Kostenloses eBook

LERNEN JSON

Free unaffiliated eBook created from
Stack Overflow contributors.

#json

Inhaltsverzeichnis

Über

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

Kapitel 1: Erste Schritte mit JSON

Bemerkungen

JSON (JavaScript Object Notation) ist eines der beliebtesten und allgemein akzeptierten Datenaustauschformate, das ursprünglich von Douglas Crockford angegeben wurde.

Es wird derzeit von zwei konkurrierenden Standards, RFC 71592 und ECMA-404, beschrieben . Der ECMA-Standard ist minimal und beschreibt nur die zulässige Grammatik-Syntax, während der RFC auch einige semantische und Sicherheitsaspekte enthält.

- JSON wird weithin in Software akzeptiert, die eine Client-Server-Architektur für den Datenaustausch zwischen Client und Server umfasst.
- JSON ist ein benutzerfreundliches und rein textbasiertes, leichtes und für den Menschen lesbares Format, und die Leute missverstehen oft den Austausch von XML.
- Obwohl die Abkürzung mit JavaScript beginnt, ist JSON keine Sprache oder Sprachliterale, sondern lediglich eine Spezifikation für die Notation von Daten.
- Es ist plattform- und sprachunabhängig und wird von fast allen Frontline-Sprachen / Frameworks unterstützt. Das JSON-Datenformat ist in allen gängigen Sprachen verfügbar, darunter C #, PHP, Java, C ++, Python, Ruby und viele mehr.
- Der offizielle Internet-Medientyp für JSON ist application / json.
- Die JSON-Dateinamenerweiterung lautet .json.

Versionen

Da JSON keine Updates erhalten hat, gibt es nur eine Version von JSON. Der Link unten führt zum ursprünglichen RFC-Dokument (RFC 4627).

Ausführung	Veröffentlichungsdatum
Original	2006-07-28

Examples

JSON-Syntaxregeln

Die JSON-Syntax (JavaScript Object Notation) basiert auf einer Teilmenge von JavaScript (siehe auch json.org).

Ein gültiger JSON-Ausdruck kann einer der folgenden Datentypen sein

- einfache Datentypen: String, Number, Boolean, Null
- zusammengesetzte Datentypen: Wert, Objekt, Array

Einfache Datentypen

Eine JSON-Zeichenfolge muss in Anführungszeichen gesetzt werden und darf null oder mehr Unicode-Zeichen enthalten. Backslash-Fluchten sind erlaubt. Akzeptierte JSON-Nummern sind in E-Notation . Boolean ist `true` , `false` . NULL ist das reservierte Schlüsselwort `null` .

Datentyp	Beispiele für gültige JSON
### String	`"apple"`
	`"□□"`
	`"\u00c4pfel\n"`
	`""`
### Nummer	`3`
	`1.4`
	`-1.5e3`
### Boolean	`true`
	`false`
### Null	`null`

Zusammengesetzte Datentypen

Wert

Ein JSON-Wert kann einer der folgenden sein: String, Number, Boolean, Null, Object, Array.

Objekt

Ein JSON-Objekt ist eine durch Kommas getrennte, ungeordnete Auflistung von Name: Wert-Paaren, die in geschweiften Klammern eingeschlossen sind, wobei Name eine Zeichenfolge und Wert ein JSON-Wert ist.

Array

Ein JSON-Array ist eine geordnete Sammlung von JSON-Werten.

Beispiel eines JSON-Arrays:

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

Beispiele für JSON-Objekte:

```
{
```

```
    "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"]
  }
]
```

Online-Tools zum Validieren und Formatieren von JSON-Daten:

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

JSON-Objekt

Ein JSON-Objekt ist von geschweiften Klammern umgeben und enthält Schlüsselwertpaare.

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

Schlüssel müssen gültige Zeichenfolgen sein, also von doppelten Anführungszeichen umgeben. Werte können JSON-Objekte, Zahlen, Zeichenfolgen, Arrays oder einer der folgenden 'Literalnamen' sein: `false` , `null` oder `true` In einem Schlüssel-Wert-Paar wird der Schlüssel durch einen Doppelpunkt vom Wert getrennt. Mehrere Schlüsselwertpaare werden durch Kommas getrennt.

Reihenfolge in Objekten ist nicht wichtig. Das folgende JSON-Objekt entspricht somit dem obigen:

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

Zusammenfassend ist dies ein Beispiel für ein gültiges JSON-Objekt:

```
{
  "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]
  }
}
```

Häufige Beispiele für JSON-Objekte mit zugehörigen Gegenstücken (Java)

In diesem Beispiel wird davon ausgegangen, dass das 'root'-Objekt, das in JSON serialisiert wird, eine Instanz der folgenden Klasse ist:

```
public class MyJson {
}
```

Beispiel 1: Ein Beispiel für eine Instanz von `MyJson` :

```
{}
```

Da die Klasse keine Felder enthält, werden nur geschweifte Klammern serialisiert. **Geschweifte Klammern sind die üblichen Trennzeichen, um ein Objekt darzustellen** . Beachten Sie auch, wie das Stammobjekt nicht als Schlüssel-Wert-Paar serialisiert wird. Dies gilt auch für einfache Typen (String, Zahlen, Arrays), wenn es sich nicht um Felder eines (äußeren) Objekts handelt.

Beispiel 2: `MyJson` Sie einige Felder zu `MyJson` und initialisieren Sie sie mit einigen Werten:

```
// 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();
}
```

Dies ist die zugehörige JSON-Darstellung:

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

Beachten Sie, wie alle Felder in einer Schlüsselwertstruktur serialisiert werden, wobei der Schlüssel der Name des Felds ist, das den Wert enthält. Die üblichen Trennzeichen für Arrays sind eckige Klammern. Beachten Sie auch, dass auf jedes Schlüsselwertpaar ein Komma mit Ausnahme des letzten Paares folgt.

JSON-Array

Ein JSON-Array ist eine geordnete Sammlung von Werten. Es wird von eckigen Klammern, dh `[]` , umgeben und die Werte werden durch Kommas getrennt.

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

JSON-Arrays können auch ein beliebiges gültiges JSON-Element enthalten, einschließlich Objekte, wie in diesem Beispiel eines Arrays mit 2 Objekten (aus dem RFC-Dokument):

```
[
  {
     "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"
  }
]
```

Sie können auch Elemente mit gemischten Typen enthalten, zum Beispiel:

```
[
    "red",
    51,
    true,
    null,
    {
        "state": "complete"
    }
]
```

Ein häufiger Fehler beim Schreiben von JSON-Arrays (und -Objekten) besteht darin, nach dem letzten Element ein nachfolgendes Komma zu hinterlassen. Dies ist in vielen Sprachen üblich, ist aber in JSON leider nicht gültig. Das folgende Array ist beispielsweise ungültig:

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

Um dies zu bestätigen, müssen Sie das Komma nach dem letzten Element entfernen und es in Folgendes ändern:

```
[
    1,
    2
]
```

JSON von Hand bearbeiten

JSON ist ein sehr striktes Format (siehe http://json.org) . Das macht es einfach, für Maschinen zu parsen und zu schreiben, überrascht aber die Menschen, wenn ein unauffälliger Fehler das Dokument bricht.

Allgemeine Probleme

Nachlaufendes Komma

Im Gegensatz zu den meisten Programmiersprachen dürfen Sie kein nachfolgendes Komma hinzufügen:

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

Das Hinzufügen eines Kommas nach `3` führt zu einem Synax-Fehler.

Dasselbe Problem besteht für Arrays:

```
[
  1,
  2
]
```

Sie müssen besonders vorsichtig sein, wenn Sie die Artikel nachbestellen müssen.

Fehlendes Komma

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

Da nachfolgende Kommas nicht zulässig sind, kann man leicht vergessen, vor dem Hinzufügen

eines neuen Wertes eines anzuhängen (in diesem Fall nach `3`).

Bemerkungen

JSON erlaubt keine Kommentare, da es sich um ein Datenaustauschformat handelt. Dies ist immer noch ein heißes Thema, jedoch ohne klare Antworten, außer, sie nicht zu verwenden.

Es gibt mehrere Problemumgehungen:

- Verwenden Sie Kommentare im C-Stil und entfernen Sie sie, bevor Sie sie an den Parser übergeben
- Kommentare in die Daten einbetten

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

- Kommentare einbetten und mit Daten überschreiben

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

Die zweite `data` überschreibt den Kommentar *in den meisten Parsern* .

Lösungen

Um das Schreiben von JSON zu vereinfachen, verwenden Sie eine IDE, die auf Syntaxfehler überprüft und Syntaxfärbung liefert. Plugins sind für die meisten Editoren verfügbar.

Wenn Sie Anwendungen und Tools entwickeln, verwenden Sie JSON intern und als Protokoll. Versuchen Sie jedoch, das JSON nicht an Stellen offenzulegen, an denen ein Benutzer es wahrscheinlich manuell bearbeiten müsste (außer beim Debugging).

Bewerten Sie andere Formate, die für diese Verwendung besser geeignet sind, wie zum Beispiel:

- Hjson , kann nahtlos in und aus JSON konvertiert werden
- TOML , ähnlich zu INI-Dateien
- YAML , mehr Funktionen, jedoch auf Kosten zusätzlicher Komplexität

Gründe für Array vs Object (dh wann soll was verwendet werden)

JSON-Arrays repräsentieren eine Sammlung von Objekten. In JS gibt es eine Reihe von Sammlungsfunktionen wie `slice` , `pop` , `push` . Objekte haben nur mehr Rohdaten.

Ein **JSONArray** ist eine *geordnete* Folge von *Werten* . Die externe Textform ist eine Zeichenfolge,

die in eckige Klammern eingeschlossen ist und durch Kommas getrennt ist.

Ein **JSONObject** ist eine *ungeordnete* Sammlung von *Name / Wert*- Paaren. Seine äußere Form ist eine Zeichenfolge, die in geschweifte Klammern eingeschlossen ist und zwischen den Namen und Werten Doppelpunkte und zwischen den Werten und Namen Kommas steht.

Objekt - Schlüssel und Wert, Array - Zahlen, Strings, Booleans. Wann benutzt du das oder das?

Sie können sich Arrays als "ist ein / ein" und Objekte als "hat ein" vorstellen. Als Beispiel verwenden wir "Fruit". Jeder Artikel in der Fruchtreihe ist eine Obstsorte.

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

Arrays können Objekte, Strings, Zahlen und Arrays enthalten, können jedoch nur mit Objekten und Arrays behandelt werden.

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

. Sie können sehen, dass Orange auch ein Array ist. Dies bedeutet, dass jedes Element, das in Orange geht, eine Orangetyp ist, beispielsweise: bitter_orange, mandarin, sweet_orange.

Bei Fruchtobjekten ist jedes Element darin ein Attribut von Obst. so hat die Frucht eine

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

Dies impliziert auch, dass alles innerhalb des Saatgutobjekts Eigentum des Saatguts sein sollte, beispielsweise: Farbe, ..

JSON ist in erster Linie eine Sprache, mit der Javascript-Objekte in Strings serialisiert werden können. Beim Deserialisieren eines JSON-Strings sollten Sie daher eine Javascript-Objektstruktur erhalten. Wenn Ihr Json in ein Objekt deserialisiert wird, das 100 Objekte mit dem Namen object1 in object100 speichert, ist das sehr umständlich. Die meisten Deserialisierer erwarten von Ihnen bekannte Objekte und Arrays bekannter Objekte, damit sie die Zeichenfolgen in die tatsächliche Objektstruktur in der von Ihnen verwendeten Sprache konvertieren können. Auch diese Frage würde Ihnen die Philosophie des objektorientierten Designs beantworten.

Kredite an alle Teilnehmer Was sind die Unterschiede zwischen der Verwendung von JSON-Arrays und JSON-Objekten?

Erste Schritte mit JSON online lesen: https://riptutorial.com/de/json/topic/889/erste-schritte-mit-json

Kapitel 2: JSON-String wird analysiert

Examples

Analysieren Sie die JSON-Zeichenfolge mithilfe der Bibliothek com.google.gson in Java

`com.google.gson` Verwendung dieses Codes `com.google.gson` Bibliothek `com.google.gson` hinzugefügt werden.

Hier ist der Beispielstring:

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

JSON-Zeichenfolgen können mit der folgenden Syntax in Java analysiert werden:

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

Analysieren Sie den JSON-String in JavaScript

In JavaScript wird das `JSON` Objekt zum Analysieren einer JSON-Zeichenfolge verwendet. Diese Methode ist nur in modernen Browsern verfügbar (IE8 +, Firefox 3.5+ usw.).

Wenn eine gültige JSON-Zeichenfolge analysiert wird, ist das Ergebnis ein JavaScript-Objekt, ein Array oder ein anderer Wert.

```
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)
```

Ungültige Zeichenfolgen lösen einen JavaScript-Fehler aus

```
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
```

Die `JSON.parse` Methode enthält eine optionale Reviver-Funktion, die das `JSON.parse` einschränken oder ändern kann

```
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}
```

Im letzten Beispiel gibt `JSON.parse` einen nicht `undefined` Wert zurück. Um dies zu verhindern, geben Sie den `value` innerhalb der Reviver-Funktion zurück.

JSON-Datei mit Groovy analysieren

Angenommen, wir haben die folgenden JSON-Daten:

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

import groovy.json.JsonSlurper

Klasse 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");
}
```

Unten ist der Parser:

```
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
}
```

}

JSON-String wird analysiert online lesen: https://riptutorial.com/de/json/topic/3878/json-string-wird-analysiert

Kapitel 3: Stringify - Konvertieren Sie JSON in String

Parameter

Param	Einzelheiten
Objekt	(Object) Das JSON-Objekt
Replacer	(Funktion | Array <string | number> - optiotinal) filter Funktion | Array
Platz	(Number | string - optional) Anzahl der Leerzeichen in der JSON

Examples

Konvertieren Sie ein einfaches JSON-Objekt in einen einfachen String

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

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

Mit Filter faden

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

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

Stringify mit Leerraum

```
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
 *   }
 */
```

Stringify - Konvertieren Sie JSON in String online lesen:
https://riptutorial.com/de/json/topic/7824/stringify---konvertieren-sie-json-in-string

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit JSON	AndriuZ, Asaph, ccprog, Community, depperm, francesco foresti , gandreadis, Gavishiddappa Gadagi, itzmukeshy7, laktak, Mirec Miskuf, Nilanchala Panigrahy, pickypg, reidzeibel, user2314737, Vitor Baptista
2	JSON-String wird analysiert	Kruti Patel, Mahbub Rahman, Mottie, Vitor Baptista
3	Stringify - Konvertieren Sie JSON in String	Mosh Feu