

Introducing JSON

العربية Български 中文 Český Dansk Nederlands English Esperanto Français Deutsch Ελληνικά עברית Magyar Indonesia Italiano 日本 한국어 فارسی Polski Português Română Русский Српско-хрватски Slovenščina Español Svenska Türkçe Tiếng Việt

ECMA-404 The JSON Data Interchange Standard.

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the [JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999](#). JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

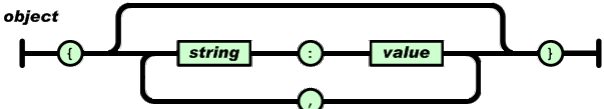
JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

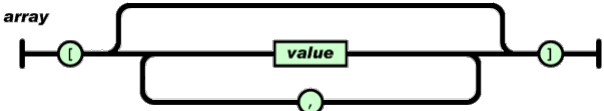
These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

In JSON, they take on these forms:

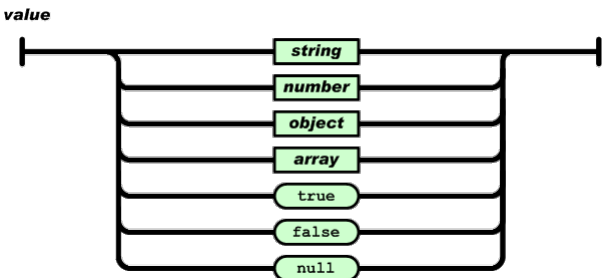
An *object* is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma).



An *array* is an ordered collection of values. An array begins with [(left bracket) and ends with] (right bracket). Values are separated by , (comma).



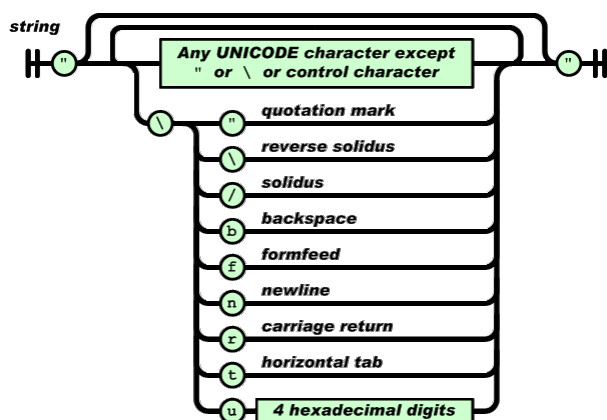
A *value* can be a *string* in double quotes, or a *number*, or *true* or *false* or *null*, or an *object* or an *array*. These structures can be nested.



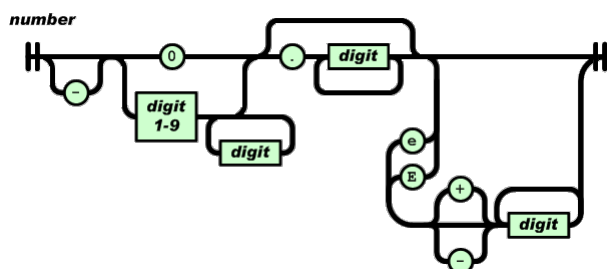
A *string* is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. A string is very much like a C or Java string.

```
object
{
  { members }
members
pair
pair , members
pair
string : value
array
[
  { elements }
elements
value
value , elements
value
string
number
object
array
true
false
null

string
""
" chars "
chars
char
char chars
char
any-Unicode-character-
except-"-or-\-or-
control-character
\"
\\
\ /
\b
\f
\n
\r
\t
\u four-hex-digits
number
int
int frac
int exp
int frac exp
int
digit
digit1-9 digits
- digit
- digit1-9 digits
frac
. digits
exp
e digits
digits
digit
digit digits
e
e
e+
e-
E
E+
E-
E-
```



A number is very much like a C or Java number, except that the octal and hexadecimal formats are not used.



Whitespace can be inserted between any pair of tokens. Excepting a few encoding details, that completely describes the language.

- ABAP:
 - [EPO Connector](#).
- ActionScript:
 - [ActionScript3](#).
- Ada:
 - [GNATCOLL.JSON](#).
- AdvPL:
 - [JSON-ADVPL](#).
- ASP:
 - [JSON for ASP](#).
 - [JSON ASP utility class](#).
- AWK:
 - [JSON.awk](#).
 - [rhawk](#).
- Bash:
 - [Jshon](#).
 - [JSON.sh](#).
- BlitzMax:
 - [bmx-rjson](#).
- C:
 - [JSON_checker](#).
 - [YAJL](#).
 - [LibU](#).
 - [json-c](#).
 - [json-parser](#).
 - [jsonsl](#).
 - [WJElement](#).
 - [M's JSON parser](#).
 - [cJSON](#).
 - [jansson](#).
 - [jsmn](#).
 - [parson](#).
 - [ujson4c](#).
 - [nxjson](#).
 - [frozen](#).
 - [microjson](#).
- C++:
 - [JSONKit](#).
 - [jsonme--](#).
 - [ThorsSerializer](#).
 - [JsonBox](#).
 - [jvar](#).
 - [rapidjson](#).
 - [JSON for Modern C++](#).
 - [qmjson](#).
 - [minijson](#).
 - [jsoncons](#).
 - [JSON Support in Qt](#).
 - [Qjson](#).
 - [jsoncpp](#).
 - [JOST](#).
- ColdFusion:
 - [SerializeJSON](#).
 - [toJSON](#).
- D:
 - [Libdjson](#).
- Dart:
 - [json library](#).
- Delphi:
 - [Delphi Web Utils](#).
 - [JSON Delphi Library](#).
- E:
 - [JSON in TermL](#).
- Fantom:
 - [Json](#).
- FileMaker:
 - [JSON](#).
- Fortran:
 - [json-fortran](#).
 - [YAJL-Fort](#).
- Go:
 - [package json](#).
- Groovy:
 - [groovy-io](#).
- Haskell:
 - [Rjson package](#).
 - [json package](#).
- Java:
 - [JSON-java](#).
 - [JSONUtil](#).
 - [jsonp](#).
 - [Json-lib](#).
 - [Stringtree](#).
 - [SOJO](#).
 - [json-taglib](#).
 - [Flexjson](#).
 - [JON tools](#).
 - [Argo](#).
 - [jsonij](#).
 - [fastjson](#).
 - [mjson](#).
 - [ijjson](#).
 - [json-simple](#).
 - [json-io](#).
 - [JsonMarshaller](#).
 - [google-gson](#).
 - [Json-smart](#).
 - [FOSS Nova JSON](#).
 - [Corn CONVERTER](#).
 - [Apache johnzon](#).
 - [Genson](#).
 - [JSONUtil](#).
- Net.Data:
 - [netdata-json](#).
- Objective C:
 - [NSJSONSerialization](#).
 - [json-framework](#).
 - [JSONKit](#).
 - [yajl-objc](#).
 - [TouchJSON](#).
- OCaml:
 - [Yojson](#).
 - [jsonm](#).
- PascalScript:
 - [JsonParser](#).
- Perl:
 - [CPAN](#).
 - [perl-JSON-SL](#).
- Photoshop:
 - [JSON Photoshop Scripting](#).
- PHP:
 - [PHP 5.2](#).
- PicoLisp:
 - [picolisp-json](#).
- Pike:
 - [Public.Parser.JSON](#).
 - [Public.Parser.JSON2](#).
- PL/SQL:
 - [pljson](#).
 - [PowerShell](#).
 - [PowerShell](#).
 - [Puredata](#).
 - [PuRestJson](#).
- Python:
 - [The Python Standard Library](#).
 - [simplejson](#).
 - [pyson](#).
 - [Yajl-Py](#).
 - [ultrajson](#).
 - [metamagic.json](#).
- R:
 - [rjson](#).
 - [jsonlite](#).
- Racket:
 - [json-parsing](#).
- Rebol:
 - [json.r](#).
- RPG:
 - [JSON Utilities](#).
- Rust:
 - [Serde JSON](#).
 - [json-rust](#).
- Ruby:
 - [rjson](#).

- CAJUN.
- libjson.
- nosjob.
- JSON++.
- JSON library for IoT.
- C#:
 - fastJSON.
 - JSON_checker.
 - Jayrock.
 - Json.NET - LINQ to JSON.
 - LitJSON.
 - JSON for .NET.
 - JSON@CodeTitans.
 - JSONSharp.
 - fluent-json.
 - Manatee Json.
 - FastjsonParser.
- Ciao:
 - Ciao JSON encoder and decoder.
- Clojure:
 - data.json.
- Cobol:
 - XML Thunder.
 - Redvers COBOL JSON Interface.
- cookjson.
- JavaScript:
 - JSON.
 - json2.js.
 - clarinet.
 - Oboe.js.
- LabVIEW:
 - flatten.
- Lisp:
 - Common Lisp JSON.
 - Emacs Lisp.
- LiveCode:
 - mergJSON.
- LotusScript:
 - JSON LS.
- LPC:
 - Grimoire: LPC JSON.
- Lua:
 - JSON Modules.
- M:
 - DataBallet.
- Matlab:
 - JSONlab.
 - 20565.
 - 23393.
- json.
- yajl-ruby.
- json-stream.
- yajl-ffi.
- Scheme:
 - MZScheme.
 - PLT Scheme.
- Squeak:
 - Squeak.
- Symbian:
 - s60-json-library.
- Tcl:
 - JSON.
- Visual Basic:
 - VB-JSON.
 - PWJSON.
 - .NET-JSON-Transformer.
- Visual FoxPro:
 - fwJSON.
 - JSON.
 - vfpjson.