Table of Contents

Methods

module JSON

JavaScript Object Notation (JSON)

JSON is a lightweight data-interchange format. It is easy for us humans to read and write. Plus, equally simple for machines to generate or parse. JSON is completely language agnostic, making it the ideal interchange format.

Built on two universally available structures:

```
1. A collection of name/value pairs. Often referr
2. An ordered list of values. More commonly calle
```

To read more about JSON visit: json.org

Parsing JSON

To parse a JSON string received by another application or generated within your existing application:

```
require 'json'

my_hash = JSON.parse('{"hello": "goodbye"}')
puts my_hash["hello"] => "goodbye"
```

Notice the extra quotes `''` around the hash notation. Ruby expects the argument to be a string and can't convert objects like a hash or array.

Ruby converts your string into a hash

Generating JSON

Creating a JSON string for communication or serialization is just as simple.

```
require 'json'

my_hash = {:hello => "goodbye"}
puts JSON.generate(my_hash) => "{\"hello\":\"good
```

Or an alternative way:

```
require 'json'
puts {:hello => "goodbye"}.to_json => "{\"hello\"
```

`JSON.generate` only allows objects or arrays to be converted to JSON syntax. `to_json`, however, accepts many Ruby classes even though it acts only as a method for serialization:

```
require 'json'

1.to_json => "1"
```

Constants

`Infinity`

`JSON_LOADED`

`MinusInfinity`

`NaN`

`UnparserError`

This exception is raised if a generator or unparser error occurs.

`VERSION`

JSON version

Attributes

`create_id [RW]`

This is create identifier, which is used to decide if the *json_create* hook of a class should be called. It defaults to 'json_class'.

`dump_default_options [RW]`

The global default options for the #dump method:

```
:max_nesting: false
:allow_nan: true
:allow_blank: true
```

`generator [R]`

Returns the JSON generator module that is used by JSON. This is either JSON::Ext::Generator or JSON::Pure::Generator.

`load_default_options [RW]`

The global default options for the #load method:

```
:max_nesting: false
:allow_nan: true
:allow_blank: true
```

`parser [R]`

Returns the JSON parser class that is used by JSON. This is either JSON::Ext::Parser or JSON::Pure::Parser.

`state [RW]`

Returns the JSON generator state class that is used by JSON. This is either JSON::Ext::Generator::State or JSON::Pure::Generator::State.

Public Class Methods

`[](object, opts = {})`

If *object* is string-like, parse the string and return the parsed result as a Ruby data structure. Otherwise generate a JSON text from the Ruby data structure object and return it.

The *opts* argument is passed through to generate/parse respectively. See generate and parse for their documentation.

`iconv(to, from, string)`

Encodes string using Ruby's *String.encode*

`restore(source, proc = nil, options = {})`

Alias for: load

Public Instance Methods

`dump(obj, anIO = nil, limit = nil)`

Dumps *obj* as a JSON string, i.e. calls generate on the object and returns the result.

If anIO (an IO-like object or an object that responds to the write method) was given, the resulting JSON is written to it.

If the number of nested arrays or objects exceeds *limit*, an ArgumentError exception is raised. This argument is similar (but not exactly the same!) to the *limit* argument in Marshal.dump.

The default options for the generator can be changed via the ::dump_default_options method.

This method is part of the implementation of the load/dump interface of Marshal and YAML.

`fast_generate(obj, opts = nil)`

Generate a JSON document from the Ruby data structure *obj* and return it. This method disables the checks for circles in Ruby objects.

WARNING: Be careful not to pass any Ruby data structures with circles as *obj* argument because this will cause JSON to go into an infinite loop.

`generate(obj, opts = nil)`

Generate a JSON document from the Ruby data structure *obj* and return it. *state* is * a JSON::State object,

- or a Hash like object (responding to to_hash),
- an object convertible into a hash by a to_h method,

that is used as or to configure a State object.

It defaults to a state object, that creates the shortest possible JSON text in one line, checks for circular data structures and doesn't allow NaN, Infinity, and -Infinity.

A *state* hash can have the following keys:

- **indent**: a string used to indent levels (default: ''),
- **space**: a string that is put after, a : or , delimiter (default: ''),
- **space_before**: a string that is put before a : pair delimiter (default: ''),
- **object_nl**: a string that is put at the end of a JSON object (default: ''),
- **array_nl**: a string that is put at the end of a JSON array (default: ''),
- **allow_nan**: true if NaN, Infinity, and -Infinity should be generated, otherwise an exception is thrown if these values are encountered. This options defaults to false.
- **max_nesting**: The maximum depth of nesting allowed in the data structures from which JSON is to be generated. Disable depth checking with :max_nesting => false, it defaults to 100.

See also the #fast_generate for the fastest creation method with the least amount of sanity checks, and the #pretty_generate method for some defaults for pretty output.

`load(source, proc = nil, options = {})`

Load a ruby data structure from a JSON *source* and return it. A source can either be a string-like object, an IO-like object, or an object responding to the read method. If *proc* was given, it will be called with any nested Ruby object as an argument recursively in depth first order. To modify the default options pass in the optional *options* argument as well.

BEWARE: This method is meant to serialise data from trusted user input, like from your own database server or clients under your control, it could be dangerous to allow untrusted users to pass JSON sources into it. The default options for the parser can be changed via the ::load_default_options method.

This method is part of the implementation of the load/dump interface of Marshal and YAML.

Also aliased as: restore

`parse(source, opts = {})`

Parse the JSON document *source* into a Ruby data structure and return it.

opts can have the following keys:

- max_nesting: The maximum depth of nesting allowed in the parsed data structures. Disable depth checking with :max_nesting => false. It defaults to 100.
- allow_nan: If set to true, allow NaN, Infinity and -Infinity in defiance of RFC 7159 to be parsed by the Parser. This option defaults to false.
- symbolize_names: If set to true, returns symbols for the names (keys) in a JSON object. Otherwise strings are returned. Strings are the default.
- create_additions: If set to false, the Parser doesn't create additions even if a matching class and ::create_id was found. This option defaults to false.
- object_class: Defaults to Hash
- array_class: Defaults to Array

`parse!(source, opts = {})`

Parse the JSON document *source* into a Ruby data structure and return it. The bang version of the parse method defaults to the more dangerous values for the *opts* hash, so be sure only to parse trusted *source* documents.

opts can have the following keys:

- max_nesting: The maximum depth of nesting allowed in the parsed data structures. Enable depth checking with :max_nesting => anInteger. The parse! methods defaults to not doing max depth checking: This can be dangerous if someone wants to fill up your stack.
- allow_nan: If set to true, allow NaN, Infinity, and -Infinity in defiance of RFC 7159 to be parsed by the Parser. This option defaults to true.

- create_additions: If set to false, the Parser doesn't create additions even if a matching class and ::create_id was found. This option defaults to false.

`pretty_generate(obj, opts = nil)`

Generate a JSON document from the Ruby data structure *obj* and return it. The returned document is a prettier form of the document returned by unparse.

The *opts* argument can be used to configure the generator. See the generate method for a more detailed explanation.

`recurse_proc(result, &proc)`

Recursively calls passed *Proc* if the parsed data structure is an *Array* or *Hash*

Private Instance Methods

`restore(source, proc = nil, options = {})`

Alias for: load