Бесплатная электронная книга

УЧУСЬ JSON

Free unaffiliated eBook created from
Stack Overflow contributors.

#json

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: json

It is an unofficial and free JSON ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official JSON.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с JSON

замечания

JSON (JavaScript Object Notation) - один из самых популярных и широко распространенных форматов обмена данными, первоначально описанных Дугласом Крокфордом.

В настоящее время он описывается двумя конкурирующими стандартами: RFC 71592 и ECMA-404 . Стандарт ECMA минимален, описывая только разрешенный синтаксис грамматики, тогда как RFC также предоставляет некоторые семантические и соображения безопасности.

- JSON широко распространен в программном обеспечении, которое включает в себя архитектуру клиент-сервер для обмена данными между клиентом и сервером.
- JSON прост в использовании и имеет чисто текстовый, легкий и удобный для чтения формат, и люди часто неправильно понимают как замену XML.
- Хотя аббревиатура начинается с JavaScript, JSON не является языком или имеет какие-либо языковые литералы, это просто спецификация для обозначения данных.
- Это независимая от платформы и языка и встроенная поддержка практически всех языков / фреймворков первой линии, таких как поддержка формата данных JSON, доступна на всех популярных языках, некоторые из которых являются C #, PHP, Java, C ++, Python, Ruby и многое другое.
- Официальным типом интернет-медиа для JSON является application / json.
- Расширение имени файла JSON является .json.

Версии

Поскольку у JSON нет никаких обновлений, есть только 1 версия JSON, ссылка ниже перенаправляется на исходный документ RFC, который является RFC 4627.

Версия	Дата выхода
оригинал	2006-07-28

Examples

Правила синтаксиса JSON

Синтаксис JSON (JavaScript Object Notation) основан на подмножестве JavaScript (см. Также json.org).

Действительным выражением JSON может быть один из следующих типов данных

- простые типы данных: String, Number, Boolean, Null
- составные типы данных: значение, объект, массив

Простые типы данных

Строка JSON должна быть заключена в двойные кавычки и может содержать ноль или более символов Unicode; Разрешения обратной косой черты разрешены. Принятые номера JSON находятся в нотации E. Булевы - одно из `true` , `false` . Null - зарезервированное ключевое слово `null` .

Тип данных	Примеры действительных JSON
### String	`"apple"`
	`"□□"`
	`"\u00c4pfel\n"`
	`""`
### Число	`3`
	`1.4`
	`-1.5e3`
### Boolean	`true`
	`false`
### Ноль	`null`

Композитные типы данных

Значение

Значение JSON может быть одним из следующих: String, Number, Boolean, Null, Object, Array.

объект

Объект JSON является разделенным запятыми неупорядоченным набором пар name: value, заключенных в фигурные скобки, где name - это String и значение JSON.

массив

JSON Array - упорядоченная коллекция значений JSON.

Пример массива JSON:

```
["home", "wooden"]
```

Примеры объектов JSON:

```
{
    "id": 1,
    "name": "A wooden door",
    "price": 12.50,
    "tags": ["home", "wooden"]
}
```

```
[
  1,
  2,
  [3, 4, 5, 6],
  {
    "id": 1,
    "name": "A wooden door",
    "price": 12.50,
    "tags": ["home", "wooden"]
  }
]
```

Онлайн-инструменты для проверки и форматирования данных JSON:

- http://jsonlint.com/
- http://www.freeformatter.com/json-validator.html
- http://jsonviewer.stack.hu/
- http://json.parser.online.fr/

Объект JSON

Объект JSON окружен фигурными фигурными скобками и содержит пары ключ-значение.

```
{ "key1": "value1", "key2": "value2", ... }
```

Ключи должны быть действительными строками, окруженными двойными кавычками. Значения могут быть объектами JSON, числами, строками, массивами или одним из следующих «буквальных имен»: `false` , `null` или `true` . В паре ключ-значение ключ отделяется от значения двоеточием. Несколько пар ключ-значение разделяются запятыми.

Заказ на объекты не важен. Таким образом, следующий объект JSON эквивалентен приведенному выше:

```
{ "key2": "value2", "key1": "value1", ... }
```

Чтобы суммировать все это, это пример действительного объекта JSON:

```
{
  "image": {
    "width":  800,
    "height": 600,
    "title":  "View from 15th Floor",
    "thumbnail": {
      "url": "http://www.example.com/image/481989943",
      "height": 125,
      "width":  100
    },
    "visible": true,
    "ids": [116, 943, 234, 38793]
  }
}
```

Общие примеры объектов JSON со связанными (Java) объектными аналогами

В этом примере предполагается, что объект «root», который сериализуется в JSON, является экземпляром следующего класса:

```
public class MyJson {
}
```

Пример 1: Пример экземпляра `MyJson` , а именно:

```
{}
```

т.е. поскольку класс не имеет полей, сериализуются только фигурные скобки. **Кудрявые скобки являются общими разделителями для представления объекта** . Обратите внимание также, как корневой объект не сериализуется как пара ключ-значение. Это справедливо и для простых типов (String, numbers, array), когда они не являются полями внешнего объекта.

Пример 2: добавим некоторые поля в `MyJson` и инициализируем их некоторыми значениями:

```
// another class, useful to show how objects are serialized when inside other objects
public class MyOtherJson {}

// an enriched version of our test class
public class MyJson {
  String myString = "my string";
  int myInt = 5;
  double[] myArrayOfDoubles = new double[] { 3.14, 2.72 };
  MyOtherJson objectInObject = new MyOtherJson();
}
```

Это связанное JSON-представление:

```
{
  "myString" : "my string",
  "myInt" : 5,
  "myArrayOfDoubles" : [ 3.14, 2.72 ],
  "objectInObject" : {}
}
```

Обратите внимание, как все поля сериализуются в структуре значений ключа, где ключ - это имя поля, содержащего значение. Общие разделители для массивов являются квадратными скобками. Также обратите внимание, что за каждой парой ключ-значение следует запятая, за исключением последней пары.

JSON Array

JSON Array - упорядоченный набор значений. Он окружен квадратными скобками, то есть `[]` , а значения разделены запятой:

```
{ "colors" : [ "red", "green", "blue" ] }
```

JSON Arrays также может содержать любой действительный элемент JSON, включая объекты, как в этом примере массива с двумя объектами (взятыми из документа RFC):

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

Они также могут содержать элементы со смешанными типами, например:

```
[
    "red",
    51,
    true,
    null,
    {
```

```
        "state": "complete"
    }
]
```

Распространенная ошибка при записи массивов JSON (и объектов) заключается в том, чтобы оставить конечную запятую после последнего элемента. Это распространенная картина на многих языках, но, к сожалению, она недействительна в JSON. Например, следующий массив недопустим:

```
[
    1,
    2,
]
```

Чтобы сделать это действительным, вам нужно будет удалить запятую после последнего элемента, превратив ее в:

```
[
    1,
    2
]
```

Редактирование JSON вручную

JSON - очень строгий формат (см. Http://json.org) . Это позволяет легко анализировать и писать для машин, но удивляет людей, когда незаметные ошибки ломают документ.

Общие проблемы

Trailing Comma

В отличие от большинства языков программирования, вы не можете добавлять запятую:

```
{
  a: 1,
  b: 2,
  c: 3
}
```

Добавление запятой после `3` приведет к ошибке синтаксиса.

Такая же проблема возникает для массивов:

```
[
  1,
  2
]
```

Вы должны проявлять особую осторожность, если вам нужно изменить порядок предметов.

Отсутствует запятая

```
{
  a: 1,
  b: 2,
  c: 3
  d: 4
}
```

Поскольку конечные запятые не допускаются, легко забыть добавить их перед добавлением нового значения (в этом случае после `3`).

Комментарии

JSON не допускает комментариев, поскольку это формат обмена данными. Это по-прежнему горячая тема, хотя и без ясных ответов, кроме как не использовать их.

Существует несколько способов:

- Используйте комментарии стиля C, а затем разделите их перед передачей в парсер
- Вставить комментарии в данные

```
{
  "//": "comment",
  "data": 1
}
```

- Вставить комментарии и переписать их с данными

```
{
  "data": "comment",
  "data": 1
}
```

Вторая запись `data` перезапишет комментарий *в большинстве парсеров* .

Решения

Чтобы упростить запись JSON, используйте IDE, которая проверяет наличие синтаксических ошибок и обеспечивает синтаксическую раскраску. Плагины доступны для большинства редакторов.

Когда вы разрабатываете приложения и инструменты, используйте JSON как внутри, так и

в качестве протокола, но старайтесь не раскрывать его в тех местах, где человеку, вероятно, потребуется отредактировать его вручную (кроме отладки).

Оцените другие форматы, которые лучше подходят для этого использования, например:

- Hjson , можно легко конвертировать в JSON и из него
- TOML , подобно файлам INI
- YAML , больше возможностей, но за счет дополнительной сложности

Обоснование для Array vs Object (т.е. когда использовать что)

Массивы JSON представляют собой совокупность объектов. В JS существует множество функций сбора, таких как `slice` , `pop` , `push` . Объекты имеют только более сырые данные.

JSONArray - *упорядоченная* последовательность *значений* . Его внешняя текстовая форма представляет собой строку, заключенную в квадратные скобки с запятыми, разделяющими значения.

JSONObject - это *неупорядоченный* набор пар *имя / значение* . Его внешняя форма - это строка, завернутая в фигурные скобки с двоеточиями между именами и значениями и запятыми между значениями и именами.

Object - ключ и значение, Array - цифры, строки, булевы. Когда вы используете то или это?

Вы можете думать о массивах как «is a / an», а Objects - «имеет». В качестве примера можно использовать «Фрукты». Каждый элемент в фруктовой корзине - это тип фруктов.

```
array fruit : [orange, mango, banana]
```

Массивы могут содержать объекты, строки, числа, массивы, но позволяют обрабатывать только объекты и массивы.

```
array fruit : [orange:[], mango:{}, banana:{}]
```

, Вы можете видеть, что оранжевый тоже массив. Это означает, что любой предмет, который идет внутри оранжевого, является оранжевым, скажем: bitter_orange, mandarin, sweet_orange.

для фруктового объекта любой предмет в нем является атрибутом плода. таким образом, у плода есть

```
object fruit :{seed:{}, endocarp:{},flesh:{}}
```

Это также подразумевает, что что-либо внутри семенного объекта должно быть свойством семени, скажем: цвет, ..

JSON - это прежде всего язык, который позволяет сериализовать объекты javascript в строки. Поэтому при десериализации строки JSON вы должны получить структуру

объектов javascript. Если ваш json десериализуется в объект, который хранит 100 объектов, называемых object1 to object100, тогда это будет очень неудобно. Большинство десериализаторов ожидают, что у вас будут известные объекты и массивы известных объектов, чтобы они могли преобразовать строки в фактическую структуру объекта на используемом вами языке. Также возникает вопрос, что философия объектно-ориентированного проектирования ответит вам.

кредиты всем участникам Каковы различия между использованием массивов JSON и объектов JSON?

Прочитайте Начало работы с JSON онлайн: https://riptutorial.com/ru/json/topic/889/начало-работы-с-json

глава 2: Stringify - Преобразование JSON в строку

параметры

Param	подробности
объект	(Объект) Объект JSON
Заменитель	(Функция | Array <string | number> - optiotinal) filter Функция | массив
Космос	(Номер | строка - необязательно). Количество пробелов в JSON

Examples

Преобразование простого объекта JSON в простую строку

```
var JSONObject = {
   stringProp: 'stringProp',
   booleanProp: false,
   intProp: 8
}

var JSONString = JSON.stringify(JSONObject);
console.log(JSONString);
/* output
 * {"stringProp":"stringProp","booleanProp":false,"intProp":8}
 */
```

Стрингировать с фильтром

```
var JSONObject = {
   stringProp: 'stringProp',
   booleanProp: false,
   intProp: 8
}

var JSONString = JSON.stringify(JSONObject, ['intProp']);
console.log(JSONString);
/* output
 * {"intProp":8}
 */
```

Строка с белым пространством

```
var JSONObject = {
```

```
   stringProp: 'stringProp',
   booleanProp: false,
   intProp: 8
}

var JSONString = JSON.stringify(JSONObject, null, 2);
console.log(JSONString);
/* output:
 *  {
 *    "stringProp": "stringProp",
 *    "booleanProp": false,
 *    "intProp": 8
 *   }
 */
```

Прочитайте Stringify - Преобразование JSON в строку онлайн:
https://riptutorial.com/ru/json/topic/7824/stringify---преобразование-json-в-строку

глава 3: Разбор строки JSON

Examples

Строка Parse JSON с использованием библиотеки com.google.gson в Java

Для использования этого кода необходимо добавить библиотеку `com.google.gson` .

Вот пример строки:

```
String companyDetails = {"companyName":"abcd","address":"abcdefg"}
```

Строки JSON могут быть проанализированы с использованием синтаксиса Java:

```
JsonParser parser =  new JsonParser();
JsonElement jsonElement = parser.parse(companyDetails);
JsonObject jsonObj = jsonElement.getAsJsonObject();
String comapnyName = jsonObj.get("companyName").getAsString();
```

Разбор строки JSON в JavaScript

В JavaScript объект `JSON` используется для анализа строки JSON. Этот метод доступен только в современных браузерах (IE8 +, Firefox 3.5+ и т. Д.).

Когда синтаксическая строка JSON обрабатывается, результатом будет объект JavaScript, массив или другое значение.

```
JSON.parse('"bar of foo"')
// "bar of foo" (type string)
JSON.parse("true")
// true (type boolean)
JSON.parse("1")
// 1 (type number)
JSON.parse("[1,2,3]")
// [1, 2, 3] (type array)
JSON.parse('{"foo":"bar"}')
// {foo: "bar"} (type object)
JSON.parse("null")
// null (type object)
```

Недопустимые строки будут вызывать ошибку JavaScript

```
JSON.parse('{foo:"bar"}')
// Uncaught SyntaxError: Unexpected token f in JSON at position 1
JSON.parse("[1,2,3,]")
// Uncaught SyntaxError: Unexpected token ] in JSON at position 7
JSON.parse("undefined")
// Uncaught SyntaxError: Unexpected token u in JSON at position 0
```

Метод `JSON.parse` включает в себя необязательную функцию `JSON.parse` которая может ограничивать или изменять результат анализа

```
JSON.parse("[1,2,3,4,5,6]", function(key, value) {
  return value > 3 ? '' : value;
})
// [1, 2, 3, "", "", ""]

var x = {};
JSON.parse('{"a":1,"b":2,"c":3,"d":4,"e":5,"f":6}', function(key, value) {
  if (value > 3) { x[key] = value; }
})
// x = {d: 4, e: 5, f: 6}
```

В последнем примере `JSON.parse` возвращает `undefined` значение. Чтобы предотвратить это, верните `value` внутри функции reviver.

Parse JSON файл с Groovy

Предположим, что мы имеем следующие данные JSON:

```
{
    "TESTS":
    [
         {
              "YEAR": "2017",
              "MONTH": "June",
              "DATE": "28"
         }
    ]
}
```

import groovy.json.JsonSlurper

класс JSONUtils {

```
private def data;
private def fileName = System.getProperty("jsonFileName")

public static void main(String[] args)
{
    JSONUtils jutils = new JSONUtils()
    def month = jutils.get("MONTH");
}
```

Ниже представлен синтаксический анализатор:

```
private parseJSON(String fileName = "data.json")
{
    def jsonSlurper = new JsonSlurper()
    def reader

    if(this.fileName?.trim())
    {
```

```
        fileName = this.fileName
    }

    reader = new BufferedReader(new InputStreamReader(new FileInputStream(fileName),"UTF-8"));
    data = jsonSlurper.parse(reader);
    return data
}

def get(String item)
{
    def result = new ArrayList<String>();
    data = parseJSON()
    data.TESTS.each{result.add(it."${item}")}
    return  result
}
```

}

Прочитайте Разбор строки JSON онлайн: https://riptutorial.com/ru/json/topic/3878/разбор-строки-json

кредиты

S. No	Главы	Contributors
1	Начало работы с JSON	AndriuZ, Asaph, ccprog, Community, depperm, francesco foresti , gandreadis, Gavishiddappa Gadagi, itzmukeshy7, laktak, Mirec Miskuf, Nilanchala Panigrahy, pickypg, reidzeibel, user2314737, Vitor Baptista
2	Stringify - Преобразование JSON в строку	Mosh Feu
3	Разбор строки JSON	Kruti Patel, Mahbub Rahman, Mottie, Vitor Baptista