ecma INTERNATIONAL

Standard ECMA-404

1st Edition / October 2013

The JSON Data Interchange Format

Standard

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

is the registered trademark of Ecma International

COPYRIGHT PROTECTED DOCUMENT

© Ecma International 2013

Contents

Page

1 Scope 1
2 Conformance 1
3 Normative references 1
4 JSON Text 1
5 JSON Values 2
6 Objects 2
7 Arrays 3
8 Numbers 3
9 String 4

© Ecma International 2013

Introduction

JSON is a text format that facilitates structured data interchange between all programming languages. JSON is syntax of braces, brackets, colons, and commas that is useful in many contexts, profiles, and applications. JSON was inspired by the object literals of JavaScript aka ECMAScript as defined in the ECMAScript Language Specification, third Edition [1]. It does not attempt to impose ECMAScript's internal data representations on other programming languages. Instead, it shares a small subset of ECMAScript's textual representations with all other programming languages.

JSON is agnostic about numbers. In any programming language, there can be a variety of number types of various capacities and complements, fixed or floating, binary or decimal. That can make interchange between different programming languages difficult. JSON instead offers only the representation of numbers that humans use: a sequence of digits. All programming languages know how to make sense of digit sequences even if they disagree on internal representations. That is enough to allow interchange.

JSON text is a sequence of Unicode code points. JSON also depends on Unicode in the hex numbers used in the `\u` escapement notation.

Programming languages vary widely on whether they support objects, and if so, what characteristics and constraints the objects offer. The models of object systems can be wildly divergent and are continuing to evolve. JSON instead provides a simple notation for expressing collections of name/value pairs. Most programming languages will have some feature for representing such collections, which can go by names like `record`, `struct`, `dict`, `map`, `hash`, or `object`.

JSON also provides support for ordered lists of values. All programming languages will have some feature for representing such lists, which can go by names like `array`, `vector`, or `list`. Because objects and arrays can nest, trees and other complex data structures can be represented. By accepting JSON's simple convention, complex data structures can be easily interchanged between incompatible programming languages.

JSON does not support cyclic graphs, at least not directly. JSON is not indicated for applications requiring binary data.

It is expected that other standards will refer to this one, strictly adhering to the JSON text format, while imposing restrictions on various encoding details. Such standards may require specific behaviours. JSON itself specifies no behaviour.

Because it is so simple, it is not expected that the JSON grammar will ever change. This gives JSON, as a foundational notation, tremendous stability. JSON was first presented to the world at the `JSON.org` website in 2001. JSON stands for JavaScript Object Notation.

This Ecma Standard was adopted by a General Assembly postal vote in October 2013.

© Ecma International 2013

COPYRIGHT NOTICE

© 2013 Ecma International

This document may be copied, published and distributed to others, and certain derivative works of it may be prepared, copied, published, and distributed, in whole or in part, provided that the above copyright notice and this Copyright License and Disclaimer are included on all such copies and derivative works. The only derivative works that are permissible under this Copyright License and Disclaimer are:

(i) works which incorporate all or portion of this document for the purpose of providing commentary or explanation (such as an annotated version of the document),

(ii) works which incorporate all or portion of this document for the purpose of incorporating features that provide accessibility,

(iii) translations of this document into languages other than English and into different formats and

(iv) works by making use of this specification in standard conformant products by implementing (e.g. by copy and paste wholly or partly) the functionality therein.

However, the content of this document itself may not be modified in any way, including by removing the copyright notice or references to Ecma International, except as required to translate it into languages other than English or into a different format.

The official version of an Ecma International document is the English language version on the Ecma International website. In the event of discrepancies between a translated version and the official version, the official version shall govern.

The limited permissions granted above are perpetual and will not be revoked by Ecma International or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and ECMA INTERNATIONAL DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

Software License

All Software contained in this document ("Software)" is protected by copyright and is being made available under the "BSD License", included below. This Software may be subject to third party rights (rights from parties other than Ecma International), including patent rights, and no licenses under such third party rights are granted under this license even if the third party concerned is a member of Ecma International. SEE THE ECMA CODE OF CONDUCT IN PATENT MATTERS AVAILABLE AT http://www.ecma-international.org/memento/codeofconduct.htm FOR INFORMATION REGARDING THE LICENSING OF PATENT CLAIMS THAT ARE REQUIRED TO IMPLEMENT ECMA INTERNATIONAL STANDARDS*.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the authors nor Ecma International may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE ECMA INTERNATIONAL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL ECMA INTERNATIONAL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

© Ecma International 2013

The JSON Data Interchange Format

1 Scope

JSON is a lightweight, text-based, language-independent data interchange format. It was derived from the ECMAScript programming language, but is programming language independent. JSON defines a small set of structuring rules for the portable representation of structured data.

2 Conformance

Conforming JSON text is a sequence of Unicode code points that strictly conforms to the JSON grammar.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 10646:2012*, Information Technology – Universal Coded Character Set (UCS)*

The Unicode Consortium. The Unicode Standard, Version 6.2.0, (Mountain View, CA: The Unicode Consortium, 2012. ISBN 978-1-936213-07-8)

http://www.unicode.org/versions/Unicode6.2.0/.

4 JSON Text

A JSON text is a sequence of tokens formed from Unicode code points that conforms to the JSON value grammar. The set of tokens includes six structural tokens, strings, numbers, and three literal name tokens.

The six structural tokens:

[U+005B left square bracket

{ U+007B left curly bracket

] U+005D right square bracket

} U+007D right curly bracket

: U+003A colon

, U+002C comma

These are three literal name tokens:

`true` U+0074 U+0072 U+0075 U+0065

`false` U+0066 U+0061 U+006c U+0073 U+0065

© Ecma International 2013

`null` U+006E U+0075 U+006C U+006C

Insignificant whitespace is allowed before or after any token. The whitespace characters are: character tabulation (U+0009), line feed (U+000A), carriage return (U+000D), and space (U+0020). Whitespace is not allowed within any token, except that space is allowed in strings.

5 JSON Values

A JSON value can be an *object*, *array*, *number*, *string*, `true`, `false`, or `null`.

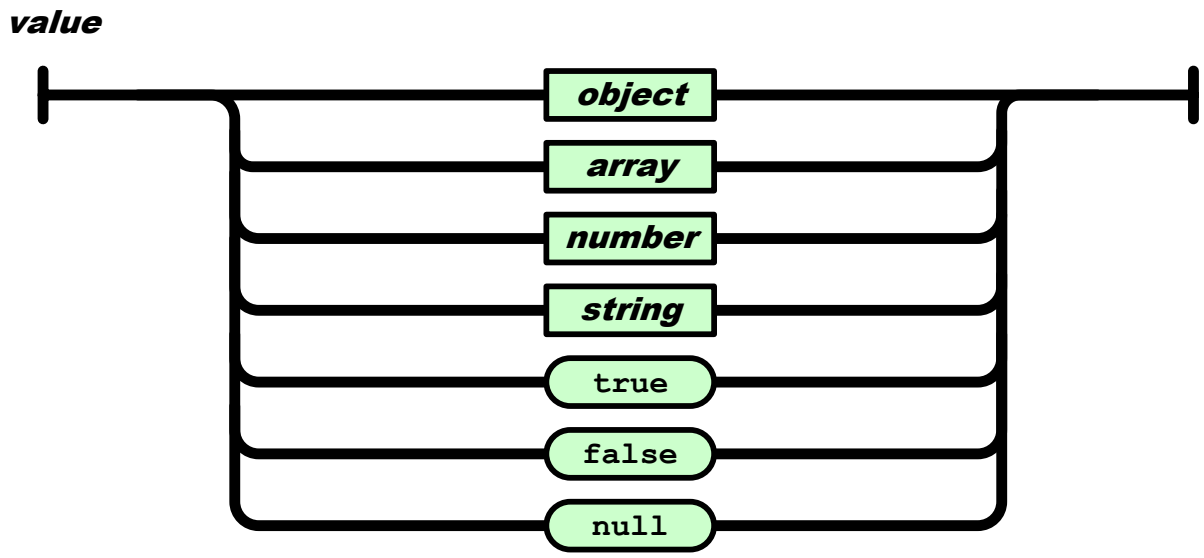

Figure 1 — value

6 Objects

An object structure is represented as a pair of curly bracket tokens surrounding zero or more name/value pairs. A name is a *string*. A single colon token follows each name, separating the name from the *value*. A single comma token separates a *value* from a following name.

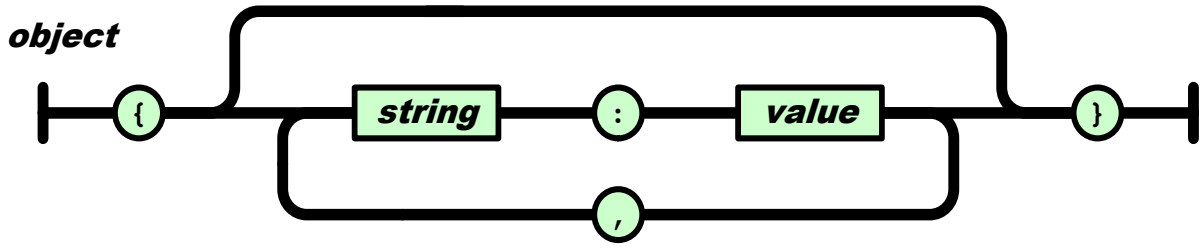

Figure 2 — object

© Ecma International 2013

7 Arrays

An array structure is a pair of square bracket tokens surrounding zero or more *values*. The *values* are separated by commas. The order of the *values* is significant.

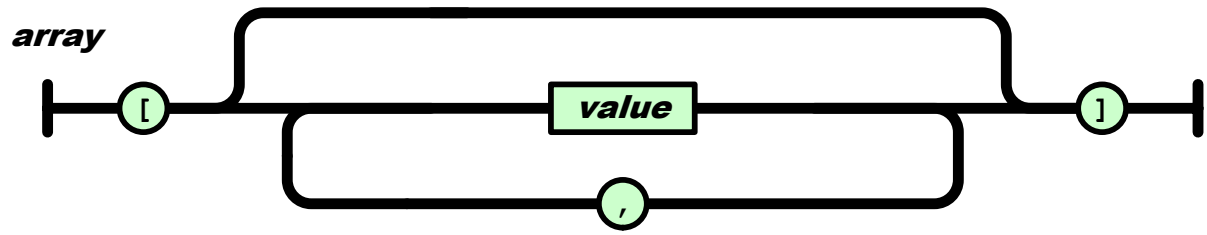

Figure 3 — array

8 Numbers

A number is represented in base 10 with no superfluous leading zero. It may have a preceding minus sign (U+002D). It may have a `.` (U+002E) prefixed fractional part. It may have an exponent of ten, prefixed by `e` (U+0065) or `E` (U+0045) and optionally `+` (U+002B) or `−` (U+002D). The digits are the code points U+0030 through U+0039.

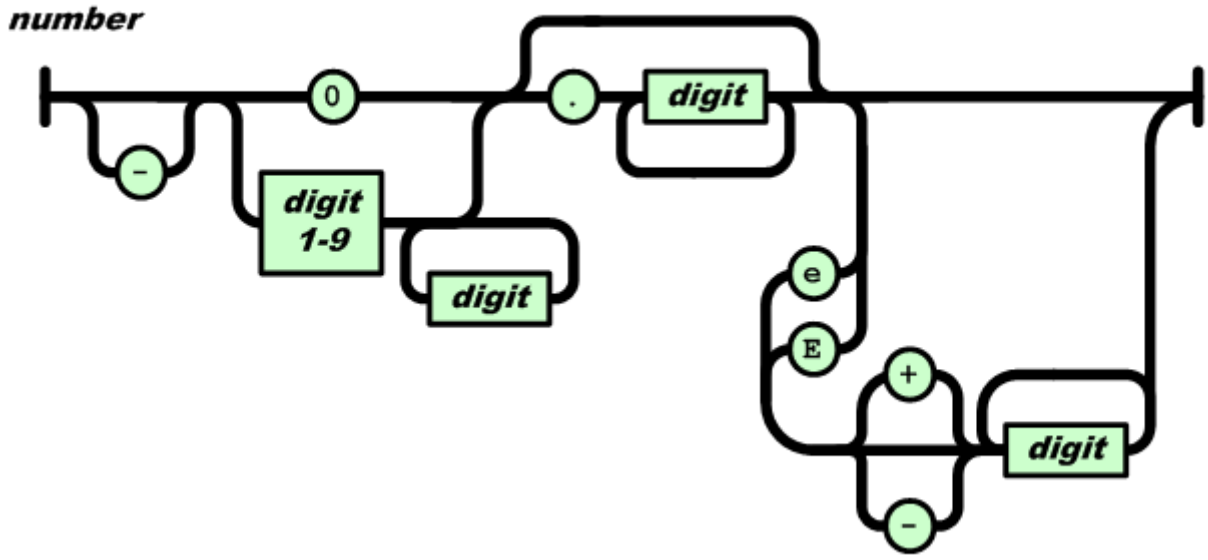

Figure 4 — number

Numeric values that cannot be represented as sequences of digits (such as `Infinity` and `NaN`) are not permitted.

© Ecma International 2013

9 String

A string is a sequence of Unicode code points wrapped with quotation marks (U+0022). All characters may be placed within the quotation marks except for the characters that must be escaped: quotation mark (U+0022), reverse solidus (U+005C), and the control characters U+0000 to U+001F. There are two-character escape sequence representations of some characters.

`\"` represents the quotation mark character (U+0022).

`\\` represents the reverse solidus character (U+005C).

`\/` represents the solidus character (U+002F).

`\b` represents the backspace character (U+0008).

`\f` represents the form feed character (U+000C).

`\n` represents the line feed character (U+000A).

`\r` represents the carriage return character (U+000D).

`\t` represents the character tabulation character (U+0009).

So, for example, a string containing only a single reverse solidus character may be represented as `"\\"`.

Any code point may be represented as a hexadecimal number. The meaning of such a number is determined by ISO/IEC 10646. If the code point is in the Basic Multilingual Plane (U+0000 through U+FFFF), then it may be represented as a six-character sequence: a reverse solidus, followed by the lowercase letter `u`, followed by four hexadecimal digits that encode the code point. Hexadecimal digits can be digits (U+0030 through U+0039) or the hexadecimal letters `A` through `F` in uppercase (U+0041 through U+0046) or lowercase (U+0061 through U+0066). So, for example, a string containing only a single reverse solidus character may be represented as `"\u005C"`.

The following four cases all produce the same result:

```
"\u002F"

"\u002f"

"\/"

"/"
```

To escape a code point that is not in the Basic Multilingual Plane, the character is represented as a twelve-character sequence, encoding the UTF-16 surrogate pair. So for example, a string containing only the G clef character (U+1D11E) may be represented as `"\uD834\uDD1E"`.

© Ecma International 2013

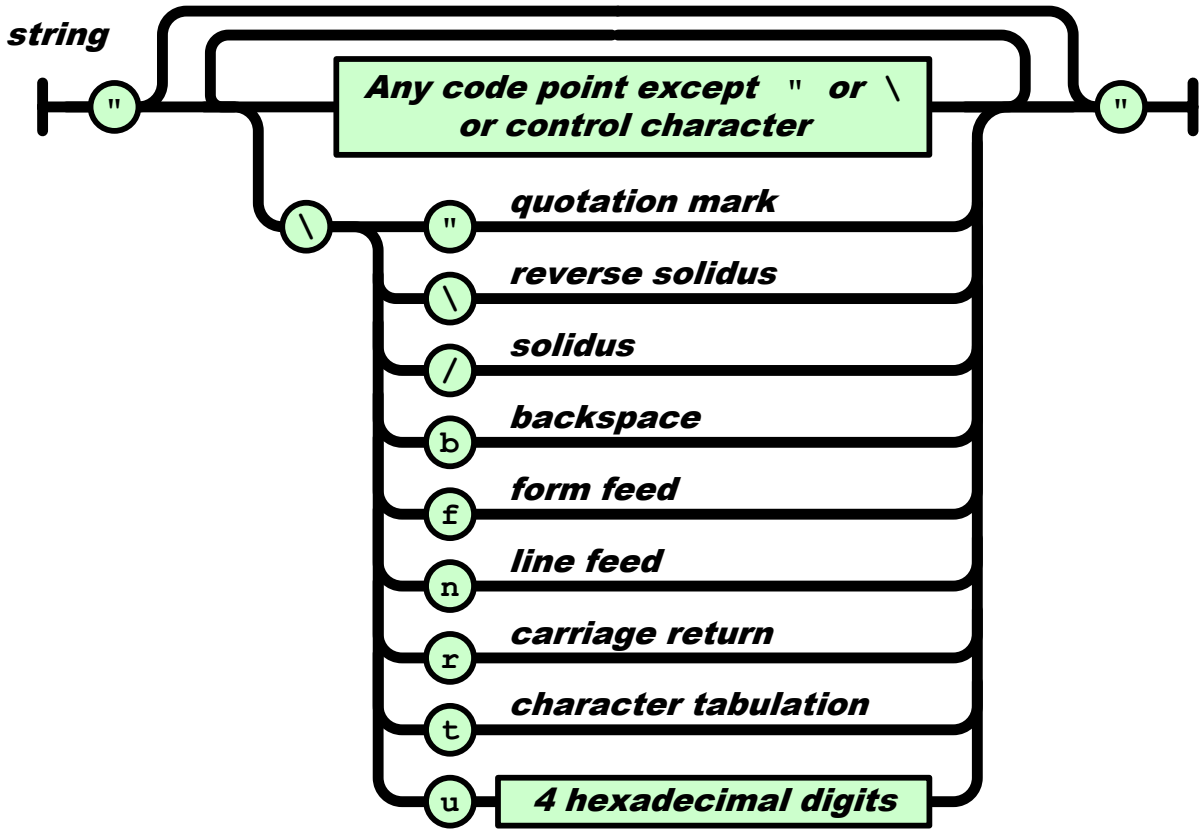

Figure 5 — string

© Ecma International 2013

© Ecma International 2013

Bibliography

[1] ECMA-262 (ISO/IEC 16262), *ECMAScript® Language Specification, 3rd edition (December 1999)*

© Ecma International 2013

© Ecma International 2013