

Internet Engineering Task Force (IETF)
Request for Comments: 8259
Obsoletes: 7159
Category: Standards Track
ISSN: 2070-1721

T. Bray, Ed.
Textuality
December 2017

The JavaScript Object Notation (JSON) Data Interchange Format

Формат обмена данными JSON

Тезисы

JSON¹ представляет собой облегченный текстовый формат обмена данными, независимый от языка и разработанный на основе стандарта языков программирования сценариев ECMAScript. JSON определяет небольшой набор правил форматирования для переносимого представления структурированных данных.

В этом документе устраняются расхождения с другими спецификациями JSON, исправляются ошибки и приводится основанное на практике руководство по совместимости.

Статус документа

Документ относится к категории Internet Standards Track.

Документ является результатом работы IETF² и представляет согласованный взгляд сообщества IETF. Документ прошел открытое обсуждение и был одобрен для публикации IESG³. Не все одобренные IESG документы претендуют на статус Internet Standard (раздел 2 в RFC 7841).

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке <https://www.rfc-editor.org/info/rfc8259>.

Авторские права

Авторские права (Copyright (c) 2017) принадлежат IETF Trust и лицам, указанным в качестве авторов документа. Все права защищены.

Этот документ является субъектом прав и ограничений, перечисленных в BCP 78 и IETF Trust Legal Provisions и относящихся к документам IETF (<http://trustee.ietf.org/license-info>), на момент публикации данного документа. Прочтите упомянутые документы внимательно, поскольку в них описаны права и ограничения, относящиеся к данному документу. Фрагменты программного кода, включенные в этот документ, распространяются в соответствии с упрощенной лицензией BSD, как указано в параграфе 4.e документа Trust Legal Provisions, без каких-либо гарантий (как указано в Simplified BSD License).

Документ может содержать материалы из IETF Document или IETF Contribution, опубликованных или публично доступных до 10 ноября 2008 года. Лица, контролирующие авторские права на некоторые из таких документов, могли не предоставить IETF Trust права разрешать внесение изменений в такие документы за рамками процессов IETF Standards. Без получения соответствующего разрешения от лиц, контролирующих авторские права этот документ не может быть изменен вне рамок процесса IETF Standards, не могут также создаваться производные документы за рамками процесса IETF Standards за исключением форматирования документа для публикации или перевода с английского языка на другие языки.

Оглавление

1. Введение.....	2
1.1. Используемые соглашения.....	2
1.2. Спецификация JSON.....	2
1.3. Введение для этого выпуска.....	2
2. Грамматика JSON.....	2
3. Значения.....	3
4. Объекты.....	3
5. Массивы.....	3
6. Числа.....	3
7. Строки.....	4
8. Строки и символы.....	4
8.1. Кодировка символов.....	4
8.2. Символы Unicode.....	4
8.3. Сравнение строк.....	4
9. Анализаторы.....	4
10. Генераторы.....	5
11. Взаимодействие с IANA.....	5
12. Вопросы безопасности.....	5
13. Примеры.....	5
14. Литература.....	6

¹JavaScript Object Notation - обозначения объектов JavaScript.

²Internet Engineering Task Force.

³Internet Engineering Steering Group.

14.1. Нормативные документы.....	6
14.2. Дополнительная литература.....	6
Приложение А. Отличия от RFC 7159.....	6
Участники работы.....	7
Адрес автора.....	7

1. Введение

JSON представляет собой текстовый формат для публикации структурированных данных. Формат основан на литералах объектов JavaScript в соответствии с третьей редакцией стандарта ECMAScript [ECMA-262].

JSON может представлять 4 типа примитивов (строки - string, числа - number, логические элементы - boolean и пустые элементы - null), а также 2 структурированных типа (объекты - object и массивы - array).

Строка состоит из множества (возможно, пустого - 0) символов Unicode [UNICODE]. Отметим, что имеется в виду последняя, а не какая-либо из предшествующих версий Unicode. Не предполагается влияние возможных изменений спецификации Unicode на синтаксис JSON.

Объект представляет собой неупорядоченный набор (возможно, пустой) пар «имя-значение», где имя (name) является строкой, а значение (value) может быть строкой, числом, логическим или пустым элементом, объектом или массивом.

Массив представляет собой неупорядоченный (возможно, пустой) набор значений.

Термины «объект» и «массив» следуют соглашениям JavaScript.

Целью разработки JSON была организация минимального и переносимого текстового подмножества JavaScript.

1.1. Используемые соглашения

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **не рекомендуется** (NOT RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с BCP 14 [RFC2119] [RFC8174] тогда и только тогда, когда они выделены шрифтом, как показано здесь.

Грамматические правила в этом документе интерпретируются в соответствии с [RFC5234].

1.2. Спецификация JSON

Этот документ заменяет собой [RFC7159], который был заменой [RFC4627], содержавшего исходное описание JSON и регистрацию типа носителя application/json.

JSON описан также в [ECMA-404].

Ссылка на ECMA-404 в предыдущем предложении является нормативной, а не просто указанием разработчикам на ознакомление с документом для понимания данной спецификации. Эта ссылка подчеркивает отсутствие несоответствий в определении термина «текст JSON» в спецификациях. Однако следует отметить, что ECMA-404 допускает несколько методов, применения которых данная спецификация рекомендует избегать в интересах максимальной совместимости.

Цель заключается в соблюдении обоими документами одинаковой грамматики, несмотря на использование разных описаний. При обнаружении различий ECMA и IETF будут совместно работать над обновлением обоих документов.

При обнаружении ошибки в одном из документов следует проверить наличие подобной ошибки в другом и при обнаружении такой ошибки следует внести исправление, если это возможно.

Если какой-либо из документов будет впоследствии изменен, ECMA и IETF будут совместно работать над обеспечением согласованности обоих документов.

1.3. Введение для этого выпуска

За годы с момента публикации RFC 4627 формат JSON получил очень широкое распространение. Опыт показал, что некоторые формы, разрешенные спецификацией, могут вызывать проблемы при взаимодействии.

Кроме того, было обнаружено некоторое число ошибок в RFC 4627 (см. [Err607] и 3607 [Err3607]), а затем и в RFC 7159 (см. [Err3915], [Err4264], [Err4336] и [Err4388]).

Целью этого документа является исправление ошибок, устранение несоответствий с другими спецификациями JSON и указание применений, которые могут вызывать проблемы при взаимодействии.

2. Грамматика JSON

Текст JSON представляет последовательность маркеров (token). Набор маркеров включает 6 структурных символов, строки, числа и 3 имени литералов.

Текст JSON является последовательностью. Отметим, что в некоторых прежних спецификациях JSON текст JSON считался объектом или массивом. Реализации, которые создают лишь совместимые с текстом JSON объекты и массивы, будут совместимы в том смысле, что все реализации будут воспринимать их как корректные тексты JSON.

JSON-text = ws value ws

Ниже перечислены 6 структурных символов

begin-array	= ws %x5B ws ;	[левая квадратная скобка, начало массива
begin-object	= ws %x7B ws ;	{ левая фигурная скобка, начало объекта
end-array	= ws %x5D ws ;] правая квадратная скобка, конец массива
end-object	= ws %x7D ws ;	} правая фигурная скобка, конец объекта

```
name-separator = ws %x3A ws ; : двоеточие
value-separator = ws %x2C ws ; , запятая
```

Можно применять незначимые пробельные символы (whitespace) перед любым структурным символом и после него.

```
ws = *(
    %x20 / ; пробел
    %x09 / ; горизонтальная табуляция
    %x0A / ; перевод строки или новая строка
    %x0D ) ; возврат каретки
```

3. Значения

Значение JSON **должно** быть объектом, массивом, числом, строкой или одним из трех приведенных ниже литералов.

```
false
null
true
```

Имена литералов **должны** указываться в нижнем регистре. Использование других литералов не допускается.

```
value = false / null / true / object / array / number / string
false = %x66.61.6c.73.65 ; ложь
null = %x6e.75.6c.6c ; пусто
true = %x74.72.75.65 ; истина
```

4. Объекты

Структура объекта представляется парой фигурных скобок, внутри которых могут содержаться пары «имя-значение» (элементы - member). Имя является строкой и отделяется от значения одним символом двоеточия (:). Одиночный символ запятой (,) отделяет значение от следующего за ним имени. Именам в объекте **следует** быть уникальными.

```
object = begin-object [ member *( value-separator member ) ]
        end-object
member = string name-separator value
```

Объект, все имена в котором уникальны, является функционально совместимым в том смысле, что все реализации, получившие объект, будут принимать его отображения «имя-значение». Если имена в объекте не уникальны, поведение получившей объект программы не предсказуемо. Многие реализации сообщают лишь последнюю пару «имя-значение», а другие будут сообщать об ошибке или не смогут разобрать объект, а некоторые реализации будут сообщать все пары «имя-значение», включая дубликаты имен.

Отмечено, что библиотеки для анализа JSON различаются в части упорядочения объектов, видимых вызывающей программе. Реализации, чье поведение не зависит от порядка элементов, будут совместимы в том смысле, что изменение порядка не будет влиять на них.

5. Массивы

Структура массива представляется парой квадратных скобок внутри которых могут содержаться значения (элементы), разделяемые запятыми.

```
array = begin-array [ value *( value-separator value ) ] end-array
```

Значения массива не обязаны иметь одинаковые типы.

6. Числа

Представление чисел похоже на используемое в большинстве языков программирования. Числа указываются по основанию 10 с использованием десятичных цифр. Число включает целую часть, которой может предшествовать знак «минус» (-), а далее может следовать дробная и/или экспоненциальная часть. Нули перед целой частью не допускаются.

Дробная часть представляет десятичную точку, за которой следует одна или множество цифр.

Экспоненциальная часть начинается с буквы «Е» или «е», за которой может следовать знак «плюс» (+) или «минус» (-), а далее - одна или множество цифр.

Численные значения, которые не могут быть представлены в показанной ниже грамматике (такие, как бесконечность или NaN - не число), не допускаются.

```
number = [ minus ] int [ frac ] [ exp ]
decimal-point = %x2E ; .
digit1-9 = %x31-39 ; 1-9
e = %x65 / %x45 ; e E
exp = e [ minus / plus ] 1*DIGIT
frac = decimal-point 1*DIGIT
int = zero / ( digit1-9 *DIGIT )
minus = %x2D ; -
plus = %x2B ; +
zero = %x30 ; 0
```

Эта спецификация позволяет реализациям вносить ограничения для диапазона и точности чисел. Поскольку программы, реализующие числа двойной точности IEEE 754 binary64 [IEEE754], доступны и применяются широко, для эффективного взаимодействия реализациям не следует ожидать большего диапазона или точности, чем они обеспечивают, в смысле точности аппроксимации чисел JSON. Числа JSON, такие как 1E400 или 3.141592653589793238462643383279, могут приводить к несовместимости, поскольку они предполагают, что создавшее их приложение ждет от принимающей программы слишком широкие возможности в плане величины и точности чисел.

Отметим, при использовании программ целые числа из диапазона $[-(2^{53})+1, (2^{53})-1]$ не вызовут проблем взаимодействия в том смысле, что реализации будут воспринимать эти значения.

7. Строки

Представление строк похоже на соглашения, используемые в семействе языков программирования C. Строки начинаются и заканчиваются символами кавычек. Все символы Unicode должны размещаться внутри кавычек, за исключением символов, которые **должны** использовать escape-кодирование - кавычки, обратная дробная черта и символы управления от U+0000 до U+001F.

Для всех символов может применяться escape-кодирование. Если символ относится к Basic Multilingual Plane (U+0000 - U+FFFF), он может быть представлен 6-символьной последовательностью из обратной дробной черты, буквы u и в нижнем регистре и четырех шестнадцатеричных цифр, указывающих код символа. Шестнадцатеричные цифра от A до F могут указываться в любом регистре. Например, строка, содержащая лишь символ обратной дробной черты, может быть представлена в форме "\u005C".

Для некоторых популярных символов имеется двухсимвольный вариант escape-кодирования. Например, строку из одиночного символа обратной дробной черты можно представить в виде "\\".

Escape-кодирование расширенных символов, не входящих в Basic Multilingual Plane, использует 12-символьное представление, кодирующее суррогатную пару UTF-16. Например, строка, содержащая лишь символ G clef (U+1D11E) может быть представлена в виде "\uD834\uDD1E".

```
string = quotation-mark *char quotation-mark
char = unescaped /
      escape (
        %x22 /           ; "      кавычка                U+0022
        %x5C /           ; \      обратная дробная черта    U+005C
        %x2F /           ; /      дробная черта            U+002F
        %x62 /           ; b      «забой»                  U+0008
        %x66 /           ; f      перевод страницы          U+000C
        %x6E /           ; n      перевод строки            U+000A
        %x72 /           ; r      возврат каретки           U+000D
        %x74 /           ; t      табуляция                 U+0009
        %x75 4HEXDIG ) ; uXXXX    U+XXXX
escape = %x5C           ; \
quotation-mark = %x22   ; "
unescaped = %x20-21 / %x23-5B / %x5D-10FFFF
```

8. Строки и символы

8.1. Кодировка символов

Тексты JSON, передаваемые между системами, не являющимися частью замкнутой экосистемы, **должны** кодироваться с использованием UTF-8 [RFC3629].

Прежние спецификации JSON не требовали использования кодировки UTF-8 при передаче текста JSON. Однако в большинстве реализаций программ на основе JSON используется кодировка UTF-8, поскольку лишь она обеспечивает совместимость.

Реализациям **недопустимо** добавлять метку порядка байтов (U+FEFF) в начале передаваемого через сеть текста JSON. Для обеспечения совместимости реализации, выполняющие анализ текста JSON, **могут** игнорировать метку порядка байтов, не считая ее ошибкой.

8.2. Символы Unicode

Когда все строки в тексте JSON состоят лишь из символов Unicode [UNICODE] (хотя и в escape-форме), этот текст JSON будет совместимым в том смысле, что все реализации программ, которые будут его анализировать, получат совпадающие строки имен и значений в объектах и массивах.

Однако ABNF в этой спецификации разрешает имена элементов и значения строк с битовыми последовательностями, которые не могут быть представлены символами Unicode, например, "\uDEAD" (один непарный суррогат UTF-16). Такие экземпляры могут встречаться, например, при отсечке библиотекой строк UTF-16 без проверки разрыва суррогатных пар. Поведение программы, получившей текст JSON с такими значениями, не предсказуемо. Например, реализации могут возвращать разные значения для размера строк или даже сталкиваться с критическими ошибками при выполнении.

8.3. Сравнение строк

Программная реализация обычно требует проверки совпадения имен. Реализации, которые преобразуют текстовое представление в последовательность кодовых блоков Unicode, а затем выполняют численное поблочное сравнение, будут совместимы в том смысле, что результаты сравнения строк будут совпадать. Если же реализация будет сравнивать строки с escape-символами без преобразования, она ошибочно сочтет строки "a\b" и "a\u005Cb" разными.

9. Анализаторы

Анализатор JSON преобразует текст JSON в иное представление. Анализатор JSON **должен** воспринимать все тексты, соответствующие грамматике JSON, и **может** воспринимать на являющиеся JSON формы и расширения.

Реализация может ограничивать размер воспринимаемого текста, а также уровень вложенности. Допускаются ограничения для диапазона и точности числовых значений, а также для размеров строк и кодировки символов в строках.

10. Генераторы

Генераторы JSON создают тексты JSON. Текст **должен** строго соблюдать грамматику JSON.

11. Взаимодействие с IANA

Для текста JSON применяется тип носителя application/json.

```
Type name: application
Subtype name: json
Required parameters: n/a
Optional parameters: n/a
Encoding considerations: binary
Security considerations: See RFC 8259, Section 12
Interoperability considerations: Described in RFC 8259
Published specification: RFC 8259
Applications that use this media type:
    JSON has been used to exchange data between applications written
    in all of these programming languages: ActionScript, C, C#,
    Clojure, ColdFusion, Common Lisp, E, Erlang, Go, Java, JavaScript,
    Lua, Objective CAML, Perl, PHP, Python, Rebol, Ruby, Scala, and
    Scheme.
Additional information:
    Magic number(s): n/a
    File extension(s): .json
    Macintosh file type code(s): TEXT
Person & email address to contact for further information:
    IESG
    <iesg@ietf.org>
Intended usage: COMMON
Restrictions on usage: none
Author:
    Douglas Crockford
    <douglas@crockford.com>
Change controller:
    IESG
    <iesg@ietf.org>
Note: No "charset" parameter is defined for this registration.
    Adding one really has no effect on compliant recipients.1
```

12. Вопросы безопасности

Обычно с языками сценариев связаны те или иные вопросы безопасности. JSON является подмножеством JavaScript но не включает операций присваивания и вызовов.

Поскольку синтаксис JSON заимствован из JavaScript, можно использовать функцию eval() этого языка для анализа большинства текстов JSON (но не всех, поскольку такие символы как U+2028 LINE SEPARATOR и U+2029 PARAGRAPH SEPARATOR разрешены в JSON, но не приемлемы в JavaScript). В общем случае использование этой функции считается недопустимым риском, поскольку тексты могут содержать исполняемый код наряду с объявлениями данных. То же самое относится к использованию похожих на eval() функций в любом языке программирования, для которого тексты JSON соответствуют синтаксису языка.

13. Примеры

Ниже представлен объект JSON

```
{
  "Image": {
    "Width": 800,
    "Height": 600,
    "Title": "View from 15th Floor",
    "Thumbnail": {
      "Url": "http://www.example.com/image/481989943",
      "Height": 125,
      "Width": 100
    },
    "Animated" : false,
    "IDs": [116, 943, 234, 38793]
  }
}
```

Здесь элемент Image является объектом, чей элемент Thumbnail является объектом, а элемент IDs - массивом чисел.

Ниже представлен массив JSON, содержащий два объекта.

```
[
  {
    "precision": "zip",
    "Latitude": 37.7668,
```

¹Параметр charset не задан в этой регистрации, но его добавление не будет влиять на поведение совместимых со спецификацией получателей.

```

    "Longitude": -122.3959,
    "Address": "",
    "City": "SAN FRANCISCO",
    "State": "CA",
    "Zip": "94107",
    "Country": "US"
  },
  {
    "precision": "zip",
    "Latitude": 37.371991,
    "Longitude": -122.026020,
    "Address": "",
    "City": "SUNNYVALE",
    "State": "CA",
    "Zip": "94085",
    "Country": "US"
  }
]

```

Ниже представлены короткие тексты JSON, содержащие только значение.

```

"Hello world!"
42
true

```

14. Литература

14.1. Нормативные документы

- [ECMA-404] Ecma International, "The JSON Data Interchange Format", Standard ECMA-404, <<http://www.ecma-international.org/publications/standards/Ecma-404.htm>>.
- [IEEE754] IEEE, "IEEE Standard for Floating-Point Arithmetic", IEEE 754.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, [RFC 2119](https://www.rfc-editor.org/info/rfc2119), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <<https://www.rfc-editor.org/info/rfc3629>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, [RFC 5234](https://www.rfc-editor.org/info/rfc5234), DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, [RFC 8174](https://www.rfc-editor.org/info/rfc8174), DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [UNICODE] The Unicode Consortium, "The Unicode Standard", <<http://www.unicode.org/versions/latest/>>.

14.2. Дополнительная литература

- [ECMA-262] Ecma International, "ECMAScript Language Specification", Standard ECMA-262, Third Edition, December 1999, <<http://www.ecma-international.org/publications/files/ECMA-ST-ARCH/ECMA-262.%203rd%20edition.%20December%201999.pdf>>.
- [Err3607] RFC Errata, Erratum ID 3607, RFC 4627, <<https://www.rfc-editor.org/errata/eid3607>>.
- [Err3915] RFC Errata, Erratum ID 3915, RFC 7159, <<https://www.rfc-editor.org/errata/eid3915>>.
- [Err4264] RFC Errata, Erratum ID 4264, RFC 7159, <<https://www.rfc-editor.org/errata/eid4264>>.
- [Err4336] RFC Errata, Erratum ID 4336, RFC 7159, <<https://www.rfc-editor.org/errata/eid4336>>.
- [Err4388] RFC Errata, Erratum ID 4388, RFC 7159, <<https://www.rfc-editor.org/errata/eid4388>>.
- [Err607] RFC Errata, Erratum ID 607, RFC 4627, <<https://www.rfc-editor.org/errata/eid607>>.
- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, DOI 10.17487/RFC4627, July 2006, <<https://www.rfc-editor.org/info/rfc4627>>.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<https://www.rfc-editor.org/info/rfc7159>>.

Приложение А. Отличия от RFC 7159

В этом приложении указаны различия между этим документом и текстом RFC 7159.

- Параграф 1.2 обновлен с учетом удаления спецификации JSON из ECMA-262, добавлена нормативная ссылка на ECMA-404 и разъяснено понятие «нормативная».
- Параграф 1.3 был обновлен с учетом ошибок в RFC 7159 (но не в RFC 4627).
- Параграф 8.1 был изменен с учетом требования кодировки UTF-8 при передаче через сеть.
- Раздел 12 был обновлен с уточнением описания рисков безопасности, связанных с применением функции ECMAScript "eval()".
- Параграф 14.1 был обновлен с включением нормативной ссылки на ECMA-404.

- Параграф 14.2 был обновлен с удалением ECMA-404, обновлением версии ECMA-262 и списка ошибок.

Участники работы

Автором RFC 4627 был Douglas Crockford. Данный документ вносит сравнительно небольшое число изменений, поэтому основная часть текста является результатом его работы.

Адрес автора

Tim Bray (редактор)

Textuality

Email: tbray@textuality.com

Перевод на русский язык

Николай Малых

nmalykh@protocols.ru