Introduction to JSON Schema

Julian Berman
https://github.com/Julian | @JulianWasTaken | tos9 on Freenode

MAGNETIC™

MAGNETIC™

Julian@Magnetic.com

```
{
    "name": "PyCon Sette",
    "location": [
        43.7661479,
        11.2699767
    ]
}
```

`jsonschema`

https://github.com/Julian/jsonschema

42

instance

{“**type**”: “*integer*”}

schema

MAGNETIC™

```
>>> from jsonschema import validate
>>> validate(instance=42, schema={"type": "integer"})
```

“Fourty Two”

{“**type**”: “*integer*”}

MAGNETIC™

```
>>> validate(instance="Fourty Two", schema={"type": "integer"})
Traceback (most recent call last):
   ...
ValidationError: 'Fourty Two' is not of type 'integer'

Failed validating 'type' in schema:
    {'type': 'integer'}

On instance:
    'Fourty Two'
```

What Can We Validate?

- Types
- Ranges, Sizes & Lengths
- Container Contents
- Form
- Higher-Order Operators

```
{
    "name": "PyCon Sette",
    "location": [
        43.7661479,
        11.2699767
    ]
}
```

```
{
  "type": "object",
  "properties": {
    "name": {"type": "string"},
    "location": {
      "type": "array",
      "description": "Latitude & Longitude",
      "items": [
        {"type": "integer", "minimum": -90, "maximum": 90},
        {"type": "integer", "minimum": -180, "maximum": 180}
      ]
    }
  }
}
```

*jsonschema, kind of a lie.

`*pythonschema`, less of a lie.

When Things Go Very Wrong

[1]

```
{
  "items": {
    "type": "boolean"
  },
  "minItems": 2
}
```

MAGNETIC™

```
>>> from jsonschema.validators import Draft4Validator
>>> validator = Draft4Validator(
...     schema={"items": {"type": "boolean"}, "minItems": 2},
... )
```

```
>>> validator.validate([True, False])
>>> validator.validate([True, True, True])
```

```
>>> validator.validate(instance=[1])
Traceback (most recent call last):
    ...
ValidationError: 1 is not of type 'boolean'

Failed validating 'type' in schema['items']:
    {'type': 'boolean'}

On instance[0]:
    1
```

```
>>> [error.message for error in validator.iter_errors(instance=[1])]
["1 is not of type 'boolean'", '[1] is too short']
```

What happened?

- **`message`**

Why did it happen?

- `context`
- `cause`

What was being *validated*?

- `instance`
- `path`
- `schema`
- `schema_path`
- `validator`
- `validator_value`

```
>>> jsonschema.ErrorTree(errors=validator.iter_errors(instance=[1]))
<ErrorTree (2 total errors)>

>>> tree = _
>>> tree[0].errors
{'type': <ValidationError: "1 is not of type 'boolean'">}
```

References

```
{
  "properties": {
    "name": {"type": "string", "minLength": 1},
    "nickname": {"type": "string", "minLength": 1},
    "username": {"type": "string", "minLength": 1}
  }
}
```

```
{
  "properties": {
    "name": {"$ref": "#/definitions/nameLike"},
    "nicknname": {"$ref": "#/definitions/nameLike"},
    "username": {"$ref": "#/definitions/nameLike"}
  },
  "definitions": {
    "nameLike": {"type": "string", "minLength": 1}
  }
}
```

```
{
  "properties": {
    "name": {"$ref": "#/definitions/nameLike"},
    "nicknname": {"$ref": "#/definitions/nameLike"},
    "username": {"$ref": "#/definitions/nameLike"}
  },
  "definitions": {
    "nameLike": {
      "type": "string", "minLength": 1, "maxLength": 4096
    }
  }
}
```

"Non-empty Sequence"

```
{
  "type": ["string", "array"],
  "minItems": 1,
  "minLength": 1
}
```

"A moniker, a name or an alias"

"S"

{"name": "Clark Kent"}

{"alias": "Superman"}

```
{
  "oneOf": [
    {"$ref": "#/definitions/moniker"},
    {"$ref": "#/definitions/name"},
    {"$ref": "#/definitions/alias"},
  ],

  ...
```

```
  "definitions": {
    "moniker": {"type": "string", "pattern": "[A-Z]+"},
    "name": {
      "type": "object",
      "required": ["name"],
      "properties": {"name": {"type": "string"}}
    },
    "alias": {
      "type": "object",
      "required": ["alias"],
      "properties": {"alias": {"type": "string"}}
    }
  }
}
```

```
Traceback (most recent call last):
  ...
ValidationError: '' is not valid under any of the given schemas

Failed validating 'oneOf' in schema:
    {'definitions': {'alias': {'properties': {'alias': {'type': 'string'}},
                               'required': ['alias'],
                               'type': 'object'},
                     'moniker': {'pattern': '[A-Z]+', 'type': 'string'},
                     'name': {'properties': {'name': {'type': 'string'}},
                              'required': ['name'],
                              'type': 'object'}},
     'oneOf': [{'$ref': '#/definitions/moniker'},
               {'$ref': '#/definitions/name'},
               {'$ref': '#/definitions/alias'}]}

On instance:
    ''
```

```
>>> list(validator.iter_errors(""))
[<ValidationError: "'' is not valid under any of the given schemas">]
>>> next(validator.iter_errors("")).context
[<ValidationError: "'' does not match '[A-Z]+'">,
 <ValidationError: "'' is not of type 'object'">,
 <ValidationError: "'' is not of type 'object'">]
```

```
>>> import jsonschema.exceptions
>>> print jsonschema.exceptions.best_match(errors=validator.iter_errors(""))
'' does not match '[A-Z]+'

Failed validating 'pattern' in schema[0]:
    {'pattern': '[A-Z]+', 'type': 'string'}

On instance:
    ''
```

Format

MAGNETIC™

```
{"type": "string", "format": "date"}
```

```
>>> from jsonschema import FormatChecker
>>> validator = Draft4Validator(
...     schema=schema, format_checker=FormatChecker(),
... )
```

MAGNETIC™

```
>>> validator.validate("2014-01")
Traceback (most recent call last):
  ...
ValidationError: '2014-01' is not a 'date'

Failed validating 'format' in schema:
    {'format': 'date', 'type': 'string'}

On instance:
    '2014-01'
```

```
>>> checker = jsonschema.FormatChecker()
>>> @checker.checks(format="prime")
... def is_prime(value):
...     return miller_rabin(value, iterations=10)
```

Meta-schemas

MAGNETIC™

```
Draft4Validator.check_schema(
    {"type": {"whoops": "an object"}},
)
```

```
Traceback (most recent call last):
  ...
SchemaError: {'whoops': 'an object'} is not valid under any of the given schemas

Failed validating u'anyOf' in schema[u'properties'][u'type']:
    {u'anyOf': [{u'$ref': u'#/definitions/simpleTypes'},
                {u'items': {u'$ref': u'#/definitions/simpleTypes'},
                 u'minItems': 1,
                 u'type': u'array',
                 u'uniqueItems': True}]}

On instance[u'type']:
    {'whoops': 'an object'}
```

```
>>> validator = Draft4Validator(schema=Draft4Validator.META_SCHEMA)
>>> for error in validator.iter_errors(
...     instance={"type": {"whoops": "an object"}, "minimum": "wage"}
... ):
...     print repr(error)

<ValidationError: "'wage' is not of type u'number'">
<ValidationError: "{'whoops': 'an object'} is not valid under any of the
given schemas">
```

A New Frontier

MAGNETIC™

Check Out...

- `pip install jsonschema`
- https://spacetelescope.github.io/understanding-json-schema/
- https://python-jsonschema.readthedocs.org/
- https://json-schema.org/
- https://github.com/json-schema/JSON-Schema-Test-Suite
- http://jsonschema.net/

Finito

MAGNETIC™