Erratum for ECMAScript, 5th Edition Specification (ECMA-262-5)

(Last Updated January 27, 2010)

6 Source Text

(First paragraph)

ECMAScript source text is represented as a sequence of characters in the Unicode character encoding, version 3.0 or later. 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 16-bit 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 converted to UTF-16.

7.1 Unicode Format-Control Characters

(Table 1)

Code Unit Value	*Name*	*Formal Name*	*Usage*
\u200C	Zero width non-joiner	<ZWNJ>	*IdentifierPart*
\u200D~~C~~	Zero width joiner	<ZWJ>	*IdentifierPart*
\uFEFF	Byte Order Mark	<BOM>	*Whitespace*

7.6 Identifier Names and Identifiers

(Missing :: in several grammar productions)

UnicodeLetter **::**
any character in the Unicode categories “Uppercase letter (Lu)”, “Lowercase letter (Ll)”, “Titlecase letter (Lt)”, “Modifier letter (Lm)”, “Other letter (Lo)”, or “Letter number (Nl)”.

UnicodeCombiningMark **::**
any character in the Unicode categories “Non-spacing mark (Mn)” or “Combining spacing mark (Mc)”

UnicodeDigit **::**
any character in the Unicode category “Decimal number (Nd)”

UnicodeConnectorPunctuation **::**
any character in the Unicode category “Connector punctuation (Pc)”

UnicodeEscapeSequence **::**
see 7.8.4.

15.2.3.7 Object.defineProperties (O, Properties)

(confusing use of *P* in steps 5 and 6 of algorithm)

5. For each element *P* of *names* in list order,
    a. Let *descObj* be the result of calling the [[Get]] internal method of *props* with *P* as the argument.
    b. Let *desc* be the result of calling ToPropertyDescriptor with *descObj* as the argument.
    c. Append the pair (a two element List) consisting of *P* and *desc* to the end of *descriptors*.
6. For each element *pair* from ~~*desc*~~ ~~of~~ *descriptors* in list order,
    a. Let *P* be the first element of *pair*.
    b. Let *desc* be the second element of *pair*.
    c.~~a.~~ Call the [[DefineOwnProperty]] internal method of *O* with arguments *P*, *desc*, and **true**.

15.4.4.21 Array.prototype.reduce (callbackfn [, initialValue])

(Fourth paragraph)

The range of elements processed by **reduce** is set before the first call to *callbackfn*. Elements that are appended to the array after the call to **reduce** begins will not be visited by *callbackfn*. If existing elements of the array are changed, their value as passed to *callbackfn* will be the value at the time **reduce** visits them; elements that are deleted after the call to **reduce** ~~**filter**~~ begins and before being visited are not visited.

15.4.4.22 Array.prototype.reduceRight (callbackfn [, initialValue])

(Fourth paragraph)

The range of elements processed by **reduceRight** is set before the first call to *callbackfn*. Elements that are appended to the array after the call to **reduceRight** begins will not be visited by *callbackfn*. If existing elements of the array are changed by *callbackfn*, their value as passed to *callbackfn* will be the value at the time **reduceRight** visits them; elements that are deleted after the call to **reduceRight** ~~**filter**~~ begins and before being visited are not visited.

15.8.2 Function Properties of the Math Object

(First NOTE paragraph)

NOTE The behaviour of the functions **acos**, **asin**, **atan**, **atan2**, **cos**, **exp**, **log**, **pow**, **sin**, ~~and~~ **sqrt**, and **tan** is not precisely specified here except to require specific results for certain argument values that represent boundary cases of interest. For other argument values, these functions are intended to compute approximations to the results of familiar mathematical functions, but some latitude is allowed in the choice of approximation algorithms. The general intent is that an implementer should be able to use the same mathematical library for ECMAScript on a given hardware platform that is available to C programmers on that platform.

15.10.6.3 RegExp.prototype.test(string)

(Section reference in first step of algorithm)

1. Let *match* be the result of evaluating the **`RegExp.prototype.exec`** (15.10.6.2~~3~~) algorithm upon this RegExp object using *string* as the argument.

A.1 Lexical Grammar

(Missing :: in several grammar productions)

UnicodeLetter **::** *See 7.6*
any character in the Unicode categories “Uppercase letter (Lu)”, “Lowercase letter (Ll)”, “Titlecase letter (Lt)”, “Modifier letter (Lm)”, “Other letter (Lo)”, or “Letter number (Nl)”.

UnicodeCombiningMark **::** *See 7.6*
any character in the Unicode categories “Non-spacing mark (Mn)” or “Combining spacing mark (Mc)”

UnicodeDigit **::** *See 7.6*
any character in the Unicode category “Decimal number (Nd)”

UnicodeConnectorPunctuation **::** *See 7.6*
any character in the Unicode category “Connector punctuation (Pc)”

(Insert between *DecimalDigit* and *ExponentIndicator* production)

DecimalDigit **:: one of** *See 7.8.3*
`0 1 2 3 4 5 6 7 8 9`

NonZeroDigit **:: one of** *See 7.8.3*
`1 2 3 4 5 6 7 8 9`

ExponentPart **::** *See 7.8.3*
ExponentIndicator SignedInteger

ExponentIndicator **:: one of** *See 7.8.3*
`e E`

(incorrect right-hand-side)

RegularExpressionBackslashSequence **::** *See 7.8.5*
\ *RegularExpressionNonTerminator*

A.8.1 JSON Lexical Grammar

(incorrect right-hand-side)

JSONStringCharacter **::** *See 15.12.1.1*
~~*JSON*~~*SourceCharacter* **but not** *double-quote* **"** **or** *backslash* **** **or** *U+0000* ***thru*** *U+001F*
**** *JSONEscapeSequence*

ANNEX C

(next to last bullet item, confusing wording)

- An implementation may not extend, beyond that defined in this specification, the ~~associate special~~ meanings within strict mode functions of ~~to~~ properties named **`caller`** or **`arguments`** of function instances. ECMAScript code may not create or modify properties with these names on function objects that correspond to strict mode functions (10.6, 13.2, 15.3.4.5.3).