ecma INTERNATIONAL

Standard ECMA-404

2nd Edition / December 2017

The JSON Data Interchange Syntax

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

Contents

Page

Introduction

JSON* 是一种文本语法，它促进了所有编程语言之间的结构化数据交换。JSON 是一种包括大括号、方括号、冒号和逗号的语法，在许多情境、概况和应用中都非常有用。JSON 代表 JavaScript 对象表示法，其灵感来自于 JavaScript（又称 ECMAScript）中的对象字面量，如第三版 ECMAScript 语言规范 [1] 所定义。然而，它并不试图将 ECMAScript 的内部数据表示强加于其他编程语言。相反，它与所有其他编程语言共享 ECMAScript 语法的一个小子集。JSON 语法不是完整数据交换的规范。有意义的数据交换需要生产者和消费者就特定使用 JSON 语法的语义达成一致。JSON 提供的是可以附加这些语义的语法框架。

JSON 语法描述了一系列 Unicode 码点。JSON 在 \u 转义表示法中使用的十六进制数字也依赖于 Unicode。

JSON 对数字的语义是不可知的。在任何编程语言中，都可能存在各种容量和补码的不同数字类型，固定的或浮动的，二进制或十进制。这可能使不同编程语言之间的交换变得困难。JSON 仅提供人类使用的数字表示：一串数字。所有编程语言都知道如何理解数字序列，即使它们对内部表示有不同的看法。这足以允许交换。

编程语言在是否支持对象以及对象提供哪些特性和约束方面差异很大。对象系统的模型可能截然不同且不断发展。JSON 相反提供了一种表达名称/值对集合的简单记法。大多数编程语言都会有某种表示这种集合的特性，这些特性可能被称为记录、结构、字典、映射、哈希或对象。

JSON 还提供对有序值列表的支持。所有编程语言都会有某种表示这种列表的特性，这些特性可能被称为数组、向量或列表。因为对象和数组可以嵌套，所以可以表示树和其他复杂的数据结构。通过接受 JSON 的简单约定，可以轻松地在不兼容的编程语言之间交换复杂数据结构。

JSON 至少不直接支持循环图。JSON 不适用于需要二进制数据的应用。

预计其他标准将参考这一标准，严格遵守 JSON 语法，同时对各种编码细节的语义解释和限制施加限制。这些标准可能要求特定的行为。JSON 本身不指定任何行为。

因为它非常简单，不预期 JSON 语法会发生变化。这使得 JSON 作为一种基础性记法具有极大的稳定性。

JSON 首次于 2001 年在 JSON.org 网站向世界展示。随后，JSON 语法的定义以 IETF RFC 4627 的形式于 2006 年 7 月发布。ECMA-262 第五版（2009年）包含了 JSON 语法的规范性规范。本规范，ECMA-404，取代了之前对 JSON 语法的定义。同时，IETF 发布了 RFC 7158/7159，并在 2017 年发布 RFC 8259 作为对 RFC 4627 的更新。本规范和 RFC 8259 所指定的 JSON 语法旨在完全一致。

* Pronounced /ˈdʒeɪˌsən/, as in “Jason and The Argonauts”.

这一ECMA标准由技术委员会39开发，并在2017年12月得到大会通过。

The JSON Data Interchange Syntax

1 Scope （范围）

JSON 是一种轻量级的、基于文本的、与编程语言无关的语法，用于定义数据交换格式。它源自 ECMAScript 编程语言，但与编程语言无关。JSON 定义了一小套规则，用于便携式表示结构化数据。

这个规范的目标仅是定义有效 JSON 文本的语法。它的目的不是提供任何符合该语法的文本的语义或解释。它也有意不定义如何将有效的 JSON 文本内化为编程语言的数据结构。可以应用于 JSON 语法的语义有很多种，JSON 文本可以通过多种方式被编程语言处理或映射。使用 JSON 进行有意义的信息交换需要各方在应用的具体语义上达成一致。定义 JSON 的具体语义解释可能是其他规范的话题。同样，JSON 的语言映射也可以独立指定。例如，ECMA-262 定义了有效 JSON 文本与 ECMAScript 运行时数据结构之间的映射。

2 Conformance （一致性）

“符合规范的 JSON 文本是一系列严格符合本规范定义的 JSON 语法的 Unicode 码点序列。

处理 JSON 文本的符合规范的处理器不应接受任何不符合规范的 JSON 文本作为输入。符合规范的处理器可能施加语义限制，限制它将处理的符合规范的 JSON 文本的集合。”

3 Normative References （规范性引用）

以下引用的文件对于本文档的应用是必不可少的。对于有日期的引用，仅适用所引用的版本。对于无日期的引用，适用引用文档的最新版本（包括任何修订）

ISO/IEC 10646 信息技术 - 通用编码字符集（UCS）

Unicode联盟。Unicode标准 http://www.unicode.org/versions/latest。

Bray, T., 编辑. 'JavaScript 对象表示法（JSON）数据交换格式', RFC 8259。

本规范和 [RFC 8259] 都提供了 JSON 语法的规范，但使用了不同的形式。目的是使这两个规范定义相同的语法语言。如果在它们之间发现差异，国际欧洲电脑制造商协会（Ecma International）和互联网工程任务组（IETF）将共同努力更新这两个文档。如果发现任何一个文档有错误，应检查另一个文档看是否有类似的错误，并尽可能修复。如果未来任何一个文档发生变化，国际欧洲电脑制造商协会和互联网工程任务组将共同工作，确保这两份文档在变更过程中保持一致。RFC 8259 还定义了对 JSON 语法使用的各种语义限制。这些限制对本规范并非规范性要求。

4 JSON Text

JSON 文本是由符合 JSON 值语法的 Unicode 码点组成的标记序列。这组标记包括六个结构标记（方括号、花括号、冒号、逗号）、字符串、数字和三个字面名标记。

六个结构标记：

[U+005B 左方括号

{ U+007B 左花括号

] U+005D 右方括号

} U+007D 右花括号

: U+003A 冒号

U+002C 逗号

这些是三个字面名标记：

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

false U+0066 U+0061 U+006C U+0073 U+0065

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

在任何标记之前或之后都允许有不重要的空白。空白是下列一个或多个码点的序列：字符制表符（U+0009）、换行符（U+000A）、回车符（U+000D）和空格（U+0020）。除了字符串中允许有空格之外，标记内不允许有空白。

5 JSON Values

JSON 值可以是对象、数组、数字、字符串、true、false 或 null。

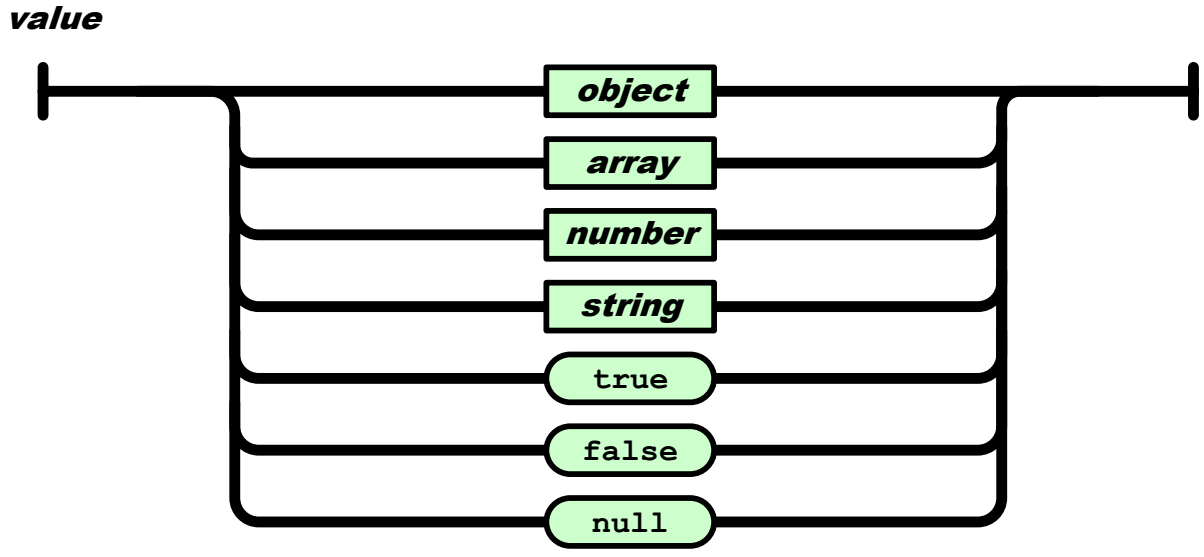

Figure 1 — value

6 Objects

对象结构由一对花括号标记表示，围绕着零个或多个名称/值对。名称是一个字符串。每个名称后跟一个冒号标记，用来将名称与值分开。一个逗号标记用来将一个值与后续的名称分开。JSON 语法对用作名称的字符串不施加任何限制，不要求名称字符串是唯一的，也不赋予名称/值对的顺序任何意义。这些都是语义上的考虑，可能由 JSON 处理器定义，或在定义 JSON 数据交换的特定用途的规范中定义。

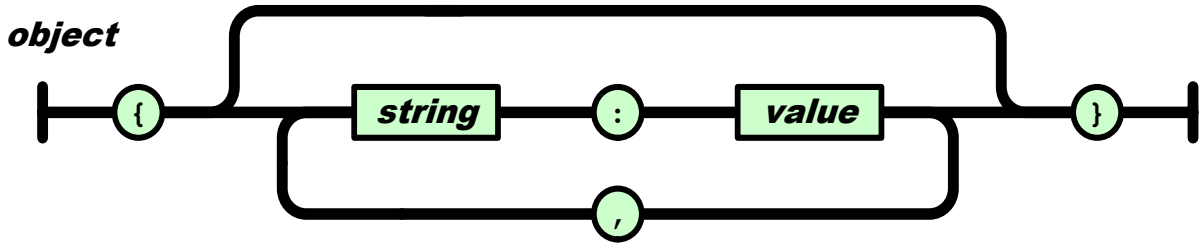

Figure 2 — object

7 Arrays

数组结构是由一对方括号标记包围的零个或多个值。这些值由逗号分隔。JSON 语法没有为值的顺序定义任何特定的含义。然而，JSON 数组结构通常用于值的顺序具有一定语义含义的情况。

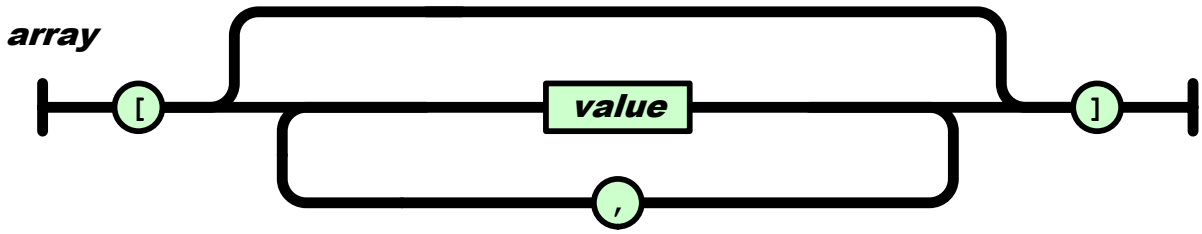

Figure 3 — array

8 Numbers

数字是一串十进制数字，没有多余的前导零。它可能有一个前置的负号（U+002D）。它可能有一个由小数点（U+002E）前缀的小数部分。它可能有一个由 e（U+0065）或 E（U+0045）前缀的指数，并可选地加上 +（U+002B）或 –（U+002D）。数字是代码点 U+0030 到 U+0039。

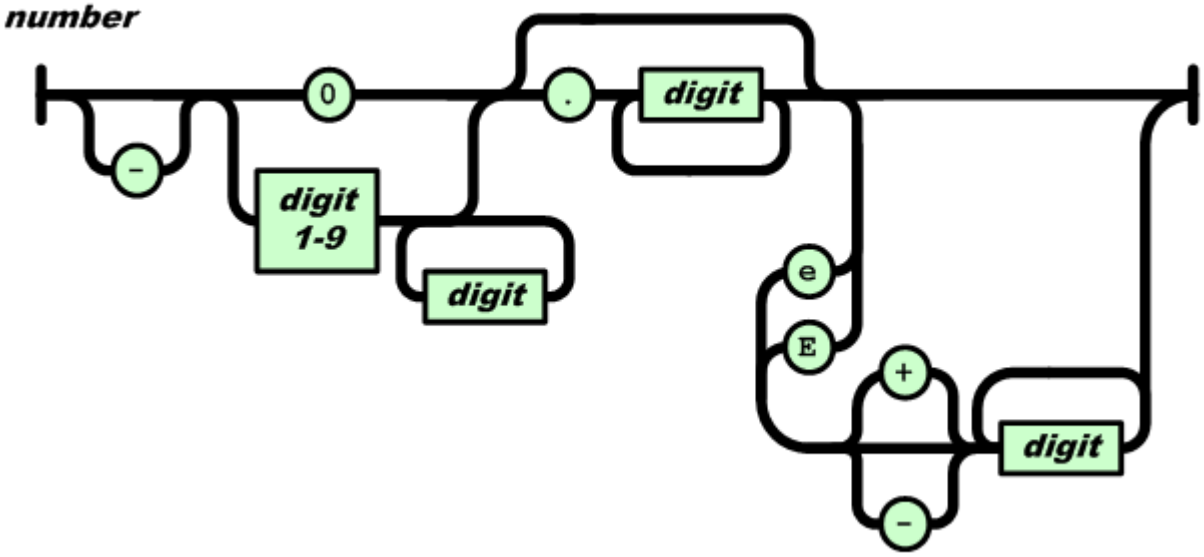

Figure 4 — number

不能表示为数字序列的数值（如 Infinity 和 NaN）是不允许的。

9 String

字符串是用引号（U+0022）包裹的 Unicode 码点序列。所有码点都可以放在引号内，除了必须转义的码点：引号（U+0022）、反斜杠（U+005C）和控制字符 U+0000 到 U+001F。有些字符有两个字符的转义序列表示。

\" 代表引号字符（U+0022）。

\\ 代表反斜杠字符（U+005C）。

\/ 代表斜杠字符（U+002F）。

\b 代表退格字符（U+0008）。

\f 代表换页字符（U+000C）。

\n 代表换行字符（U+000A）。

\r 代表回车字符（U+000D）。

\t 代表制表符字符（U+0009）。

例如，一个只包含单个反斜杠字符的字符串可以表示为"\\".

任何码点都可以用十六进制转义序列表示。这样的十六进制数的含义由 ISO/IEC 10646 决定。如果码点在基本多语种平面（U+0000 至 U+FFFF）内，则可以表示为一个六字符序列：一个反斜杠，后跟小写字母 u，再跟四个十六进制数字，这些数字编码了该码点。十六进制数字可以是数字（U+0030 至 U+0039）或大写（U+0041 至 U+0046）或小写（U+0061 至 U+0066）的 A 至 F 十六进制字母。因此，例如，一个只包含单个反斜杠字符的字符串可以表示为 '\u005C'。

以下四种情况都产生相同的结果：

```
"\u002F"

"\u002f"

"\/"

"/"
```

要转义不在基本多语种平面内的码点，字符可以表示为编码 UTF-16 代理对的十二字符序列，对应于该码点。例如，一个只包含高音谱号字符（U+1D11E）的字符串可以表示为 "\uD834\uDD1E" 。然而，JSON 文本的处理器是将这样的代理对解释为单个码点还是明确的代理对，是由特定处理器决定的语义决策。

请注意，JSON语法允许Unicode当前不提供字符分配的代码点

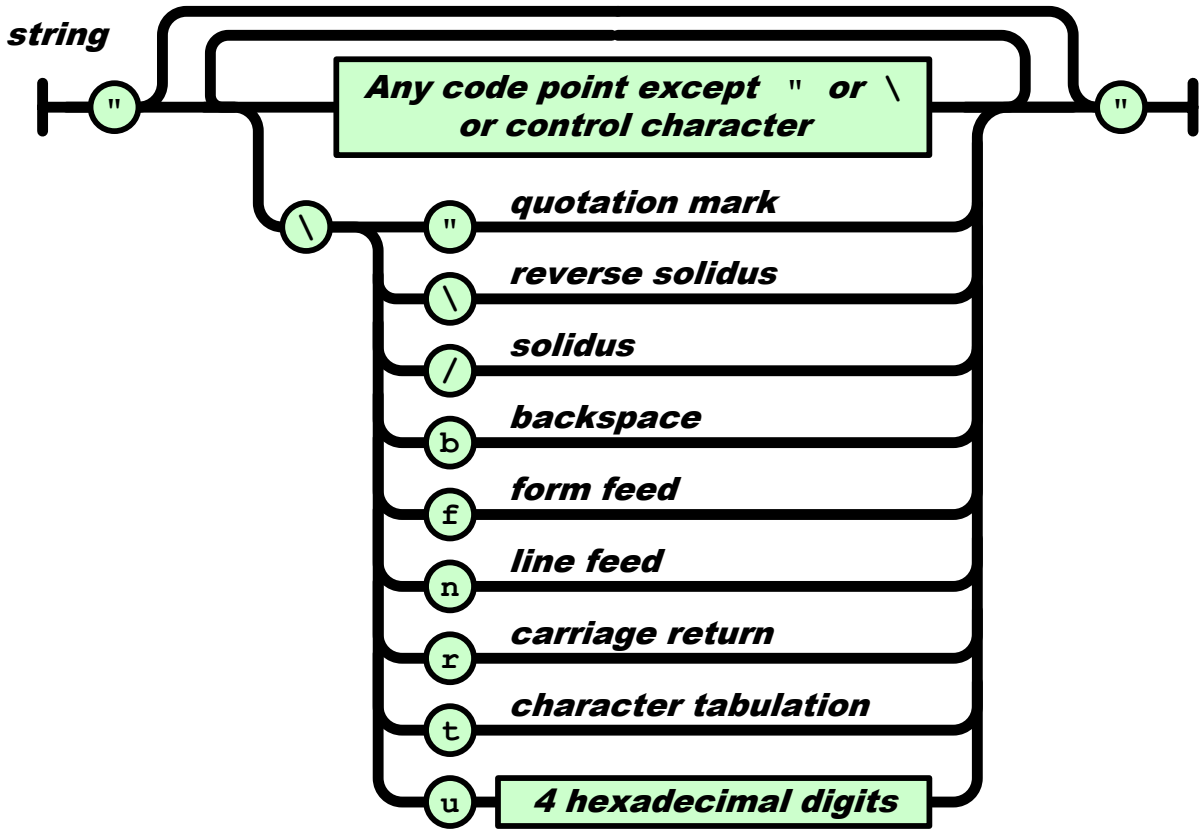

Figure 5 — string

Bibliography

[1] ECMA-262, *ECMAScript® Language Specification*

[2] IETF RFC 8259 *The JavaScript Object Notation (JSON) Data Interchange Format*