expanded definition in 4.3.27

Issue: bottom of page 29: Add "by this language specification" after "is not explicitly specified". Otherwise, it sounds as if user code must explicitly specify these attributes for each object property.

Fix: Make the suggested addition. However, every place a property is defined by this specification its attributes should be explicitly specified. So, it’s possible that we could just delete this table. These defaults may be somewhat confusing because they describe the default attributes of a property created via assignment or via an object literal. They are not the defaults used when a property is defined using Object.defineProperty.

8.6.2 Table 5 Internal Properties of some objects

Issue: table 4: section number crossrefs to the various SpecOp definitions would be helpful. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Proposed resolution: Leave as is. A number of the [[Internal]] methods have multiple alternative definitions and if we listed any we probably should list them all. This would probably require some explanation and possibly complicate the table formatting (should there a x-ref column). We an revisit this issue latter if we have available time.

Issue: For [[Call]], “SpecOp(a List of any)” should be “SpecOp(*any*, a List of *any*)”

Issue: For [[Call]] , use of term “function” in description is inconsistent with other use of the term in the specification.

https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed resolution: fix signature and change description to:

> Executes code associated with the object. Invoked via a function call expression. The arguments to the SpecOp are a this object and a list containing the arguments passed to the function call expression. Objects that implement this internal method are *callable*. Only callable objects that are host objects may return Reference values.

Issue: For [[HasInstance]]: The first sentence is wrong. [[HasInstance]] tests whether the argument delegates to this object's ".prototype". "built-in" above should probably be "native", since it applies to all objects whose behavior is defined by this spec (i.e., not host objects). "built-in" objects are specifically those that exist before execution begins, which is not the distinction you intend. This problem comes up several places, such as the next 3 rows of table 5 , the [[Match]] row, and the first paragraph of 8.7. However, the above paragraph is still wrong. Function.prototype is not an instance of the constructor Function, though presumably it does implement [[HasInstance]]. Also, Object.create(Function.prototype) would create an instance of the constructor Function that doesn't implement [[HasInstance]]. Perhaps the distinction is: Functions are objects whose [[Class]] is "Function". https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html
Proposed resolution: This is non-normative that is largely carry over text from ES3. The first sentence can be improved by saying "Returns a Boolean value indicating whether the argument is likely an Object that was constructed by this object.". Use of "instance of the standard built-in constructor X" is new ES5 language that sounds precise without actually being precise. It probably is better to revert the ES3 language which was "X objects" which is obviously less precise.

Issue: [[PrimitiveValue]] should move from table 4 to table 5.
Resolution: done

Issue: 8.6.2 paragraph before Table 5: Agreed to nail this down: Remove the first sentence and spell out the rules one by one. Note that the rules about [[Writable]]==false implying no value changes is false (the property can be redefined). The rule about [[Configurable]]==false implying no attribute changes is also false (the [[Writable]] attribute can be legally cleared). A rule about no new properties appearing on non-[[Extensible]] host objects is missing. A rule about not flipping [[Extensible]] from false to true is missing. (May F2F)
Resolution: replaced with:

> Host objects may implement these internal methods in any manner unless specified otherwise; for example, one possibility is that [[Get]] and [[Put]] for a particular host object indeed fetch and store property values but [[HasProperty]] always generates **false**.
>
> The [[GetOwnProperty]] method of a host object must conform to the following invariants for each property of the host object:
>
> - If a property is described as a data property and it may return different values over time, then either or both of the [[Writable]] and [[Configurable] attributes must be **true** even if no mechanism to change the value is exposed via the other internal methods.
> - If a property is described as a data property and its [[Writable]] and [[Configurable]] are both, then the SameValue (9.12) must be returned for the [[Value]] attribute of the property on all calls to [[GetOwnProperty]].

- If the attributes other than [[Writable]] may change over time or if the property might disappear, then the [[Configurable]] attribute must be **true**.
- If the [[Writable]] attribute may change from **false** to **true**, then the [[Configurable]] attribute must be **true**.
- If the value of the host object's [[Extensible]] internal property is has been observed by ECMAScript code to be **false**, then if a call to [[GetOwnProperty]] describes a property as non-existent all subsequent calls must also describe that property as non-existent.

The [[DefineOwnProperty]] internal method of a host object must not permit the addition of a new property to a host object if the [[Extensible]] internal property of that host object has been observed by ECMAScript code to be **false**.

If the [[Extensible]] internal property of that host object has been observed by ECMAScript code to be **false** then it must not subsequently become **true**.

Issue: [[Extensible]] false needs to prevent modification of [[Prototype]] (Mark Miller)

Resolution added following:

Every ECMAScript object has a Boolean-valued [[Extensible]] internal property that controls whether or not named properties may be added to the object. If the value of the [[Extensible]] internal property is **false** then additional named properties may not be added to the object. In addition, if [[Extensible]] is **false** the value of the [[Prototype]] internal property of the object may not be modified. Once the value of an [[Extensible]] internal property has been set to **false** if may not be subsequently changed to **true**.

NOTE
*This specification defines no ECMAScript language operators or built-in functions that permit a program to modify an object's [[Prototype]] internal property or to change the value of [[Extensible]] from false to **true**. Implementation specific extensions that modify [[Prototype]] or [[Extensible]] must not violate the invariants defined in the preceding paragraph.*

8.7 and others, Use of null to tag unresolved Reference Values

Issue: "The base value is either null, an Object, a Boolean, a String, a Number, or an environment record (10.2.1). A base value of null indicates that the reference could not be resolved to a binding." This use of null implies that calling a strict function as a function will bind its this to "null". This is inconsistent with previous decisions as well as, for example, Annex D "called as a function, undefined as passed as the this value." This problem reappears several places: 10.2.1.1.7, 10.2.1.2.5, step 7 of 11.2.3, If all of these were simply changed from "null" to "undefined", I believe the only observable difference would be to correct the "this" binding of strict functions called as functions. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed Resolution: In most cases, the use of null to tag an unresolvable Reference isn't directly observable because the primary constructor of References (11.2.1) throws if the base value of the Reference would be set to either null or undefined. To some degree this use of null is a carryover from the ES3 spec. but undefined is probably a better choice and should be fixed. Passing null in 11.2.3 is independent of the Reference issue and also needs to be fixed. In 10.2.2.1 step 1.a, 10.2.1.1.7 , 10.2.1.2.5 and 11.2.3 step 7 replace all "null" with "undefined"

Issue : “it s not clear what the term ‘binding’ means above. Likewise ‘binding values’ in the table in 10.2.1.” https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html
Proposed Resolution: No fix…’binding’ and ‘bind value’ is used relatively informally here and throughout section 10 but the intent seems pretty clear. The language could be improved but I don’t think it is an essential or high priority work item.

8.7.1 GetValue

Typo: UnresolvedReference should be IsUnresolvedReference

8.7.2 PutValue

Issue: At May F2Fgreed to delete both EvalError cases from 8.7.2. They're unnecessary and cause trouble.

Resolution: deleted steps 3.a.i and 5.a

~~Issue: Step 5.a should only throw an EvalError if the reference is strict. jimb@mozilla.com~~

~~Fix: line 5.a should be~~

~~a. If IsStrictReference(*V*) is **true** and GetReferencedName(*V*) is **"eval"**, then throw an **EvalError** exception.~~

Typo: UnresolvedReference should be IsUnresolvedReference

8.10 Property Descriptors

Issue: Needed to define “fully populated property descriptor” and minor editorial cleanups

8.12.9 [[DefaultValue]]

Issue: Throughout the spec, when speaking of hints, String and Number appear in normal font and without quotes. However, presumably, these are actually the strings "String" and "Number". To clarify, these should have quotes and perhaps be in code font. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed resolution: Low priority, time available cleanup. Admittedly annoying but it has a carry over from Es3 that we can live with if necessary.

Issue: 2nd to last paragraph, page 39. Change "O is a Date object" to something like "O is an instance of the Date constructor".

Proposed resolution: Leave as written. The “x object’ phrasing is used throughout the specification to talk about instances of the built-in constructors. However, the section references to date instances should change from “15.9” to “15.9.6”

8.12.10 [[DefineOwnProperty]]

Issue: algorithm steps 5 and 6 should have “**true**” immediately after the “Return”. AllenWB@microsoft.com

Resolution: make change

Issue: In 8.12.10, the definition of [[DefineOwnProperty]], algorithm step 7b should say "fields", not "field". jimb@mozilla.com

Resolution: make change

Issue: algorithm step 10.b: is it intentional that a property value can be changed even if [[Writable]] is false, as long as [[Configurable]] is true? That makes sense, but I think it is worth noting explicitly at the end of the section. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html
Resolution: Yes, this is intentional. The note is a good idea:

> *NOTE*
> *Step 10.b allows any field of* Desc *to be different from the corresponding field of* current *if* current’s *[[Configurable]] field is* **true**. *This even permits changing the [[Value]] of a property those [[Writable]] attribute is* **false**. *This is allowed because a* **true** *[[Configurable]] attribute would permit an equivalent sequence of calls where [[Writable]] is first set to* **true**, *the new [[Value]] is set, and then [[Writable]] is set to* **false**.

Issue: In step 6, it isn’t clear what “same value” means. (jwalden+es@MIT.EDU)

Resolution: add words making it explicit that the SameValue algorithm is used.

Issue: The intent of test various fields of *Desc* for specific values is not oblivious in situations where the fields may not be present . (jwalden+es@MIT.EDU)

Resolution: Added the following note:

> *The above algorithm contains steps that test various fields of the Property Descriptor* Desc *for specific values. The fields that are tested in this manner need not actually exist in* Desc. *If a field is absent then the result of any test of its value is logically false.*

9.1 ToPrimitive

Issue: 1st and 2nd lines: these mention Value and value. I think both ought to be lowercase and italics. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html
Issue: Should this title include the signature? Perhaps "ToPrimitive(value, PreferredType) → primitive | undefined | null https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed Resolution: Don't add the signature line. None of the functions in this section use the signature notation and changing them doesn't add much value at this point. For consistency with the table, call the first argument *input* rather than Value and add appropriate italics.

9.2, 9.3, 9.8, 9.9, 9.10, 9.11 Table Headings

Issue: For consistence with the prose description of these functions, the first column of their respective tables should be labeled "Argument Type" rather than "Input Type" allenwb@microsoft.com

9.4 ToInteger

Issue: Regarding step 4. Return the result of computing sign(number) * floor(abs(number)). I think this is always representable, but I'm not sure so I thought I'd ask. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed Resolution: No change unless somebody knows better. This is a direct carry over from ES3.

101.1 Strict Mode Code

Issue: the first line refers to 4.2.2, which is non-normative. Is that okay? Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: The reference isn't really essential so the easiest resolution is just to delete the opening phrase "As described in section 4.2.2,"

10.2.1 Environment Records

Issue: 1st para: cut "or variables" Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: ok, cut

Issue: 1st table, 3rd row. The description of this abstract version of GetBindingValue does not describe the circumstances in which a ReferenceError is thrown generally enough: it covers the DER case but not the OER case. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: change description to:

> Returns the value of an already existing binding from an environment record. The string value *N* is the text of the bound name. *S* is used to identify strict mode references. If *S* is **true** and the binding does not exist or is uninitialized throw a **ReferenceError** exception.

Issue: In the 4th row of the table, change "strict mode references" to"strict mode assignments"? Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: no change. “Reference” can mean either access or assignment

Issue: Update table to reflect changes needed to support deletable bindings

Resultion: added D argument to CreateMutableBinding, added DeleteBinding method

10.2.1.2.2 CreateMutableBinding (N)

Typo: In algorithm step 1, “declarative” should be replaced with “object”. allenwb@microsoft.com

10.2.1.1 Declarative Environment Records

Issue: the first paragraph has an extra comma before "and/or function declarations". There are only two alternatives there, so no comma is necessary. jimb@mozilla.com

Resolution: delete comma

Issue: 2nd para: looks like there is an unwanted period after "Declarative". Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: don’t see it…

Issue: it would be nice if the subsections that follow were in the same order as the rows of the table above. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html Also applies to 10.2.1.2

Resolution: In the table in 10.2.1, move SetMutableBinding row above GetBindingValue row. Renumber 10.2.1.1.7 ImplicitThisValue as 10.2.1.1.5, increment following section numbers.

Issue: step 4 of 10.2.1.1.6 implies that a flag is necessary to distinguish an uninitialized undefined from an initialized undefined. This is confusing because the section right before also uses the word "initialize" to describe the uninitialized undefined value. I recommend recasting the description of DER so that each binding includes a variable with three possible states: Mutable, Immutable and UninitializedImmutable. Then algorithm steps can refer explicitly to the variable and these values. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: add a sentence to 10.2.1.1 clarify that immutable bindings can existing in an uninitialized state:

> Creation and initialization of immutable binding are distinct steps so it is possible for such bindings to exist in either an initialized or uninitialized state.

10.2.1.1.1 HasBinding

Issue: this algorithm (and several that follow) include an assertion. But you've never defined how assertions work in these algorithms. Add something to section 5.2? Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Probably won't bother. Given the information nature of the pseudo code, the intent seems clear enough.

10.2.1.1.2 CreateMutableBinding

Issue: needed to add additional parameter to support creation of deletable bindings

Resolution: added formal parameter and sentence in step 3

10.2.1.1.3 SetMutableBinding

Issue: The definitions of the SetMutableBinding method for both declarative and object environment records (10.2.1.1.3 and 10.2.1.2.2) refer to "SetMutableValue" in their first sentence. This name doesn't occur anywhere else; I assume they're supposed to be SetMutableBinding. jimb@mozilla.com

Resolution: change "SetMutableValue" to "SetMutableBinding"

10.2.1.1.5 DeleteBinding

Issue: Needed to add new method to support deleting bindings

Resolution: renumbered ImplicitThisValue to 10.2.1.1.6, CreateImmutableBinding to 10.2.1.1.7, and InitializeImmutableBinding to 10.2.1.1.8. Added definition and algorithm for DeleteBinding

10.2.1.2 Object Environment Records

Issue: my first question in the first paragraph was whether OERs worked with inherited properties or own properties only. Might be worth making this clear right away. Also enumerable vs. non-enumerable properties. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution, added sentence to first paragraph:

> Both own and inherited properties are included in the set regardless of the setting of their [[Enumerable]] attribute.

Issue: the second sentence begins, "An environment record binds...". I believe that is meant to say, "An object environment record binds..." since the statement is not true of all environment records. jimb@mozilla.com

Resolution: inserted "object"

10.2.1.2.2 CreateMutableBinding

Issue: needed to add additional parameter to support creation of deletable bindings

Resolution: added new step 4 and change [[Put]] call in final step to a [[DefineOwnProperty]] call.

10.2.1.2.3 SetMutableBinding

Issue: The definitions of the SetMutableBinding method for both declarative and object environment records (10.2.1.1.3 and 10.2.1.2.2) refer to "SetMutableValue" in their first sentence. This name doesn't occur anywhere else; I assume they're supposed to be SetMutableBinding. jimb@mozilla.com

Resolution: change “SetMutableValue” to “SetMutableBinding”

10.2.1.2.4 GetBindingValue

Issue: algorithm step 4: this doesn't match the abstract description of this method in the table in 10.2.1. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Fixed table description, see item under 10.2.1

Issue: The last sentence of the first paragraph should end with "... the result depends on the value of the S argument:". jimb@mozilla.com

Resolution: added “the” before “S”

10.2.1.2.5 DeleteBinding

Issue: Needed to add new method to support deleting bindings

Resolution: renumbered ImplicitThisValue to 10.2.1.2.6. Added definition and algorithm for DeleteBinding

10.2.2.2 NewDeclarativeEnvironmentRecord

Issue: I'd cut "Record" from the name of this function. It creates an environment, not an environment record. 10.2.2.3: ditto. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: renamed these two abstract operations. In addition to 10.2.2.2 and 10.2.2.3 changes in 10.4.2, 10.4.3, 12.10, 12.14, and 13.

Issue: In step 2 “DeclarativeEnvironmentRecord” should be “declarative environment record” jimb@mozilla.com

Resolution: make the change

10.2.2.3 NewObjectEnvironmentRecord

Issue: Why not make ProvideThis another parameter of NewObjectEnvironmentRecord, rather than setting it after creation? I think it would make things simpler and clearer. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html why doesn't this function take an initial value for providesThis? Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Proposed Resolution: No Change. ProvideThis was a bug fix required to correct an over-sight in the revised specification in relationship to with binding. It is somewhat of a hack but is only needed when establishing the environment record for a with (and not for the global) environment. It seems fine to make setting it be a separate operation that is used only within with statement.

Issue: In step 2 “ObjectEnvironmentRecord” should be “object environment record” jimb@mozilla.com

Resolution: make the change

10.4.2 Eval Code

Issue: In step 1 "If there is no calling context or" is unnecessary, since any such circumstance cannot be a direct call. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed Resolution: No change. While the above observation is correct it doesn’t hurt anything and may provide useful guidance to implementers who are internally using eval for things like html event handlers.

10.5 Arguments Object

Issue: I'd move this section so it comes after 10.6. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Move as suggested and changed various cross references.. The original ordering was to avoid a forward reference to the arguments object section from the declaration instantiation section. However, I agree that swapping the order make the whole thing flow better.

Issue: The "may be created" in the first line is vague. How about a cross reference to the section that explains when it is created and when it isn't. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Replaced first paragraph with:

> When control enters an execution context for function code, an arguments object is created unless (section 10.5) the identifier **`arguments`** occurs as an *Identifier* in the function's *FormalParameterList* or as the *Identifier* of *VariableDeclaration* or *FunctionDeclaration* contained in the function code.

Issue: algorithm step 17.b and 17.c: these use the undefined identifier F. I think it should be obj. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: change both occurrences of "*F*" to "*obj*"

Issue: the MakeArgGetter and MakeArgSetter stuff seems like a kludge. But I don't have anything better to propose. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: no change

Issue: 1st sentence of the note at the end of the section: I think I read this algorithm pretty carefully, but it looks to me as if numbered properties of the arguments object are handled completely differently than named properties, and I can't see how they share their values. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: no change for now, the magic is all in the functions generated by MakeArgGetter and MakeArgSetter which get stored as accessor properties in the PrameterMap object. The second paragraph of the note touches on this. Any ideas for making this mechanism more obvious would be appreciated.

Issue: In Note at end of 10.5: "The "caller" property has a more specific meaning for non-strict mode functions and a "callee" property has historically been provided as an implementation-defined extension by some ECMAScript implementations." "caller" and "callee" are switched here. "callee" is the specified one. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: swap them.

Issue: MakeArgGetter and MakeArgSetter should be referred to as "functions". (AWB)
Resolution: Change use of "function" to "abstract operations"

Issue: the spec currently defines arguments objects to have an own copy of the built in Object.prototype.toString and Object.prototype.toLocaleString. This was intended to provide backwards compatibility to edition 3 where argument objects directly inherit from Object.prototype. However, this copying of the built-in methods does not account for the possibly that a program may have replaced the built-in definitions.
Resolution: rather than copying the built-in methods argument objects now have own methods that dynamically delegate to the current corresponding methods in Object.prototype. This is still not perfect

backwards compatibility but the remaining incompatibilities are extreme edge cases such as redefining Object.prototype. as a non-function value.

Issue: At May F2F it was decided that a implementation defined “caller” property on non-strict function’s arguments object should not be allowed to expose a strict function object..

Resolution: Modified custom [[Get]] method to throw a TypeError exception if getting a “caller” property tries to return a strict mode function object

Issue: for compatability with ES3, bindings created by eval code need to be deletable
Resolution: add addition argument to CrateMutableBinding that identifiers which bindings are deletable.

Issue: Step 7.c.i (now 8.c.i) had a bogus extra “and false” argument to the call to CreateMutableBinding
Resolution: deleted it

10.6 Declaration Binding Instantion

Issue: 2nd paragraph defines func as an "input" to the algorithm, but then algorithm 3a defines this same variable. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: Revise last sentence of second paragraph to remove func:

> On entering an execution context, bindings are created in the VariableEnvironment as follows using the caller provided *code* and, if it is function code, argument List *args*:

Issue: Change "Bind" to "Binding" in function names in steps 3.d.iv, 4.d, 7.b.ii, and 7.c.ii Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: do it.

Issue: step 4b: insert "section" before 13. Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: ok, but all the section/clause stuff changes when we convert to ISO format.

Issue: step 7.c.ii: Maybe change "strict" to false? Flanagan https://mail.mozilla.org/pipermail/es-discuss/2009-May/009171.html

Resolution: do it

Issue: Step 4.b should say “processing” rather than “evaluating”

Resolution: Get rid of “processing” concept. Call in function declaration instantiation instead

11.1.1 The this keyword

Issue: In “evaluates to the value of the ThisBind of the current execution context” what is “ThisBinding” https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: not a problem, ThisBinding for execution context’s is defined in 10.2

Typo: ThisBind should be ThisBinding

11.1.4 Array Initializer

Typo: last line before the Note: remove the spurious g (Flanagan)

11.1.5 Object Initialiser

Issue: step 4.a of the algorithm for production

PropertyNameAndValueList **:** *PropertyNameAndValueList* **,** *PropertyAssignment*

should be checking for multiple data property definitions using the same name rather than multiple accessor property definitions. (abdulmoh@microsoft.com)

Resolution: replace step 4.a with:

> a. This production is contained in strict code and IsDataDescriptor(*previous*) is **true** and IsDataDescriptor(*propId*.descriptor) is **true**.

11.2.4 Argument Lists

Issue: throughout, change “internal list” to “List”

Resolution: done

11.3.1 and 11.3.2 Postfix increment/decrement operators

Issue: need to throw syntax error if a strict mode an expression is a ref to eval.,

Fix: added an algorithm step with the throw conditions.

11.4.1 The delete operator

Issue: “*When a* ***`delete`*** *operator occurs within strict mode code, a* ***ReferenceError*** *exception is thrown if its UnaryExpression is a direct reference to a variable, function argument, or function name.*” Shouldn't this be an early error?

Resolution: May F2F: make it a SyntaxError

Typo: UnresolvedReference should be IsUnresolvedReference

Issue: Need to be able to delete bindings that were created by eval code.

Resolution: In step 5, call DeleteBinding to do the work. Add clause to last sentence of note clarify that the TypeError only applies to strict mode code.

11.4.2 and 11.14 void and , operators

Issue: It is obscure why GetValue is being called when its value is ignored.
https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed Resolution: add a Note to each section:

> *NOTE*
> *GetValue must be called even though its value is not used because it may have observable side-effects.*

11.4.3 typeof Operator

Issue: Last row of table should not allow host objects to claim to be "undefined", "boolean", "number", or "string".

Resolution: added this exclusion to table

11.4.4 and 11.4.5 Prefix increment/decrement operators

Issue: need to throw syntax error if a strict mode an expression is a ref to eval.,

Fix: added an algorithm step with the throw conditions.

11.5 Multiplicative Operators

Issue: In step 3, "MultiplicativeExpression" should be deleted.
Resolution: yes it should

Issue: Shouldn't step 5 occur between steps 2 and 3? https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html
Resolution: Not a bug. The specified order preserves the evaluation order of ES3.

11.5.1 Applying the * Operator:

Issue: "If the magnitude is too large to represent, the result is then an infinity of appropriate sign. If the magnitude is too small to represent, the result is then a zero of appropriate sign. The ECMAScript language requires support of gradual underflow as defined by IEEE 754." This boilerplate code occurs several places: 11.5.2, 11.6.3. Can't it simply be replaced with ToNumber, where this conversion is already covered? https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html
Resolution: No change. These sections are not algorithmic specifications so a call to ToNumber doesn't really fit it. The boiler plate doesn't hurt anything and cleaning it up isn't a high priority.

11.8.7 The in operator

Issue: Line 5. If Type(rval) is not Object, throw a TypeError exception. This seems less useful than either doing a ToObject or just returning false. I prefer doing ToObject. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: No change. Either would be an observable change from the ES3 semantics.

11.13.1 Simple Assignment

Issue: In the NOTE, need to mention that that the LHS also can't be an accessor property with attribute value {[[Setter]]:undefined}. Same issue in Annex C https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: new note text:

> *When an assignment occurs within strict mode code, its LeftHandSide must not evaluate to an unresolvable reference. If it does a* ***ReferenceError*** *exception is thrown upon assignment. The LeftHandSide also may not be a reference to a data property with the attribute value {[[Writable]]:****false****}, to an accessor property with the attribute value {[[Put]]:****undefined****} nor to a non-existent property of an object whose [[Extensible]] internal property has the value* ***false****. In these cases a* ***TypeError*** *exception is thrown.*

Issue: need to throw syntax error if a strict mode an expression is a ref to eval.,

Fix: added an algorithm step with the throw conditions.

11.13.2 Compound Assignment

Issue: need to throw syntax error if a strict mode an expression is a ref to eval.,

Fix: added an algorithm step with the throw conditions.

12 Statement

Issue: Even though these two conflicting uses of "empty" are distinguished by font, it is still confusing. Also, what kind of value is "the single element empty"?...https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: No change. This is all carry over from ES3. No doubt it could be improved but it is not something we should try to do that this point in the release process.

12.1.1 Variable Statement Strict Mode Restrictions

Issue: Use of "eval" as a variable name should be a syntax error.

Resolution: At May F2F agreed to treat "arguments" definitions the same as "eval" definitions within strict mode. Both would cause syntax errors.

12.6 Iteration Statements

Issue: the first sentence of this section is kind of contradicted by the syntax of the do/while loop. I just skimmed quickly, but I don't think you need a definition of "header" or "body" for this section, so I'd just cut the sentence. Or reword it to say that loops have a statement as their body and the iteration is controled by one or more expressions? (Flanagan)

Resolution: cut the sentence

12.12 Labelled Statements

Issue: Shouldn't "(continue, V, L)" be explicitly dealt with? https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: Probably no change, unless somebody can positively identify that something essential is missing.. These is no changes here from ES3. Note that continue termination values are handled within the looping statements. The whole non-sequential control flow mechanism of the specification could be make much clearer but probably not in this specification.

12.14 Try Statement

Issue: In strict mode code, the variable declared by the identifier of a *catch* clause should be restricted from being "eval". All other ways are declaring an identifier are strict mode code already have this restriction. jimb@mozilla.com

Resolution: Add this new section:

12.14.1 Strict Mode Restrictions

It is an **SyntaxError** if a *TryStatement* with a *Catch* occurs within strict code and the *Identifier* of the *Catch* production is either `"eval"` or `"arguments"`.

Issue: should have a note similar to the one in 12.10 about restoring the lexical environment.

Resolution: added

13 Function Definition

Issue: Step 1 of *FunctionBody* : *SourceElements* needs to be rewritten to be consistent with changes to the definition of a Use Strict Directive.

Resolution: Step 1 now says:

1. The code of this *FunctionBody* is strict mode code if it is part of a *FunctionDeclaration* or *FunctionExpression* that is contained in strict mode code or if the Directive Prologue (14.1) of its *SourceElements* contains a Use Strict Directive or if any of the conditions in 10.1.1 apply. If the code of this *FunctionBody* is strict mode code, *SourceElements* is processed and evaluated in the following steps as strict mode code. Otherwise, *SourceElements* is processed and evaluated in the following steps as non-strict mode code.

Issue: , the FunctionBody production is defined as *SourceElements*$_{opt}$, but the semantics is only given for when *SourceElements* is present.

Resolution: fix algorithm to return (**normal**, **empty**, **empty**) if *SourceElements* is not present.

13.1 Function Definition Strict Mode Restrictions

Issue: I don't remember for sure, but didn't we decide to prohibit bindings of "arguments" in strict code, just as we correctly do for "eval"? https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html +1 from David Sarah https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002430.html

Resolution: At May F2F agreed to treat "arguments" definitions the same as "eval" definitions within strict mode. Both would cause syntax errors. There is no longer anything that throws EvalError, but we will retain it for compatibility with ES3.

Issue: "eval" and "arguments" should be illegal as the name of a *FunctionExpression* in addition to *FunctionDeclaration*

Resolution : fixed

Issue: Step 2 in evaluating FunctionBody needs to be delete. The processing of FunctionDeclaration takes place in Declaration Binding Instantiation rather than here.

Resolution: done.

13.2 Creating Function Objects

Issue: the setting of the [[HasInstance]] internal property is not specified.

Resolution: inserted new step 6 that sets it to the definition in 15.3.5.3

Issue: At May F2F it was decided that a implementation defined "caller" property on non-strict function's arguments object should not be allowed to expose a strict function object..

Resolution: inserted to step to install special [[Get]] method defined in 15.3.5.4

13.2.1 [[Call]]

Issue: step 1: change FormalParameterList to [[FormalParameters]]

Resolution: done.

13.2.3 The [[ThrowTypeError]] Function Object

Issue: The inclusion of steps 4, 5, and 6 here look like a copy paste error. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: Delete step 6, but steps 4 and 5 seem appropriate.

Issue: I think the double-square brackets around ThrowTypeError are inappropriate, since it is not an internal property or property attribute. I think you could simply add a note explaining that this function object will be visible to strict mode code but that it does not have a name in the namespace. Or, just bite the bullet and give it a name: TypeError.thrower (Flanagan)

Resolution: perhaps, but leave as is for now.

14 Program

Issue: "The production: SourceElement : FunctionDeclaration is evaluated as follows: 1. Return (normal, empty, empty)." This implies that eval("3; function foo(){}") should return undefined. On FF 3.0.9 it returns 3, which I had understood was correct. I also don't think this is an intended change from ES3 to ES5. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: Changed the production *SourceElements* **:** *SourceElements SourceElement* to propagate statement values in a manner similar to what is done for Block. No change was necessary for *SourceElement* **:** *FunctionDeclaration* to make this work.

Issue: Step 1 of *Program* : *SourceElements* needs to be rewritten to be consistent with changes to the definition of a Use Strict Directive.

Resolution: Step 1 now says:

1. The code of this *Program* is strict mode code if the Directive Prologue (14.1) of its *SourceElements* contains a Use Strict Directive or if any of the conditions of 10.1.1 apply. If the code of this *Program* is strict mode code, *SourceElements* is processed and evaluated in the following steps as strict mode code. Otherwise *SourceElements* is processed and evaluated in the following steps as non-strict mode code.

Issue: the concept of "processing for function declarations" is not needed here. It is taken care of in Declaration Binding Instantiations

Resolution: remove all of the "processing" paragraphs and algorithms.

Issue: need to close the loop between evaluating a program and establishing a global execution context.

Resolution: make establishing the context and returning a result part of evaluating a program. Added note that initiating program evaluation is up to the implementation

Issue: , the *Program* production is defined as $SourceElements_{opt}$, but the semantics is only given for when *SourceElements* is present.

Resolution: fix algorithm to return (**normal**, **empty**, **empty**) if *SourceElements* is not present.

14.1 Use Strict Directive

Issue: Concerns that tokenization of use strict directive string allows comments in strange places, may make it hard to recognize and disallows text after , that doesn't tokenize using lexical grammar. Waldemar https://mail.mozilla.org/pipermail/es5-discuss/2009-May/002572.html AllenWb claims this is all intentional, a good https://mail.mozilla.org/pipermail/es5-discuss/2009-May/002590.html

Resolution: Based upon discussion at F2F, require exactly "use strict" but allow it to occur anywhere in a sequence of such expression statements at the head of Program or FunctionBody.

15 Standard Built-in Objects

Issue: "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." Is this what we decided on, or did we decide that "unless otherwise specified" extra arguments are ignored? https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed Resolution: see new text below. We decided to delete the sentence that explicitly allow an implementation to throw an exception for such arguments. However, I don't see how that really helps as an implementation can still decide that its undefined or extended behavior is to throw an error. Regardless, here is a proposed replacement paragraph:

> 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 extra arguments ~~are evaluated by the call and then~~ ignored by the function. However, an implementation may define implementation specific behaviour relating to such arguments as long as the behaviour is not the throwing of a TypeError exception that is predicated simply on the presence of an extra argument

Issue: "Every built-in prototype object has the Object prototype object, which is the initial value of the expression Object.prototype (15.2.4), as the value of its [[Prototype]] internal property, except the Object prototype object itself." Isn't, for example, the value of RangeError.prototype's [[Prototype]] property Error.prototype and not Object.prototype? https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed Resolution: Prefix the sentence with "Unless otherwise specified"

Issue: The relationship between [[Call]] / [[Construct]] and the chapter 15 sections describing "construct called as a function" and "called as part of a new expression" is not explicitly explained. (Mark Miller: https://mail.mozilla.org/pipermail/es5-discuss/2009-July/002888.html)

Resolution: Added the following paragraph to the section 25:

This section generally describes distinct behaviours for when a constructor is "called as a function" and for when it is "called as part of a **new** expression". The "called as a function" behaviour corresponds to the invocation of the constructor's [[Call]] internal method and the "called as part of a new expression" behaviour corresponds to the invocation of the constructor's [[Construct]] internal method.

15.1 The Global Object

Issue: "values of the [[Prototype]] and [[Class]] internal properties of the global object are implementation-dependent." I suspect that we're currently confused about whether the global object should be considered a native or host object. A browser Window object is clearly a host object. Weird.
https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html
https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002430.html

Resolution: Indeed, but nothing really actionable was suggested.

15.1.2.1 eval (x)

Issue: "see also clause 16". Clause?? https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: "clause" usage will get fixed when we convert to ISO format (which I believe uses tat term to mean section"

15.1.2.4 IsNaN

Issue: summary line should be modified to emphasize that the function tests the coerced value rather than the actual value. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: change summary line to: Returns **true** if the argument coerces to **NaN**, and otherwise returns **false.** Also make similar change to summary line of 15.1.2.5

Issue: Also add Note to effect that a reliable test of whether x is a NaN is "x !== x".

Resolution: add

> *NOTE*
> *A reliable way to test if a value* X *is a* **NaN** *is an expression of the form* X !== X. *The result will be* **true** *if and only if* X *is a* **NaN**.

15.1.3 URIHandler Function Propoerties

Issue: In *uriReserved* **::: one of ; / ? : @ & = + $,**
what about **#**? https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html
https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002430.html

Resolution: May F2F decision, Ok as spec'ed

Issue: Decode algorithm should not allow decoding of illegal UTF-8 values

Resolution: Added clarifying verbiage to Decode step 8. Added reference in note to RFC 2629, added new last paragraph to note.

15.2.3.4 Object.getOwnPropertyName

Issue: Should add a note that (unlike keys) the result includes non-enumerable own property names. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed Resolution: no action. The algorithm seems to be clear enough on this point.

Issue: The note says that the character array indexed properties of strings are not included. It should say that they are included. allenwb@microsoft.com

Resolution: May F2F, they are included. Fix note.

15.2.3.5 Object.create

Issue: Probably too radical a change to consider at this late date, but should we have allowed O to be null? https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002439.html

Resolution: Replace line 1 with:

1. If Type(*O*) is not Object or Null throw a **TypeError** exception.

15.2.3.4 Object.seal 15.2.3.6 Object.freeze

Issue: Ends with text explaining failure atomicity of these operations. However, I think neither of these operations can fail. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: Agreed that there are no failure conditions so delete the atomicity requirement.

15.2.3.12 Object.isFrozen

Issue: For consistency, steps three and 3 should be identical to steps 3 and 4 of 15.2.3.11 https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed Resolution: fix it.

15.2.4 Properties of the Object Prototype Object

Issue: “value of the [[Extensible]] internal property is true” should say “initially true”. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: ok. However note that all such statements are redundant with a general statement in section 15.

15.3.2.1 new Function

Typo: In step 12, “with parsing” should be “passing”

15.3.3 Properties of the Function Constructor

Issue: Specification of caller and arguments. This is inconsistent with 13.2 step 16, which defines these as accessors using the thrower to throw. 13.2 is correct. 15.3.4.5 steps 18 and 19, the last paragraph of 15.3.5, and Annex C make the same mistake. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: Deleted paragraph about caller and arguments from 15.3.3 and added requirement that the Function constructor is a function object.

15.3.4.3 apply

15.3.4.4 call

Added note:

> *NOTE*
> *The* thisArg *value is passed without modification as the* **this** *value. This is a change from Edition 3, where a* **undefined** or **null** thisArg *is replaced with the global object and ToObject is applied to all other values and that result is passed as the* **this** *value.*

15.3.4.5 bind

Issue: Step 13 isn't part of the algorithm and should be converted into anote at the end of the algorithm.

Resolution: delete step and add to note.

Issue: Step 16 should be written in active voice.

Issue: “caller” and “arguments” poison pill properties not correctly specified.

Resolution: copied appropriate steps from 13.2

15.3.5 Properties of Function Instances

Issue: the paragraph about “caller” and “argument” properties is not quite right.

Resolution: added mention of functions created by bind as well as references to actual sections that create these properties. Deleted redundant specification (and incorrect) of their attributes

15.3.5.3 [[HasInstance]]

Issue: does not apply to functions created by bind. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: add a NOTE saying such.

15.3.5.4 [[Get]]

Issue: At May F2F it was decided that a implementation defined "caller" property on non-strict function objects should not be allowed to expose a strict function object..

Resolution: Added custom [[Get]] method to throw a TypeError exception if getting a "caller" property tries to return a strict mode function object

15.4 Array Objects

Change: Added definition of *sparse* object (or array) which is then used in the definition of Array.prototype.sort.

Change: Add definition of term "element" which is used in a number of places within this section.

15.4.2.2 new Array (inten0, item1, …)

Issue: The attributes of the initially created array indexed properties are not specified. (AWB)

Resolution: add that they are {[[Writable]]: **true**, [[Enumerable]]: **true**, [[Configurable]]: **true**}

15.4.2.2 new Array (len)

Issue: "If the argument len is a Number and ToUint32(len) is equal to len, then the length property of the newly constructed object is set to ToUint32(len)." Since they're equal, the end of the sentence can be simplified from "ToUint32(len)" to "len". https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: Non-substantive change is probably not worth making.

Issue: The attributes of the array indexed property that is created if len is not a number isn't specified. (AWB)

Resolution: add that they are {[[Writable]]: **true**, [[Enumerable]]: **true**, [[Configurable]]: **true**}

15.4.3.2 Array.isArray

Issue: phrase "behaves as" is *very* misleading here

Resolution: eliminate “behaves” and make explicit that this is a [[Class]] check.

15.4.4.4 Array.prototyope.concat

Issue: step 5.b.iii.3.a is unnecessarily and left over from ES3 spec. The necessary ToString is explicitly done in step 5.b.iii.3.a (tshinnic@io.com) (john.david.dalton@gmail.com)

Resolution: delete the step

15.4.4.11 Array.prototype.sort

Issue: from May F2F

- Allow sorting non-extensible arrays if they have no holes
- Guarantee not calling delete when sorting an array that has no holes
- "this does not have a property with name ToString(i)": should be "own property"

Resolution: Used concept of sparse now defined in 15.4 of address these and related issues.

15.4.4.14 Array.prototype.indexOf

Issue: Steps 9a-9.b should only be performed if a property named *k* actually exists. This means that the result of [[HasProperty]] should be used as guard on executing those steps.
(john.david.dalton@gmail.com)

Resolution: add a call to [[HasProperty]] and make call to [[Get]] and SameValue conditional on its results.

Issue: The 'fromIndex' argument of these methods, if negative, is supposed to be relative to the end of the string. However, it is coerced using 'ToInt32', which does the wrong thing for fromIndex >= 2**31. (David-Sarah Hopwood)

Resolution: In step 5 change ToInt32 to ToInteger

15.4.4.15 Array.prototype.lastIndexOf

Issue: Steps 8a-8.b should only be performed if a property named *k* actually exists. This means that the result of [[HasProperty]] should be used as guard on executing those steps.
(john.david.dalton@gmail.com)

Resolution: add a call to [[HasProperty]] and make call to [[Get]] and SameValue conditional on its results.

Issue: The 'fromIndex' argument of these methods, if negative, is supposed to be relative to the end of the string. However, it is coerced using 'ToInt32', which does the wrong thing for fromIndex >= 2**31. (David-Sarah Hopwood)

Resolution: In step 5 change ToInt32 to ToInteger

15.4.4.17 Array.prototype.some

Issue: the definition should have a statement analogous to the one for every “like the "for all" quantifier in mathematics. In particular, for an empty array, it returns true” https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Resolution: OK added : **some** acts like the "exists" quantifier in mathematics. In particular, for an empty array, it returns **false**.

15.4.5.1 and 15.4.5.2 Array length

Issue: lack of clarity about how non-deletable array indexed properties interact with changing the length of an array. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed Resolution: Add note at end of 15.4.5.2

> *NOTE*
> *Attempting to set the length property of an Array object to a value that is numerically less than or equal to the largest numeric property name of an existing array indexed non-deletable property of the array will result in the length being set to a numeric value that is one greater than that largest numeric property name.*

Issue: Not clear why false is passed as last as throw argument to [[DefineOwnProperty]] in step 4.e.ii (jwalden+es@MIT.EDU)

Resolution: Add note to step that the call will always succeed.

Issue: In steps 3.a.iii and 3.a.vi.3.a, this algorithm mutates one of its input parameters, and that raises the question of whether implementations must make that mutation visible to callers [[DefineOwnProperty]]. I'm not sure that the question even makes sense, but this seems like an unnecessary can of worms. Can the algorithm be changed to make its own copy of Desc? If not, if that mutation is important, I think that fact should be called out in a note.

Resolution: for clarity changed to used a copy of *Desc* along the path that mutates it.

15.5.4.10 String.prototype.match

Issue: [[Put]] calls on lastIndex should be replaced with [[DefineOwnProperty]] calls.

Resolution: Not necessary because 15.10.7.5 says that every RegExp has a non-deleteable lastIndex property

15.5.4.14 String.prototype.split

Issue: [[Put]] calls on *A* should be replaced with [[DefineOwnProperty]] calls.

Resolution: done

15.4.4 Array prototype functions shouldn't throw on failed writes

Issue: Array methods that exist in ES3 have had [[Put]] and [[Delete]] operations replaced with throwing versions of the same operations in order to provided notification when the functions are applied to objects with non-writable whose existence would cause the functions to violate their normal post conditions. This may create a compatibility problem for ES3 implementations that have been extended to support String objects whose individual characters are accessible as array indexed properties. See https://mail.mozilla.org/pipermail/es-discuss/2009-May/009252.html and https://mail.mozilla.org/pipermail/es-discuss/2009-May/009255.html

Resolution: Agreed at May meet that throwing is OK.

15.4.4.16-15.4.4.20 every, some, forEach, map, filter

Issues: 1) Some of them say that the callback returns true or false. But

the algorithms call ToBoolean on the callback return value, so

really, they expect a truthy or falsy value.

2) The descriptions say that the callback is called "as a function"

despite the fact that they all accept an optional object argument

on which the callback can be called as a method.

3) The phrase "elements that are deleted are not visited" could be

expanded a bit for clarity. How about "Elements that are deleted

after the call to every begins but before being visited by

callbackfn will not be visited by callbackfn."

Resolution: fixed them all up.

15.5.5 Properties of String Instances

Issue: does not specify the value of the [[Class]] property of string instances. allenwb@microsoft.com

Fix: specify it, similar to what was done for array.

Issue: Need to mention the array index named properties of string instances.

Resolution: Added sentence:

> The array index named properties correspond to the individual characters of the string value. A special [[GetOwnProperty]] internal method is used to specify the number, values, and attributes of the array index named properties.

15.5.5.2 String [[GetOwnProperty]]

Issue: Should or shouldn't the indexable character properties of strings be enumerable.

Resolution: At May F2F, yes they are enumerable

Issue: At may F2F meeting, Consider if this method is really needed to specify the character properties of strings

Resolution: After further examination, AllenWB concluded that it is still better to define the behavior of these properties algorithmically rather than via a prose description. However, some additional prose in 15.5.5 was also added to clarify things.

Issue: Step 4 should probably directly access the [[PrimitiveValue]] rather than calling ToString on what we know is a String object.

Resolution: Replace step 4 with:

> 4. Let *str* be the String value of the [[PrimitiveValue]] internal property of S.

15.6.5 Properties of Boolean Instances

Issue: does not specify the value of the [[Class]] property of boolean instances nor its [[PrimitiveValue]] internal property. allenwb@microsoft.com

Fix: specify it, similar to what was done for string.

15.7.4 Properties of the Number Prototype Object

Issue: “if Type(this value) is Number” should be "if Type(value) is Number".
https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed Resolution: I actually don’t think this is the right clarify change but I did make the tweaks to the wording of the relevant paragraph.

15.7.5 Properties of Number Instances

Issue: does not specify the value of the [[Class]] property of Number instances nor its [[PrimitiveValue]] internal property. allenwb@microsoft.com

Fix: specify it, similar to what was done for string.

15.8.2.14 Random()

Issue: Should add a note recommending that implementations provide high enough quality randomness as to make it infeasible to infer how many times random() was called between two calls to random(). If that is unacceptable, then should add the opposite note warning that programs may so infer, creating a covert channel hazard. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed resolution: No change. This seems like a well know best practice that isn’t particularly unique to ECMAScript and hence there is no particular reason to mention it.

15.9.1.5 Date Time String Format

Issue: http://www.merlyn.demon.co.uk/js-262-5.htm takes issue with the positioning of this format in relationship to the actual ISO 8601 Extended Format and raises various minor issues WRT the description of the format.

Resolution: changed the intro to avoid inferring that this format has any official relationship to ISO 8610. Made minor a few non-substantive minor changes in the format description.

The above review raises a number of good points that it is too late to address in this addition but we should revisit them in Harmony.

15.9.5.43 Date.prototype.toISOString()

Issue: what does this method produce when applied to (new Date(NaN))? (dhtmlkitchen@gmail.com)

Resolution: throw a RangeError if the time value is not a finite number.

15.9.5.44

Issue: What should toJSON do for Date’s with non-finite date values?

Resolution: Return null. Added two new steps to algorithm:

2. Let *tv* be ToPrimitive(*O*, hint Number).
3. If *tv* is a Number and is not finite, return **null**.

15.9.6 Properties of Date Instances

Issue: does not specify the value of the [[Class]] property of Date instances nor its [[PrimitiveValue]] internal property. allenwb@microsoft.com

Fix: specify it, similar to what was done for string.

15.10 RegExp grammar

Issue: Change to *IdentifierPart* in section 7 unintentionally changes definitioun of *IdentityEscape*.

Resultion: make <ZWJ> and <ZWNJ> explicit alternatives for *IdentityEscape*. This restores it to its ES3 definition.

15.10.6.2 RegExp.prototype.exec

Issue [[Put]] calls in steps 9.a.i and 11.a should be [[DefineOwnProperty]] calls.

Resolution: Not necessary because 15.10.7.5 says that every RegExp has a non-deleteable lastIndex property

15.10.2.10 CharacterEscape

Issue: Unicode terminology usage in consistent with conventions of spec. (allenwb)

Fix: Use “code unit” appropriately instead of code point

15.10.7 Properties of RegExp Instances

Issue: does not specify the value of the [[Class]] property of Number instances nor its [[Match]] internal property. allenwb@microsoft.com

Fix: specify it, similar to what was done for string.

Typo: woird “word” representation missing after “dependent”

15.10.7.5 lastIndex

Issue: A careful reading shows that lastIndex is only coerced to an integer (using ToInteger) when it is used, in 15.10.6.2 step 4. That is, it is not conformant to coerce lastIndex to an integer when the

property is set, because RegExp objects are native and so they must have the default [[Put]] internal. However, implementations differ: SpiderMonkey does coerce lastIndex when it is set; JScript does not. (David-Sarah Hopwood)

Resolution: Modify wording to emphasize that coercion only occurs on use.

15.11.5 Properties of Error Instances.

Issue: does not specify the value of the [[Class]] property of Error instances. allenwb@microsoft.com

Fix: specify it, similar to what was done for string.

Issue: Shouldn't the "name" and "message" properties of 15.11.4.(2 & 3) instead be properties on Error instances? Likewise with 15.11.7.11. https://mail.mozilla.org/pipermail/es5-discuss/2009-April/002428.html

Proposed resolution. Leave as is. I believe that if you follow all the paths the spec. is correct and complete in this regard. There could be some refactoring that could make where these properties actual occur clear but I don't think it is an important enough issues to do that refactoring at this time.

15.11.6.1 EvalError

Issue: No longer thrown in ES5:

Resolution: changed paragraph to:

> This exception is not generated by this specification. This object remains for compatibility with previous editions of this specification.

15.11.7.1 Properties of *NativeError* Instances.

Issue: does not specify the value of the [[Class]] property of *NativeError* instances. allenwb@microsoft.com

Fix: specify it, similar to what was done for Error.

15.12 The JSON Object

Issue: Need to clarify how the specified the ECMAScript JSON format differs from RFC 4627

Resolution: add clarifying sentence and bullet points.

15.12.1.2The JSON Syntactic Grammar

Issue: The use of JSONValue as the goal symbols may cause confusion with in relationship to RFC 4627 which uses JSON-text as its goal symbol.

Resolution: add a *JSONText* production consisting of a *JSONValue*. Change goal symbol reference in 15.12.2 step to *JSONText*

Issue: *JSONSourceCharacter* as defined precludes the occurrence of tabs and new line characters anywhere within JSON text.

Resolution: eliminate *JSONSourceCharacter* production and just use *SourceCharacter*. Exclude control characters within definition of *JSONStringCharacter*.

Typo: In the first sentence of the first paragraph delete the word “from”.

15.12.2 JSON.parse

Issue: In the first paragraph need to explicitly mention difference in handling of U+2028 and U+2029 in JSONString from regular ECMAScript string literals. Fix: immediately after “characters than *WhiteSpace*” insert “ and allows Unicode code points U+2028 and U+2029 to directly appear in *JSONString* literals without using an escape sequence”. After the main algorithm add the note:

> *NOTE*
> *In parsing* JText *in step 3* JSONString *is used in place of* StringLiteral*.*

The above note has now been incorporated into step 3 as normative text.

Typo: In the second paragraph, the phrase “the member is deleted” should be “the property is deleted”.

Typo: In algorithm step 3, the phrase “this result with be” should be “this result will be”.

Issue and proposed change: In Step 2 of Walk the IsCallable test is unnecessary. Fix: delete that clause of the predicate.

Issue and proposed change: Step 2.a.iii.2 and 2.b.ii.3.a of Walk use [[Put]] to insert “revived” values into objects being constructed. However, [[Put]] has the potential of calling an inherited setter function if the property that has being set had been deleted by a previous call to reviver. The fix is to change each to these calls to a call to [[DefineOwnProperty]] using a property descriptor of the form {[[Value]]: *newElement*, [[Writable]]: true, [[Enumerable]]: true, [[Configurable]]: true}.

Typo: In Step 2.a.iii.2 of Walk replace “Let *newElement* be the result of calling” with “Call”.

Typo: In Step 2.b.i of Walk replace “**Object.key**” with “**Object.keys**”

Issue and proposed change: The prose description of the reviver function asserts that returning **undefined** caused the corresponding property to be deleted. However, this deletion does not occur (step 2.a.iii.2 of Walk) if the containing object is an Array. In that case, the corresponding array property is simply over-written with the value **undefined**. Writing undefined to an array element is not the same

as creating a "hole" in the array by deleting the element. The fix is to explicitly delete such array element analogously to what is done in steps 2.b.ii.2-3.

The revised Walk algorithm with all of the above changes is:

1. Let *val* be the result of calling the [[Get]] internal method of *holder* with argument *name*.
2. If *val* is an object, then
   a. If the [[Class]] internal property of *val* is **"Array"**
      i. Set *I* to 0.
      ii. Let *len* be the result of calling the [[Get]] internal method of *val* with argument **"length"**.
      iii. Repeat while *I* < *len*,
         1. Let *newElement* be the result of calling the abstract operation *walk*, passing *val* and ToString(*I*).
         2. If *newElement* is **undefined**, then
            a. Call the [[Delete]] internal method of *val* with ToString(*I*) and **false** as arguments.
         3. Else
            a. Call the [[DefineOwnProperty]] internal method of *val* with arguments ToString(*I*) , the Property Descriptor {[[Value]]: *newElement*, [[Writable]]: true, [[Enumerable]]: true, [[Configurable]]: true}, and **false**.
         4. Add 1 to *I*.
   b. Else
      i. Let *keys* be an internal List of Strings consisting of the names of all the own properties of *val* whose [[Enumerable]] attribute is **true**. The ordering of the strings should be the same as that used by the **Object.keys** standard built-in function.
      ii. For each string *P* in *keys* do,
         1. Let *newElement* be the result of calling the abstract operation *walk*, passing *val* and *P*.
         2. If *newElement* is **undefined**, then
            a. Call the [[Delete]] internal method of *val* with *P* and **false** as arguments.
         3. Else
            a. Call the [[DefineOwnProperty]] internal method of *val* with arguments *P* , the Property Descriptor {[[Value]]: *newElement*, [[Writable]]: true, [[Enumerable]]: true, [[Configurable]]: true}, and **false**.
3. Return the result of calling the [[Call]] internal method of *reviver* passing *holder* as the **this** value and with an argument list consisting of *name* and *val*.

15.12.3 JSON.stringify

Typo: In the first paragraph replace the two occurrences of "JavaScript" with "ECMAScript".

Typo: In the first paragraph replace "value is usually an object" with "value, which is usually an object".

Issue and proposed change: Step 4.a of main algorithm does not consider the possibility that space is not an integer. Also, it has been proposed that the max value be changed to 10. Replace step 4.a with:

a. Let *space* be min(10, ToInteger(*space*)).

Typo: In Step 4.b of main algorithm "space" should be italic in "if space is less".

Issue and proposed change: While the size of a numeric space argument is limited the length of a string space argument is not. They should both have the same length limit. Replace Step 5.a of main algorithm with:

a. If the number of characters in *space* is 10 or less, set *gap* to *space* otherwise set *gap* to a string consisting of the first 10 characters of *space*.

Issue and proposed change: In step 8 of the main algorithm the [[Put]] call should be replaced with a [[DefineOwnProperty]] all.

Revised main stringify algorithm with above changes:

1. Let *stack* be an empty List.
2. Let *indent* be the empty string.
3. If Type(*space*) is object then,
    a. If the [[Class]] internal property of *space* is **`"Number"`** then,
        i. Let *space* be ToNumber(*space*).
    b. Else if the [[Class]] internal property of *space* is **`"String"`** then,
        i. Let *space* be ToString(*space*).
4. If Type(*space*) is number
    a. Let *space* be min(10, ToInteger(*space*)).
    b. Set *gap* to a string containing *space* space characters. This will be the empty string if *space* is less than 1.
5. Else if Type(*space)* is string
    a. If the number of characters in *space* is 10 or less, set *gap* to *space* otherwise set *gap* to a string consisting of the first 10 characters of *space*.
6. Else
    a. Set *gap* to the empty string.
7. Let *wrapper* be a new object created as if by the expression **`new Object()`**, where **`Object`** is the standard built-in constructor with that name.
8. Call the [[DefineOwnProperty]] internal method of *wrapper* with arguments the empty string, the Property Descriptor {[[Value]]: *value*, [[Writable]]: **true**, [[Enumerable]]: **true**, [[Configurable]]: **true**}, and **false**.
9. Return the result of calling the abstract operation *Str* with the empty string and *wrapper*.

Issue and proposed change: Steps 7.a and 7.b of the *Str* algorithm handle String and Number wrapper objects. They would come into play if a prior call to a toJSON method had returned such an wrapper object or if the standard-built in toJSON methods for String and Number were deleted. Boolean wrapper objects could also be encountered for the same reasons, but they are not handled. This could be fixed by adding a step 7.c that deals with Boolean wrappers in a analogous manner. Note that if this change is made then the entirety of step 7 should be moved to be immediately prior to the current step 4.

Issue: The explicit handling of wrapper objects in step 7 of *Str* makes the existence of Number.prototype.toJSON and String.prototype.toJSON unnecessary. Also, Boolean.prototype.toJSON if a 7.c is added. The fix is to delete 15.5.4.21, 15.7.4.8, 15.6.4.4

Resolution: agreed to at May F2F

Typo: In Step 6.a of JO replace "**Object.key**" with "**Object.keys**"

Issue: As currently specified, a replacer array argument could be modified as a side-effect of toJSON or other methods called during execution of the algorithm.

Resolution: Compute the whitelist of properties names once in the top level stringify algorithm

Issue: What happens if a replacer array contains non-string values?

Resolution: When constructing the whitelist of property names, filter out any values that are not strings or numbers.

Revised Str algorithm with above changes:

1. Let *value* be the result of calling the [[Get]] internal method of *holder* with argument *key*.
2. If Type(*value*) is object, then
    a. Let *toJSON* be the result of calling the [[Get]] internal method of *value* with argument **`"toJSON"`**.
    b. If IsCallable(*toJSON*) is **true**
        i. Let *value* be the result of calling the [[Call]] internal method of *toJSON* passing *value* as the **this** value and with an argument list consisting of *key*.
3. If IsCallable(*replacer*) is **true**
    a. Let *value* be the result of calling the [[Call]] internal method of *replacer* passing *holder* as the **this** value and with an argument list consisting of *key* and *value*.
4. If Type(*value*) is object then,
    a. If the [[Class]] internal property of *value* is **`"Number"`** then,
        i. Let *value* be ToNumber(*value*).
    b. Else if the [[Class]] internal property of *value* is **`"String"`** then,
        i. Let *value* be ToString(*value*).
    c. Else if the [[Class]] internal property of *value* is **`"Boolean"`** then,
        i. Let *value* be the value of the [[PrimitiveValue]] internal property of *value*.
5. If *value* is **null** then return **`"null"`**.
6. If *value* is **true** then return **`"true"`**.
7. If *value* is **false** then return **`"false"`**.
8. If Type(*value*) is string, then return the result of calling the abstract operation *Quote* with argument *value*.
9. If Type(*value*) is number
    a. If *value* is finite then return ToString(*value*).
    b. else, return **`"null"`**.
10. If Type(*value*) is object, and IsCallable(*value*) is **false**
    a. If the [[Class]] internal property of *value* is **`"Array"`** then
        i. Return the result of calling the abstract operation *JA* with argument *value*.
    b. Return the result of calling the abstract operation *JO* with argument *value*.
11. Return **undefined**.

Typo: In Step 8.a of JO replace "*str*" with "*Str*"

Typo: In Step 8.a of JA replace "*str*" with "*Str*"

Error: In step 10.b.iii of the *JA* algorithm "{" should be "[" and "}" should be "]".

16 Errors

Issue: EvalError is not longer used in ES5

Resolution: deleted last bullet form early error list.

Annex A.1

Issue: <ZWNJ> and <ZWJ> missing from *IdentifierPart*

Annex A.7

Issue: Change to *IdentifierPart* in section 7 unintentionally changes definitioun of *IdentityEscape*.

Resultion: make <ZWJ> and <ZWNJ> explicit alternatives for *IdentityEscape*. This restores it to its ES3 definition.

Annex C

Add missing item:

- It is an **EvalError** if a *TryStatement* with a *Catch* occurs within strict code and the *Identifier* of the *Catch* production is **`eval`** (12.14.1)

Issues: need to make it clear that null and undefined this values are not converted to the global object
Fix: added sentence to relevant item.

Annex D

Issue: Item for 15.1.2.1 is more appropriate for Annex E

Resolution: move it to Annex E
Annex E

Issue: need to talk about that format control character that were stripped in Es3 may now be incorporated into string and regexp literals.

Resolution: added sentence to 7.1 item.

Add missing items:

> Section 15.4.4: In Edition 5 all methods of **`Array.prototype`** are intentionally generic. In Edition 3 **`toString`** and **`toLocaleString`** were not generic and would throw a **`TypeError`** exception if applied to objects that were not instances of Array.

Section 10.5: In Edition 5 the array indexed properties of argument objects that correspond to actual formal parameters are enumerable. In Edition 3, such properties were not enumerable.

> Section 10.6: In Edition 5, the [[Prototype]] of an arguments object is **`Array.prototype`**. In Edition 3 it was **`Object.prototype`**. However, Edition 5 argument objects also have own properties that over-ride **`Array.prototype.constructor`**, **`Array.prototype.toString`**, and **`Array.prototype. toLocaleString`** with the standard builtin **`Object.prototype`** versions of these properties. Using

Object.prototype.isPrototypeOf to test the prototype of an arguments object or any access to properties of an arguments object that are inherited from **Array.protype** other than **constructor**, **toString** and **toLocaleString** may produce different results.

Section 15: In Edition 5, the following new properties are defined on built-in objects that exist in Edition 3: **Object.getPrototypeOf**, **Object.getOwnPropertyDescriptor**, **Object.getOwnPropertyName**, **Object.create, Object.defineProperty**, **Object.defineProperties**, **Object.seal**, **Object.freeze**, **Object.preventExtensions**, **Object.isSealed**, **Object.isFrozen**, **Object.isExtensible**, **Object.keys**, **Function.prototype.bind**, **Array.prototype.indexOf**, **Array.prototype.lastIndexOf**, **Array.prototype.every**, **Array.prototype.some**, **Array.prototype.forEach**, **Array.prototype.map**, **Array.prototype.filter**, **Array.prototype.reduce**, **Array.prototype.reduceRight**, **String.prototype.trim**, **Date.now**, **Date.prototype.toISOString, Date.prototype.toJSON**.