EBook Gratis

APRENDIZAJE JSON

Free unaffiliated eBook created from
Stack Overflow contributors.

#json

Tabla de contenido

Acerca de

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

Capítulo 1: Empezando con JSON

Observaciones

JSON (JavaScript Object Notation) es uno de los formatos de intercambio de datos más populares y ampliamente aceptados originalmente especificado por Douglas Crockford.

Actualmente está descrito por dos estándares en competencia, RFC 71592 y ECMA-404 . El estándar ECMA es mínimo, describe solo la sintaxis gramatical permitida, mientras que el RFC también proporciona algunas consideraciones semánticas y de seguridad.

- JSON es ampliamente aceptado en el software que incluye la arquitectura cliente-servidor para intercambiar datos entre el cliente y el servidor.
- JSON es fácil de usar y está puramente basado en texto, es ligero y tiene un formato legible para el usuario, y la gente a menudo malinterpreta como reemplazo de XML.
- Aunque la abreviatura comienza con JavaScript, JSON no es un idioma o tiene literales en cualquier idioma, es solo una especificación para la notación de datos.
- Es independiente de la plataforma y del idioma y está soportado en casi todos los lenguajes / marcos de primera línea, y el soporte para el formato de datos JSON está disponible en todos los lenguajes populares, algunos de los cuales son C #, PHP, Java, C ++, Python, Ruby y muchos más.
- El tipo de medio oficial de Internet para JSON es application / json.
- La extensión de nombre de archivo JSON es .json.

Versiones

Dado que JSON no tiene actualizaciones, solo hay 1 versión de JSON, el siguiente enlace redirige al documento RFC original, que es RFC 4627.

Versión	Fecha de lanzamiento
Original	2006-07-28

Examples

Reglas de sintaxis JSON

La sintaxis de JSON (Notación de objetos de JavaScript) se basa en un subconjunto de JavaScript (consulte también json.org).

Una expresión JSON válida puede ser uno de los siguientes tipos de datos

- tipos de datos simples: cadena, número, booleano, nulo
- Tipos de datos compuestos: Valor, Objeto, Array

Tipos de datos simples

Una cadena JSON debe incluirse entre comillas dobles y puede contener cero o más caracteres Unicode; Se permiten escapes de barra invertida. Los números JSON aceptados están en notación E Boolean es uno de los `true` , los `false` . Null es la palabra clave reservada `null` .

Tipo de datos	Ejemplos de JSON válido
### Cuerda	`"apple"`
	`"□□"`
	`"\u00c4pfel\n"`
	`""`
### número	`3`
	`1.4`
	`-1.5e3`
### Booleano	`true`
	`false`
### Nulo	`null`

Tipos de datos compuestos

Valor

Un valor JSON puede ser uno de los siguientes: String, Number, Boolean, Null, Object, Array.

Objeto

Un objeto JSON es una colección no ordenada separada por comas de pares nombre: valor encerrados entre paréntesis, donde nombre es una cadena y valor un valor JSON.

Formación

Una matriz JSON es una colección ordenada de valores JSON.

Ejemplo de una matriz JSON:

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

Ejemplos de objetos 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"]
  }
]
```

Herramientas en línea para validar y formatear datos JSON:

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

Objeto JSON

Un objeto JSON está rodeado de llaves y contiene pares clave-valor.

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

Las claves deben ser cadenas válidas, rodeadas por comillas dobles. Los valores pueden ser objetos JSON, números, cadenas, matrices o uno de los siguientes 'nombres literales': `false` , `null` o `true` . En un par clave-valor, la clave se separa del valor mediante dos puntos. Múltiples pares clave-valor están separados por comas.

El orden en los objetos no es importante. El siguiente objeto JSON es, por lo tanto, equivalente al anterior:

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

En resumen, este es un ejemplo de un objeto JSON válido:

```
{
  "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]
  }
}
```

Ejemplos comunes de objetos JSON, con contrapartes de objetos relacionados (Java)

A lo largo de este ejemplo, se supone que el objeto 'raíz' que se está serializando a JSON es una instancia de la siguiente clase:

```
public class MyJson {
}
```

Ejemplo 1: Un ejemplo de una instancia de `MyJson` , como es:

```
{}
```

es decir, como la clase no tiene campos, solo los corchetes están serializados. **Los corchetes son los delimitadores comunes para representar un objeto** . Observe también cómo el objeto raíz no se serializa como un par clave-valor. Esto también es válido para los tipos simples (cadena, números, matrices) cuando no son campos de un objeto (externo).

Ejemplo 2: `MyJson` algunos campos a `MyJson` e inicialicemos con algunos valores:

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

Esta es la representación JSON relacionada:

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

Observe cómo todos los campos se serializan en una estructura de clave-valor, donde la clave es el nombre del campo que contiene el valor. Los delimitadores comunes para matrices son los corchetes. Observe también que cada par clave-valor va seguido de una coma, excepto el último par.

JSON Array

Una matriz JSON es una colección ordenada de valores. Está rodeado por llaves cuadradas, es decir, `[]` , y los valores están delimitados por comas:

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

Las matrices JSON también pueden contener cualquier elemento JSON válido, incluidos los objetos, como en este ejemplo de una matriz con 2 objetos (tomados del documento 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"
  }
]
```

También pueden contener elementos con tipos mixtos, por ejemplo:

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

Un error común al escribir matrices JSON (y objetos) es dejar una coma al final del último elemento. Este es un patrón común en muchos idiomas, pero desafortunadamente no es válido en JSON. Por ejemplo, la siguiente matriz no es válida:

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

Para que esto sea válido, deberías eliminar la coma después del último elemento, convirtiéndolo en:

```
[
    1,
    2
]
```

Editando JSON a mano

JSON es un formato muy estricto (ver http://json.org) . Eso facilita el análisis y la escritura de las máquinas, pero sorprende a los humanos cuando un error discreto rompe el documento.

Problemas comunes

Coma que se arrastra

A diferencia de la mayoría de los lenguajes de programación, no se le permite agregar una coma al final:

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

Agregar una coma después de `3` producirá un error de synax.

El mismo problema existe para las matrices:

```
[
  1,
  2
]
```

Debe tener mucho cuidado si necesita reordenar los artículos.

Falta de coma

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

Como no se permiten las comas al final, es fácil olvidar agregar una antes de agregar un nuevo valor (en este caso, después de `3`).

Comentarios

JSON no permite comentarios ya que es un formato de intercambio de datos. Este sigue siendo un tema candente, aunque no hay respuestas claras más que no utilizarlas.

Hay varias soluciones:

- Use comentarios de estilo C, luego elimínelos antes de pasarlos al analizador
- Insertar comentarios en los datos

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

- Insertar comentarios y sobrescribirlos con datos.

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

La segunda entrada de `data` sobrescribirá el comentario *en la mayoría de los analizadores* .

Soluciones

Para facilitar la escritura de JSON, use un IDE que compruebe los errores de sintaxis y proporcione colores de sintaxis. Los complementos están disponibles para la mayoría de los editores.

Cuando desarrolle aplicaciones y herramientas, use JSON internamente y como protocolo, pero intente no exponerlo en lugares donde un humano probablemente deba editarlo manualmente (excepto para la depuración).

Evalúe otros formatos que sean más adecuados para este uso, como:

- Hjson , se puede convertir perfectamente desde y hacia JSON
- TOML , similar a los archivos INI
- YAML , más funciones pero a costa de una mayor complejidad.

Razón de ser de Array vs Objeto (es decir, cuándo usar qué)

Las matrices JSON representan una colección de objetos. En JS, hay un montón de funciones de colección fuera de ellas, como `slice` , `pop` , `push` . Los objetos tienen más datos en bruto.

Un **JSONArray** es una secuencia *ordenada* de *valores* . Su forma de texto externo es una cadena envuelta entre corchetes con comas que separan los valores.

Un **JSONObject** es una colección *desordenada* de pares de *nombre* / *valor* . Su forma externa es una cadena envuelta en llaves con dos puntos entre los nombres y valores, y comas entre los valores y los nombres.

Objeto - clave y valor, Array - números, cadenas, booleanos. ¿Cuándo usas esto o aquello?

Puedes pensar en Arrays como "es un / un" y Objetos como "tiene un". Vamos a usar "Fruta" como ejemplo. Cada artículo en una variedad de frutas es un tipo de fruta.

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

Las matrices pueden contener objetos, cadenas, números, matrices, pero tratamos solo con objetos y matrices.

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

. Puedes ver que la naranja es una matriz también. Implica que cualquier artículo que se ponga en naranja es un tipo de naranja, por ejemplo: bitter_orange, mandarin, sweet_orange.

Para el objeto de fruta, cualquier artículo en él es un atributo de fruta. así el fruto tiene una

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

Esto también implica que cualquier cosa dentro del objeto semilla debe ser propiedad de la semilla, digamos: color, ..

JSON es principalmente un lenguaje que permite serializar objetos javascript en cadenas. Así que al deserializar una cadena JSON debería obtener una estructura de objeto javascript. Si su json se deserializa en un objeto que almacena 100 objetos llamados object1 a object100, eso será muy inconveniente. La mayoría de los deserializadores esperarán que usted tenga objetos conocidos y matrices de objetos conocidos para que puedan convertir las cadenas en la estructura real del objeto en el idioma que está usando. También esta es una pregunta que la filosofía del diseño orientado a objetos le respondería.

créditos para todas las partes ¿Cuáles son las diferencias entre el uso de matrices JSON y los objetos JSON?

Lea Empezando con JSON en línea: https://riptutorial.com/es/json/topic/889/empezando-con-json

Capítulo 2: Analizando la cadena JSON

Examples

Analice la cadena JSON utilizando la biblioteca com.google.gson en Java

`com.google.gson` debe agregar la biblioteca `com.google.gson` para usar este código.

Aquí está la cadena de ejemplo:

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

Las cadenas JSON se pueden analizar utilizando la siguiente sintaxis en Java:

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

Parse JSON cadena en JavaScript

En JavaScript, el objeto `JSON` se utiliza para analizar una cadena JSON. Este método solo está disponible en los navegadores modernos (IE8 +, Firefox 3.5+, etc.).

Cuando se analiza una cadena JSON válida, el resultado es un objeto JavaScript, una matriz u otro valor.

```
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)
```

Las cadenas no válidas lanzarán un error de 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
```

El método `JSON.parse` incluye una función reviver opcional que puede limitar o modificar el resultado del análisis.

```
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}
```

En el último ejemplo, `JSON.parse` devuelve un valor `undefined` . Para evitar eso, devuelva el `value` dentro de la función de revivimiento.

Parse JSON archivo con Groovy

Supongamos que tenemos los siguientes datos JSON:

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

importar groovy.json.JsonSlurper

clase 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");
}
```

A continuación se muestra el analizador:

```
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
}
```

}

Lea Analizando la cadena JSON en línea: https://riptutorial.com/es/json/topic/3878/analizando-la-cadena-json

Capítulo 3: Stringify - Convertir JSON a cadena

Parámetros

Param	Detalles
Objeto	(Objeto) El objeto JSON
Sustituto	(Función | Array <string | número> - optiotinal) filtro Función | Formación
Espacio	(Número | cadena - opcional) Cantidad de espacio en blanco en el JSON

Examples

Convertir simple objeto JSON a cadena simple

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

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

Estriar con filtro

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

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

Stringify con espacio en blanco

```
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
 *   }
 */
```

Lea Stringify - Convertir JSON a cadena en línea:
https://riptutorial.com/es/json/topic/7824/stringify---convertir-json-a-cadena

Creditos

S. No	Capítulos	Contributors
1	Empezando con JSON	AndriuZ, Asaph, ccprog, Community, depperm, francesco foresti , gandreadis, Gavishiddappa Gadagi, itzmukeshy7, laktak, Mirec Miskuf, Nilanchala Panigrahy, pickypg, reidzeibel, user2314737, Vitor Baptista
2	Analizando la cadena JSON	Kruti Patel, Mahbub Rahman, Mottie, Vitor Baptista
3	Stringify - Convertir JSON a cadena	Mosh Feu