

Errata, Issues, and Proposed Changes ECMAScript 5 Candidate Specification

May 19, 2009 Revision

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 initialises** 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".

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"

6 Source Test

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:

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>

5.1.7 Grammar Notation

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.

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***,...

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.

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`.

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>

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).

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: 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

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>.

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 **B**oolean attributes.

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: 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 can revisit this issue later 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.

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.

8.12.9 [[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 whose `[[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**.*

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 consistency 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 Codee

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

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.

10.2.1.1.2 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.

10.2.1.1.3 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 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”

10.2.1.2.3 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”

10.2.2.2 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 ParameterMap 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.

10.6 Declaration Binding Instantiation

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

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

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: ??? It isn't specified as such in section 16. Should it be?

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.

11.5 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 to 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.*

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.

12.12 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.

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: I hope not...this would probably require non-trivial changes to 10.6 Declaration Binding Instantiations.

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.

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: No change unless somebody wants to argue that this is an ES3 bug that needs to be fixed. The spec. has not changed in this regard from ES3. However, the above example does appear to be what actual implementations do. Some thought would have to go into how to specify this correctly. It might require a new kind of completion value whose ramification for other parts of the spec. would have to be understood.

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.

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”

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 that 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 Properties

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: ??? Is this a bug. There is no change from ES3. What possible side-effects would there be to adding #?

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: Probably needs some more discussion as there is some controversy about this.

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>

Proposed Resolution: sounds like a very good suggestion! 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.

15.3.3 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: ??? I think there are some subtitles here that I need to consider.

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.

15.4.2.2 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.

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.

15.5.4.10 String.prototype.match

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

15.5.4.14 String.prototype.split

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

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 provide notification when the functions are applied to objects with non-writable properties 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>

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.

15.5.5.2 String `[[GetOwnProperty]]`

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

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 working 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.

15.9.6 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.

15.10.6.2 RegExp.prototype.exec

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

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.

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.

15.11.7.1 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.

15.12.3 The JSON Syntactic Grammar

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

15.12.3 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.

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 $\min(10, \text{ToInteger}(\textit{space}))$.

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 $\min(10, \text{ToInteger}(\textit{space}))$.
 - 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 a 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

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

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 “]”.

Annex E

Add missing item:

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.