

# The `jsonparse` package

An easy way to parse, store and access JSON data from files or strings in LaTeX documents

Jasper Habicht \*

Version 0.5.0, released on 8 April 2024

---

## 1 Introduction

The `jsonparse` package provides an easy way to read in JSON data from files or strings in LaTeX documents, parse the data and store it in a user-defined token variable. The package allows accessing the stored data via a JavaScript-flavored syntax.

## 2 Loading the package

To install the package, copy the package file `jsonparse.sty` into the working directory or into the `texmf` directory. After the package has been installed, the `jsonparse` package is loaded by calling `\usepackage{jsonparse}` in the preamble of the document.

The package does not load any dependencies.

debug

The package can be loaded with the option `debug`. It will then output to the log file every instance of a string, a boolean (true or false) value, a null value, a number as well as the start and end of every object and the start and end of every array that is found while parsing the JSON string or JSON file.

## 3 Main user commands

```
\JSONParse{<token variable>}{<JSON string>}
```

The command `\JSONParse` is used to parse a JSON string and store the parsed result in a token variable (a property list). The first argument takes the name of the token variable that is created by the command. The second argument takes the JSON string to be parsed.

For example, using `\JSONParse{\myJSONdata}{ { "key" : "value" } }`, the relevant JSON string will be parsed and the result stored in the token variable `\myJSONdata` as property list. In this case, the property list only consists of one entry with the key `key` and the value `value`. The command `\JSONParseGetValue{\myJSONdata}{key}`, for example, can then be used to extract the relevant value from this property list (see the description below).

```
\JSONParseFromFile{<token variable>}{<JSON file>}
```

---

\* E-mail: [mail@jasperhabicht.de](mailto:mail@jasperhabicht.de)

The command `\JSONParseFromFile` is used to parse a JSON file and store the parsed result in a token variable (a property list). It works the same way as `\JSONParse`, but instead of a JSON string, it takes as second argument the path to the JSON file relative to the working directory.

**`\JSONParseGetValue{<token variable>}{<key>}`**

The command `\JSONParseGetValue` is used to select values from the token variable (property list) that has been created using the commands `\JSONParse` or `\JSONParseFromFile`. The first argument takes the token variable that holds the parsed JSON data. The second argument takes the key to select the relevant entry from the parsed JSON data using JavaScript syntax.

If the JSON string `{ "key" : "value" }` is parsed into the token variable `\myJSONdata`, using `\JSONParseGetValue{\myJSONdata}{key}` would extract the value associated with the key `key`, which in this case is `value`, and typeset it to the document.

Nested objects and arrays are assigned keys that adhere to JavaScript syntax. For example, if the JSON string `{ "outer_key" : { "inner_key" : "value" } }` is parsed into the token variable `\myJSONdata`, to select the value associated with the key `inner_key`, the command `\JSONParseGetValue{\myJSONdata}{outer_key.inner_key}` can be used. The command `\JSONParseGetValue{\myJSONdata}{key[0]}` selects the first value of the array associated with the key `key` in the parsed JSON string `{ "key" : [ "one" , "two" ] }`.

**`\JSONParseGetArrayValues{<token variable>}{<key>}[<subkey>]{<string>}`**

The command `\JSONParseGetArrayValues` is used to select all values from an array from a parsed JSON string or JSON file. The first argument takes the token variable that holds the parsed JSON data. The second argument takes the key to select the relevant entry from the parsed JSON data using JavaScript syntax. The third argument is optional and can be used to assign a subkey to every value. The last argument takes a string that is inserted between all values when they are typeset.

For example, let us assume the following JSON data structure is parsed into the token variable `\myJSONdata`:

```
{
  "array" : [
    {
      "key_a" : "one" ,
      "key_b" : "two"
    } ,
    {
      "key_a" : "three" ,
      "key_b" : "four"
    }
  ]
}
```

Then using `\JSONParseGetArrayValues{\myJSONdata}{array}[key_a]{, }`, the text `one, three` is typeset to the document.

**`\JSONParseUseArrayValues{<token variable>}{<key>}[<subkey>]{<command name>}`**

The command `\JSONParseUseArrayValues` takes the same first three arguments as the command `\JSONParseGetArrayValues` and works in a similar way. However, instead of a string that is added between the array items, it takes a command name as fourth argument. This command can be defined beforehand and should accept one argument that represents the value of the array item. It will be called for every array item.

For example, let us assume the same JSON data structure as defined above parsed into the token variable `\myJSONdata`. Then, the following can be done:

```

\newcommand{\myJSONitem}[1]{
  \item \emph{#1}
}

• one

• three

\begin{itemize}
  \JSONParseUseArrayValues{\myJSONdata}
    {array}[key_a]{\myJSONitem}
\end{itemize}

```

### 3.1 Commands to alter separators and output

The package provides a set of commands that can be used to change the separators used to select the relevant value in the JSON structure as well as the output that is generated from the JSON data.

```

\JSONParseSetChildSeparator{<string>}
\JSONParseSetArraySeparator{<string>}{<string>}

```

The two commands `\JSONParseSetChildSeparator` and `\JSONParseSetArraySeparator` can be used to change the separators for child objects or array items in the syntax used to select a specific value in the JSON data structure. Per default, the child separator is a dot ( `.` ) while the array separators are square brackets ( `[` and `]` ). The command `\JSONParseSetArraySeparator` takes two arguments of which the first is the character used before the array separator and the second is used after. Changing the separators can be useful if keys in the JSON structure already use these characters.

```

\JSONParseSetTrueString{<string>}
\JSONParseSetFalseString{<string>}
\JSONParseSetNullString{<string>}

```

The commands `\JSONParseSetTrueString` and `\JSONParseSetFalseString` as well as the command `\JSONParseSetNullString` can be used to change the output that is typeset if the JSON value is `true`, `false` or `null`. The default strings that are typeset are `true`, `false` or `null` respectively.

```

\JSONParseSetArrayIndexZeroBased
\JSONParseSetArrayIndexOneBased

```

The command `\JSONParseSetArrayIndexZeroBased` sets the numbering of the index of array items to zero-based. With the command `\JSONParseSetArrayIndexOneBased`, the indexing starts with one instead. Per default, the package uses zero-based indexing to resemble JavaScript notation.

### 3.2 Internal variables and commands

```

\l_jsonparse_debug_mode_bool

```

The token variable `\l_jsonparse_debug_mode_bool` holds the boolean value true if the package is loaded with the `debug` option and false otherwise. If set to true, the debug mode will be used for the following use of `\jsonparse_parse:n`.

```
\l_jsonparse_child_sep_str
\l_jsonparse_array_sep_left_str
\l_jsonparse_array_sep_right_str
\l_jsonparse_true_str
\l_jsonparse_false_str
\l_jsonparse_null_str
\l_jsonparse_array_index_zero_based_bool
```

The token variable `\l_jsonparse_child_sep_str` holds the string `.` per default or the string set by the user command `\JSONParseSetChildSeparator`.

The token variable `\l_jsonparse_array_sep_left_str` holds the string `[` per default or the string given as first argument to the user command `\JSONParseSetArraySeparator`.

The token variable `\l_jsonparse_array_sep_right_str` holds the string `]` per default or the string given as second argument to the user command `\JSONParseSetArraySeparator`.

The token variable `\l_jsonparse_true_str` holds the string `true` per default or the string set by the user command `\JSONParseSetTrueString`.

The token variable `\l_jsonparse_false_str` holds the string `false` per default or the string set by the user command `\JSONParseSetFalseString`.

The token variable `\l_jsonparse_null_str` holds the string `null` per default or the string set by the user command `\JSONParseSetNullString`.

The token variable `\l_jsonparse_array_index_zero_based_bool` holds a boolean value which is `true` per default or if the user command `\JSONParseSetArrayIndexZeroBased` was called. The boolean value is `false` if the user command `\JSONParseSetArrayIndexOneBased` was called.

## `\g_jsonparse_entries_prop`

The token variable (property list) `\g_jsonparse_entries_prop` holds key-value pairs representing all elements of a JSON data structure generated by a previous run of `\jsonparse_parse:n`.

## `\jsonparse_parse:n {<JSON string>}`

The command `\jsonparse_parse:n` takes as argument a JSON string and populates the token variable (property list) `\g_jsonparse_entries_prop` with key-value pairs representing all elements of the JSON data structure represented by this string.

## `\jsonparse_parse_to_prop:Nn <token variable> {<JSON string>}`

The command `\jsonparse_parse_to_prop:Nn` creates the token variable given as the first arguments as property list and, after having called `\jsonparse_parse:n` using the second argument, sets this newly created property list equal to `\g_jsonparse_entries_prop`.

## `\jsonparse_remove_leading_spaces:N <token variable>`

The command `\jsonparse_remove_leading_spaces:N` trims leading spaces from the string stored in the token variable that is given to this command as argument.

## 4 Changes

**v0.3.0** (2024/04/08)

First public beta release.

**v0.5.0** (2024/04/09)

Changed from string token variables to token lists to support Unicode.