Merging structured data with *jsonmerge*

Tomaž Šolc
tomaz.solc@tablix.org

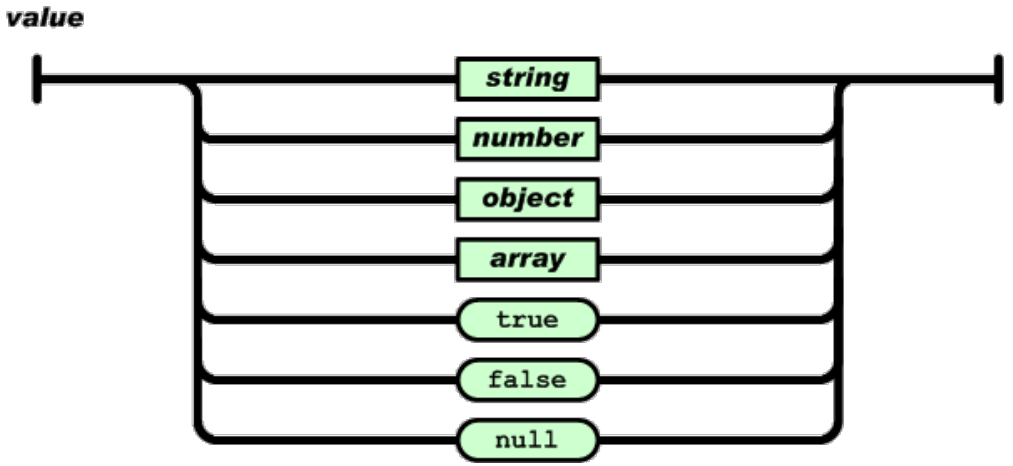
Python meetup, Ljubljana, 20 July 2017

About me

- I'm an electrical engineer (Univ. of Ljubljana)
  - Currently a research assistant at JSI
- I've been using Python since around 2007
  - These days mostly for numerical simulations and visualizations (numpy, matplotlib)
- In previous life I was doing semantic web, natural language processing, linked data.
- I sometimes help with opendata.si projects.

Structured what now?

- In short: data you can *serialize* to JSON format.




```
>>> data = {
    "title": "Alice's adventures in Wonderland",
    "publication": {
        "year": 1865,
    }
    "characters": ["Alice", "White rabbit"],
}
```

A more complicated structure


```
{
    "ocid": "ocds-213czf-000-00001",
    "id": "ocds-213czf-000-00001-05-contract",
    "date": "2010-05-10T10:30:00Z",
    "language": "en",
    "tag": [
        "contract"
    ],
    "initiationType": "tender",
    "parties": [
       {
            "identifier": {...},
            "name": "London Borough of Barnet",
            "address": {...},
            "contactPoint": {...},
            "roles": [ ... ],
            "id": "GB-LAC-E09000003"
        },
       {
            "identifier": {...},
            "additionalIdentifiers": [ ... ],
            "name": "AnyCorp Cycle Provision",
            "address": {...},
            "contactPoint": {...},
            "roles": [ ... ],
            "id": "GB-COH-1234567844"
        }
    ],
    "buyer": {...},
    "awards": [
       {
            "id": "ocds-213czf-000-00001-award-01",
            "title": "Award of contract to build new cycle lanes in the centre of town.",
            "description": "AnyCorp Ltd has been awarded the contract to build new ...",
            "status": "active",
            "date": "2010-05-10T10:30:00Z",
            "value": {
                "amount": 11000000,
                "currency": "GBP"
            },
            "suppliers": [ ... ],
            "items": [
               {
                    "id": "0001",
                    "description": "string",
                    "classification": {...},
                    "additionalClassifications": [ ... ],
                    "quantity": 8,
                    "unit": {...}
                }
            ],
            "contractPeriod": {...},
            "documents": [
               {
                    "id": "0007",
                    "documentType": "notice",
                    "title": "Award notice",
                    "description": "Award of contract ...",
                    "url": "http://example.com/tender-notices/ocds-213czf-000-00001-04.html",
                    "datePublished": "2010-05-10T10:30:00Z",
                    "format": "text/html",
                    "language": "en"
                }
            ]
        }
    ],
    "contracts": [
       {
            "id": "ocds-213czf-000-00001-contract-01",
            "awardID": "ocds-213czf-000-00001-award-01",
            "title": "Contract to build new cycle lanes in the centre of town.",
            "description": "A contract has been signed between the Council and AnyCorp Ltd ...",
            "status": "active",
            "period": {
                "startDate": "2010-07-01T00:00:00Z",
                "endDate": "2011-08-01T23:59:00Z"
            },
            "value": {
                "amount": 11000000,
                "currency": "GBP"
            },
            "items": [
               {
                    "id": "0001",
                    "description": "string",
                    "classification": {...},
                    "additionalClassifications": [ ... ],
                    "quantity": 8,
                    "unit": {...}
                }
            ],
            "dateSigned": "2015-06-10T14:23:12Z",
            "documents": [
               {
                    "id": "0008",
                    "documentType": "contractSigned",
                    "title": "Signed Contract",
                    "description": "The Signed Contract for Cycle Path Construction",
                    "url": "http://example.com/contracts/ocds-213czf-000-00001",
                    "datePublished": "2015-06-10T16:43:12Z",
                    "format": "application/pdf",
                    "language": "en"
                }
            ]
        }
    ]
}
```

jsonmerge

- *This Python module allows you to merge a series of JSON documents into a single one.*
- Supports Python 2.7 and 3.x
  - dependencies: jsonschema

pip install jsonmerge

- https://github.com/avian2/jsonmerge
  - 1.4.0 released on 5 June 2017
  - MIT license

Why is this useful?

- Gathering data about an object from different sources
  - For example, data from web scraping
  - Structured open data from different organizations (e.g. https://github.com/open-contracting)
- Data that gets updated over time
  - Merging changes from different authors
  - Recording how a document changed over time
  - jsonmerge borrows some terminology from git

Merging dicts with stdlib

```
>>> base = {
    "publication": { "year": 1865 },
    "title": "Alice's adventures in Wonderland",
    "characters": [ "Alice" ]
}
>>> head = {
    "publication": { "publisher": "Macmillan" },
    "characters": [ "White rabbit" ],
    "author": "Lewis Carroll"
}
>>> base.update(head)
>>> base
{'author': 'Lewis Carroll', ✔
 'characters': ['White rabbit'], ✘
 'publication': {'publisher': 'Macmillan'}, ✘
 'title': "Alice's adventures in Wonderland"} ✔
}
```

Merging with jsonmerge

```
>>> from jsonmerge import merge
>>> base = merge(base, head)
>>> base
{'author': 'Lewis Carroll',
 'characters': ['White rabbit'],
 'publication': {
  'publisher': 'Macmillan',
  'year': 1865
 },
 'title': "Alice's adventures in Wonderland"
}
```

Specifying a schema

```
>>> schema = {
   "properties" : {
      "characters": {
         "mergeStrategy": "append"
      }
   }
}
>>> base = merge(base, head, schema)
>>> base
{'author': 'Lewis Carroll',
 'characters': ['Alice', 'White rabbit'],
 'publication': {
  'publisher': 'Macmillan',
  'year': 1865
 },
 'title': "Alice's adventures in Wonderland"
}
```




But I thought JSON is schema-less?

- JSON can be very XML-ish if you want it to be
- JSON schema ≈ XML schema
  - http://json-schema.org
  - Python validator: https://github.com/Julian/jsonschema
  - jsonmerge extends JSON schema with the "mergeStrategy" keyword
- JSON pointer (RFC 6901) ≈ XPath
  - (including fun parts like external references...)
- JSON merge patch (RFC 7386) ≈ XML Patch
  - **not** related to jsonmerge and serves a different purpose

Example JSON schema

```
>>> schema = {
     "type": "object",
     "properties" : {
         "characters": {
             "type": "array",
             "elements": {
                 "type": "string"
             }
         },
         "title" : {
             "type": "string"
         },
         "publication": {
             "properties" : {
                 "year": {
                     "type": "number"
                 }
             }
         }
    }
}
```

Deeply nested structures

- *Without a schema,* jsonmerge will
  - recursively merge objects,
  - replace other types with newer values.

- *With a schema*, various merge strategies can be defined in (very) complex ways.
  - Each part of the document can be merged in a different way.
  - jsonmerge can automatically generate a validation schema for the resulting document.

Merge strategies

- Built-ins
  - overwrite
  - append
  - arrayMergeById (treat arrays as sets)
  - objectMerge (recursive "dict.update")
  - version (new values are appended to an array)

- Add your own by subclassing MergeStrategy

Example: versioning

```
>>> schema = { "type": "string",
               "mergeStrategy": "version" }
>>> from jsonmerge import Merger
>>> merger = Merger(schema)

>>> base = None
>>> base = merger.merge(base, "Hello, World!",
                        meta={"version": 1})
>>> base = merger.merge(base, "Howdy, World!",
                        meta={"version": 2})
>>> base
[
   {"value": "Hello, World!", "version": 1},
   {"value": "Howdy, World!", "version": 2}
]
```

Example: get_schema()

```
>>> base
[
   {"value": "Hello, World!", "version": 1},
   {"value": "Howdy, World!", "version": 2}
]
>>> merged_schema = merger.get_schema()
>>> merged_schema
{
 'type': 'array',
 'items': {
  'properties': { 'value': { 'type': 'string' } }
 }
}
>>> import jsonschema
>>> jsonschema.validate(base, merged_schema)
```

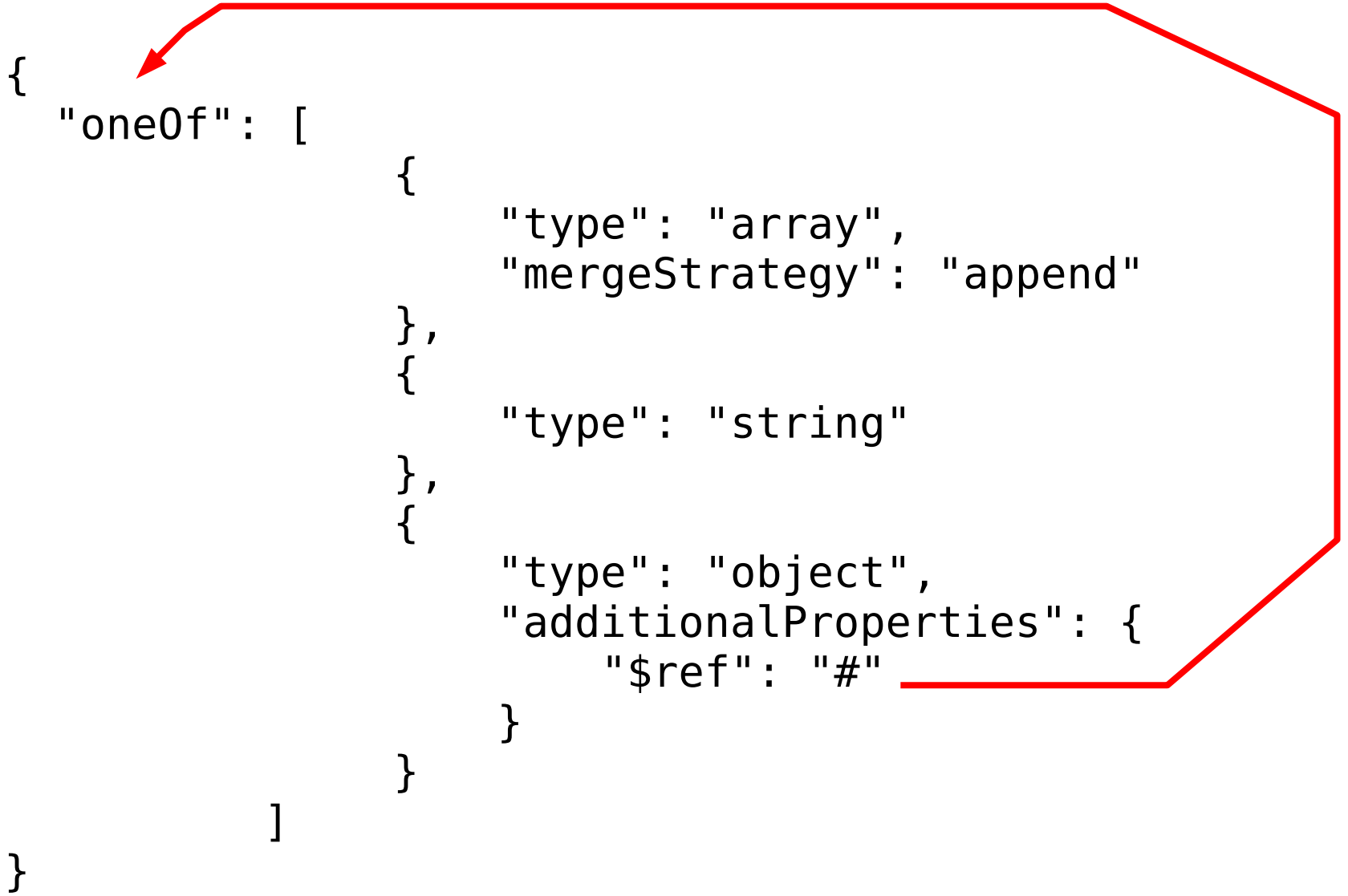
Example: array as a set

```
>>> schema = {
              "mergeStrategy": "arrayMergeById",
              "mergeOptions": {"idRef": "/"},
          }

>>> from jsonmerge import Merger
>>> merger = Merger(schema)

>>> base = [ 1, 2 ]
>>> head = [ 2, 3 ]
>>> merger.merge(base, head)
[ 1, 2, 3 ]
```

Example: all arrays with appends



```
{
  "oneOf": [
                {
                    "type": "array",
                    "mergeStrategy": "append"
                },
                {
                    "type": "string"
                },
                {
                    "type": "object",
                    "additionalProperties": {
                        "$ref": "#"
                    }
                }
            ]
}
```

Questions?

tomaz.solc@tablix.org

https://github.com/avian2/jsonmerge