Package 'jsonvalidate'

June 25, 2019

Title Validate 'JSON'

Version 1.1.0

Maintainer Rich FitzJohn <rich.fitzjohn@gmail.com>

Description Uses the node library 'is-my-json-valid' or 'ajv' to validate 'JSON' against a 'JSON' schema. Drafts 04, 06 and 07 of 'JSON' schema are supported.

License MIT + file LICENSE

LazyData true

URL https://github.com/ropensci/jsonvalidate

BugReports https://github.com/ropensci/jsonvalidate/issues

Imports V8

Suggests knitr, rmarkdown, testthat

RoxygenNote 6.1.1

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Rich FitzJohn [aut, cre],
Ian Lyttle [ctb],
Kara Woo [ctb],
Mathias Buus [cph],
Evgeny Poberezkin [cph]

Repository CRAN

Date/Publication 2019-06-25 09:00:03 UTC

R topics documented:

`json_validate` *Validate a json file*

Description

Validate a single json against a schema. This is a convenience wrapper around `json_validator(schema)(json)`

Usage

```
json_validate(json, schema, verbose = FALSE, greedy = FALSE,
  error = FALSE, engine = "imjv")
```

Arguments

`json`	Contents of a json object, or a filename containing one.
`schema`	Contents of the json schema, or a filename containing a schema.
`verbose`	Be verbose? If `TRUE`, then an attribute "errors" will list validation failures as a data.frame
`greedy`	Continue after the first error?
`error`	Throw an error on parse failure? If `TRUE`, then the function returns `NULL` on success (i.e., call only for the side-effect of an error on failure, like `stopifnot`).
`engine`	Specify the validation engine to use. Options are "imjv" (the default; which uses "is-my-json-valid") and "ajv" (Another JSON Schema Validator). The latter supports more recent json schema features.

Examples

```
# A simple schema example:
schema <- '{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Product",
    "description": "A product from Acme\'s catalog",
    "type": "object",
    "properties": {
        "id": {
            "description": "The unique identifier for a product",
            "type": "integer"
        },
        "name": {
            "description": "Name of the product",
            "type": "string"
        },
        "price": {
            "type": "number",
            "minimum": 0,
            "exclusiveMinimum": true
        },
```

```
        "tags": {
            "type": "array",
            "items": {
                "type": "string"
            },
            "minItems": 1,
            "uniqueItems": true
        }
    },
    "required": ["id", "name", "price"]
}'

# Test if some (invalid) json conforms to the schema
jsonvalidate::json_validate("{}", schema, verbose = TRUE)

# Test if some (valid) json conforms to the schema
jsonvalidate::json_validate('{
    "id": 1,
    "name": "A green door",
    "price": 12.50,
    "tags": ["home", "green"]
}', schema)
```

`json_validator` *Create a json validator*

Description

Create a validator that can validate multiple json files.

Usage

```
json_validator(schema, engine = "imjv")
```

Arguments

`schema`	Contents of the json schema, or a filename containing a schema.
`engine`	Specify the validation engine to use. Options are "imjv" (the default; which uses "is-my-json-valid") and "ajv" (Another JSON Schema Validator). The latter supports more recent json schema features.

Examples

```
# A simple schema example:
schema <- '{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Product",
    "description": "A product from Acme\'s catalog",
    "type": "object",
```

```
    "properties": {
        "id": {
            "description": "The unique identifier for a product",
            "type": "integer"
        },
        "name": {
            "description": "Name of the product",
            "type": "string"
        },
        "price": {
            "type": "number",
            "minimum": 0,
            "exclusiveMinimum": true
        },
        "tags": {
            "type": "array",
            "items": {
                "type": "string"
            },
            "minItems": 1,
            "uniqueItems": true
        }
    },
    "required": ["id", "name", "price"]
}'

# Create a validator function
v <- jsonvalidate::json_validator(schema)

# Test if some (invalid) json conforms to the schema
v("{}", verbose = TRUE)

# Test if some (valid) json conforms to the schema
v('{
    "id": 1,
    "name": "A green door",
    "price": 12.50,
    "tags": ["home", "green"]
}')

# Using features from draft-06 or draft-07 requires the ajv engine:
schema <- "{
  '$schema': 'http://json-schema.org/draft-06/schema#',
  'type': 'object',
  'properties': {
    'a': {
      'const': 'foo'
    }
  }
}"

# Create the validator
v <- jsonvalidate::json_validator(schema, engine = "ajv")
```

```
# This confirms to the schema
v('{"a": "foo"}')

# But this does not
v('{"a": "bar"}')
```

Index