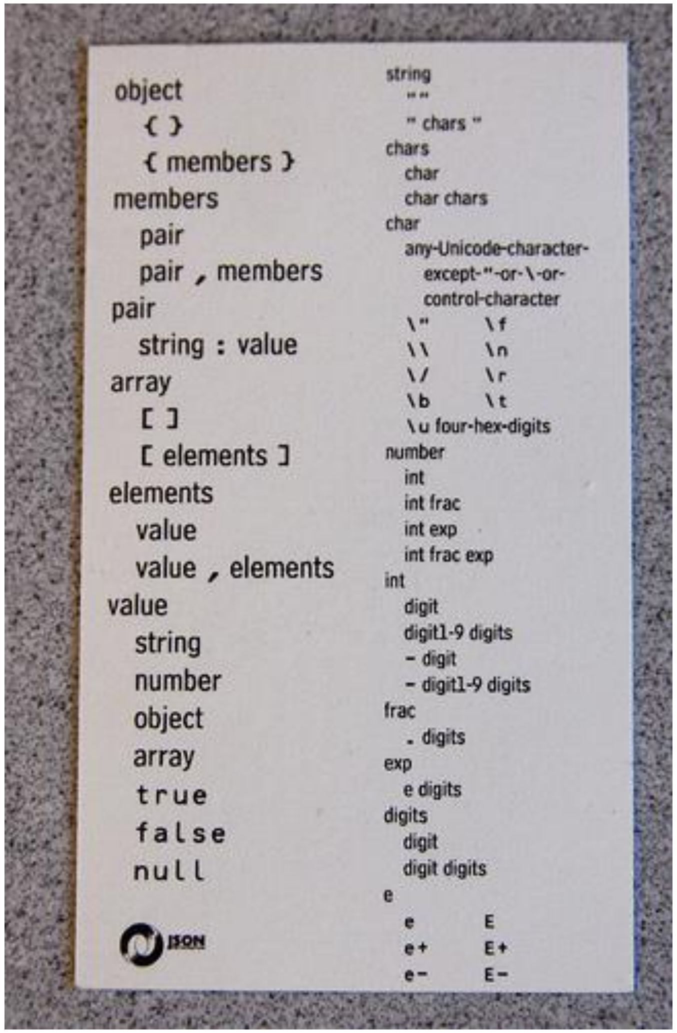
JSON Interoperability Vulns

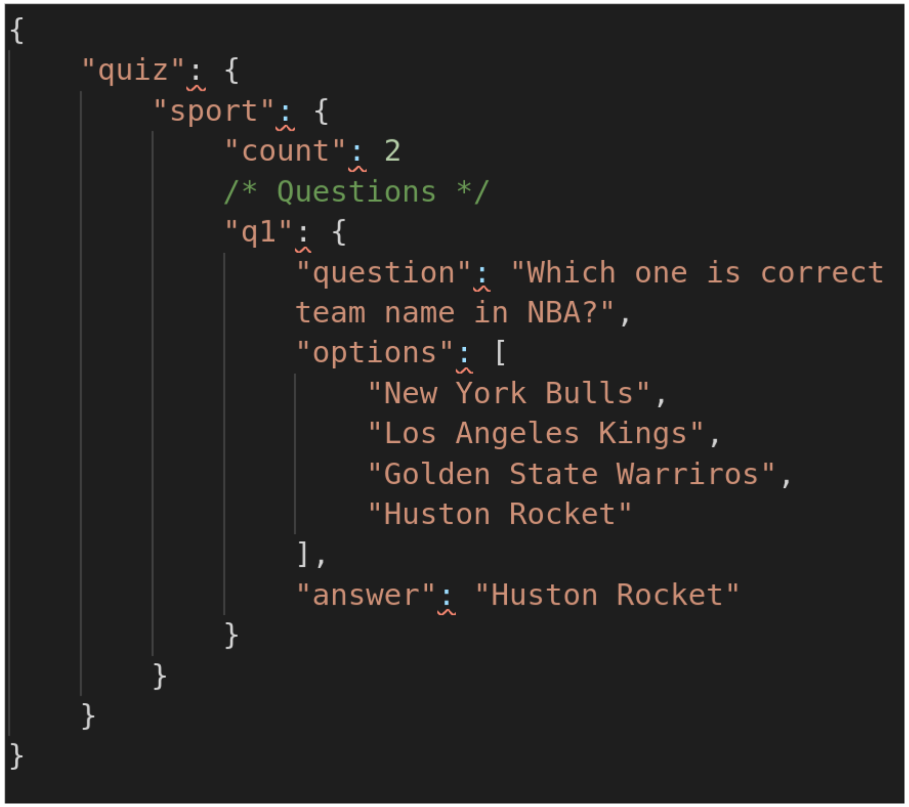
or: why JSON is (sometimes) bad and should feel bad.

What is JSON?



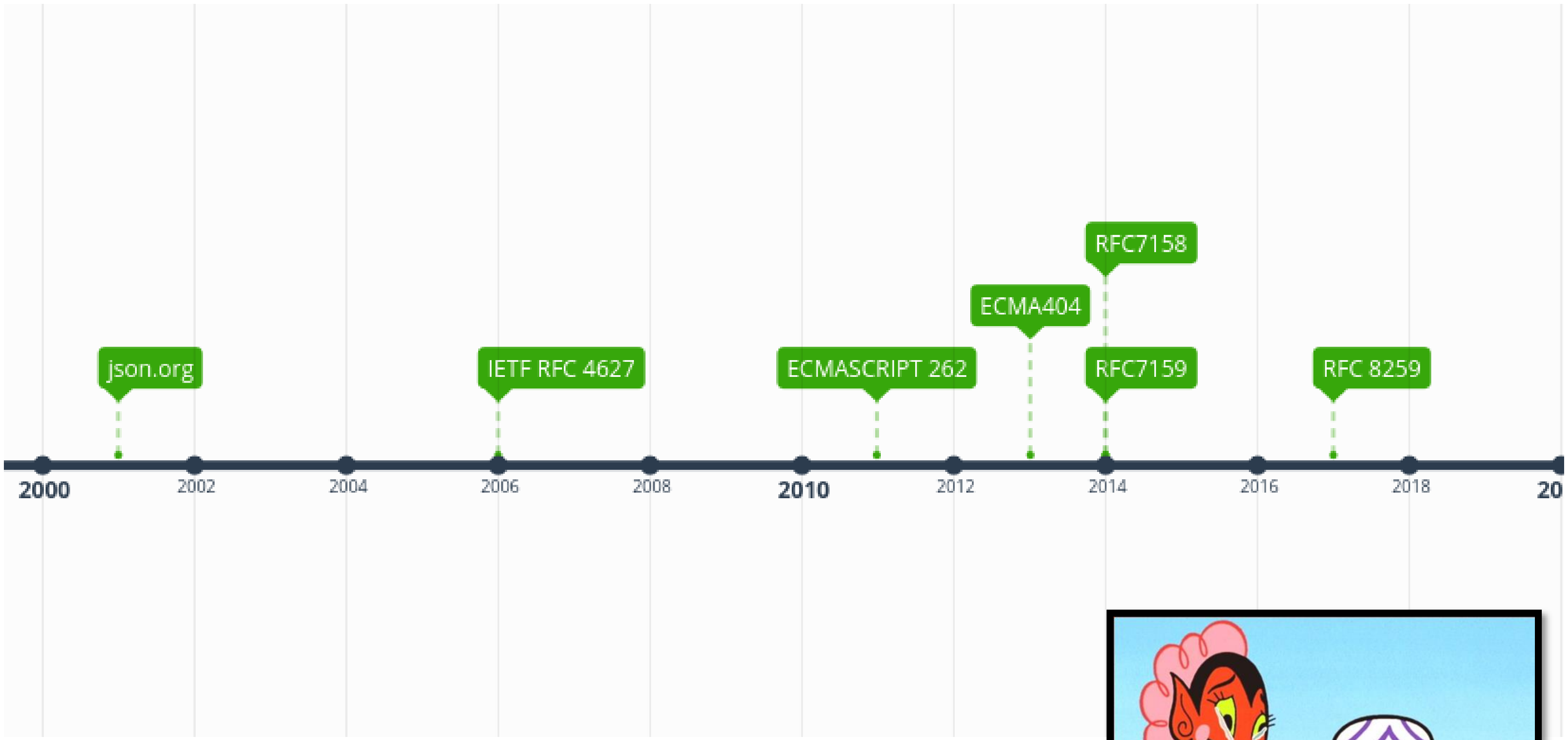
```
object
    { }
    { members }
members
    pair
    pair , members
pair
    string : value
array
    [ ]
    [ elements ]
elements
    value
    value , elements
value
    string
    number
    object
    array
    true
    false
    null
```

```
string
    ""
    " chars "
chars
    char
    char chars
char
    any-Unicode-character-
        except-"-or-\-or-
        control-character
    \"      \f
    \\      \n
    \/      \r
    \b      \t
    \u four-hex-digits
number
    int
    int frac
    int exp
    int frac exp
int
    digit
    digit1-9 digits
    - digit
    - digit1-9 digits
frac
    . digits
exp
    e digits
digits
    digit
    digit digits
e
    e       E
    e+      E+
    e-      E-
```



```
{
    "quiz": {
        "sport": {
            "count": 2
            /* Questions */
            "q1": {
                "question": "Which one is correct
                team name in NBA?",
                "options": [
                    "New York Bulls",
                    "Los Angeles Kings",
                    "Golden State Warriros",
                    "Huston Rocket"
                ],
                "answer": "Huston Rocket"
            }
        }
    }
}
```

SPECIFICATION(S?)

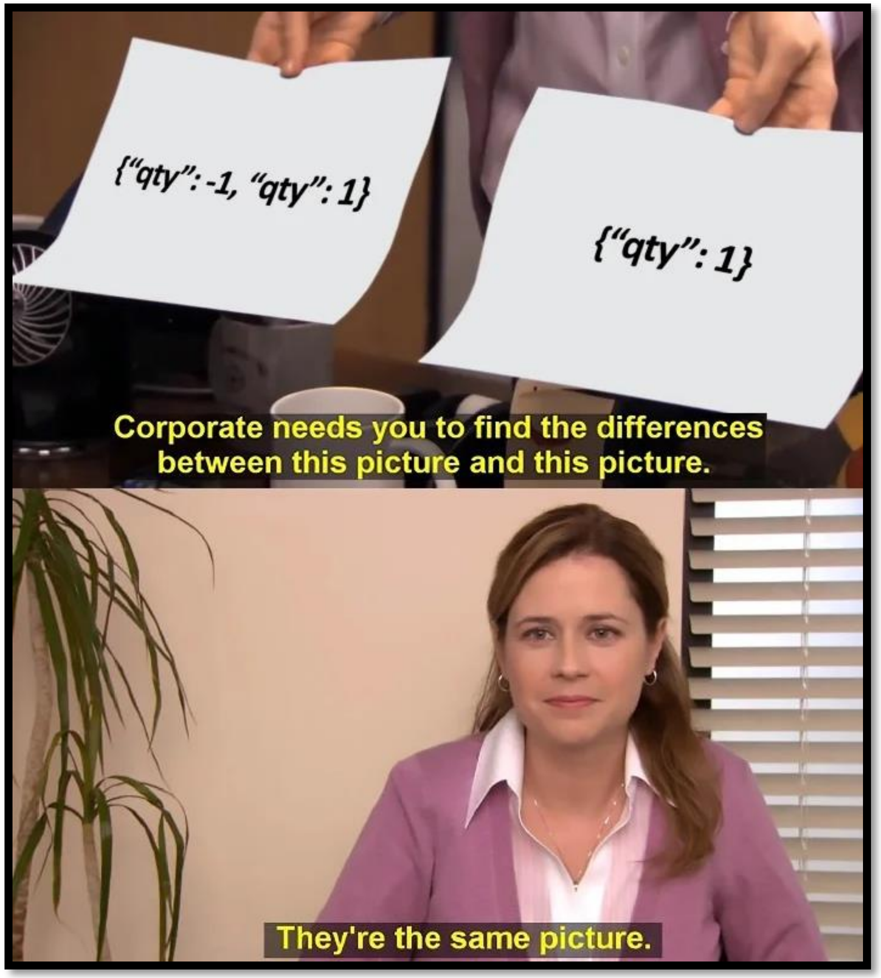


Probably the boldest design decision I made was to not put a version number on JSON so there is no mechanism for revising it. We are stuck with JSON: whatever it is in its current form, that's it.
- Douglas Crockford

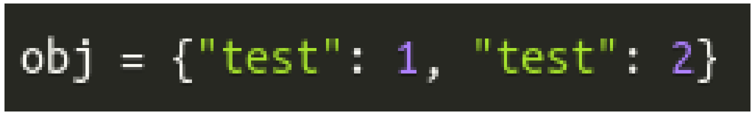
Weired Things

- Inconsistent Duplicate Key Precedence
- Character Truncation
- Comment Truncation
- Float and Integer Representation

Inconsistent Duplicate Key Precedence


{"qty": -1, "qty": 1}
{"qty": 1}
Corporate needs you to find the differences between this picture and this picture.
They're the same picture.

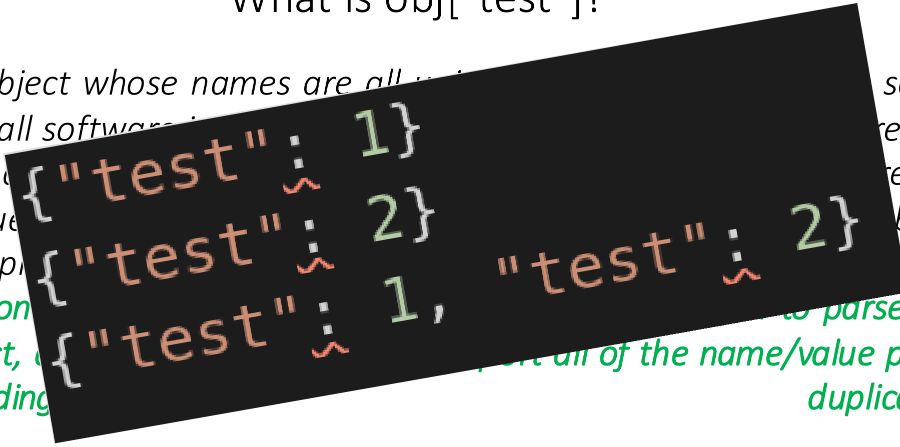
INCONSISTENT DUPLICATE KEY PRECEDENCE



```
obj = {"test": 1, "test": 2}
```

What is obj["test"]?

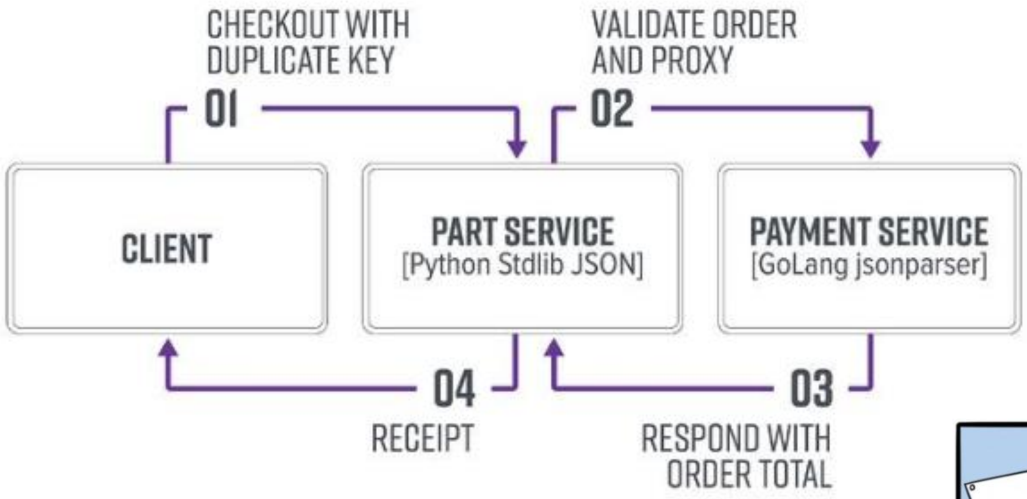
An object whose names are all u *sense*
that all softw *ree on*
the n *e not*
unique *bject*
is unpr *alue*
pair on *to parse the*
object, *rt all of the name/value pairs,*
including *duplicates.*

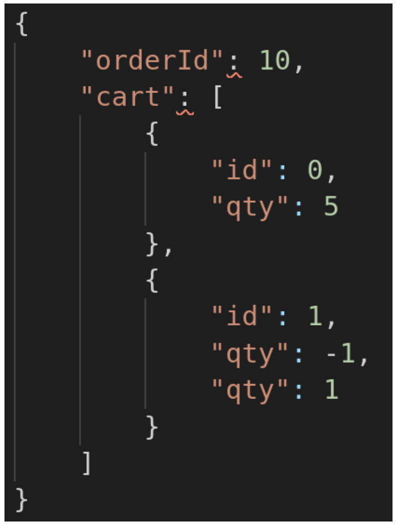


```
{"test": 1}
{"test": 2}
{"test": 1, "test": 2}
```

JSON parsing libraries have been observed to differ as to whether or not they make the ordering of object members visible to calling software. Implementations whose behavior does not depend on member ordering will be interoperable in the sense that they will not be affected by these differences.
- Source: RFC 8259

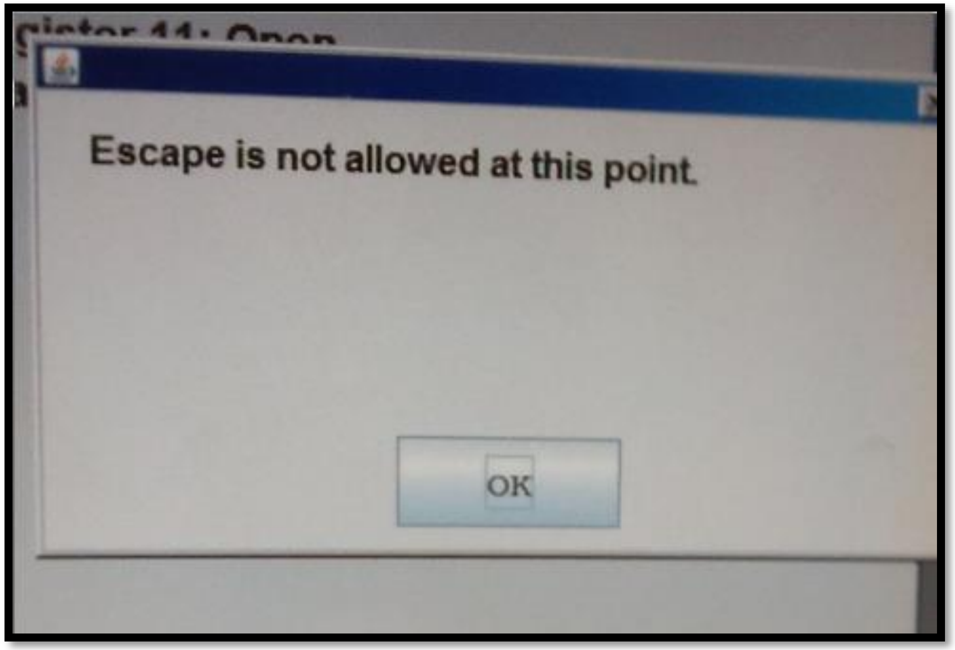
Inconsistent Duplicate Key Precedence


CHECKOUT WITH DUPLICATE KEY
01
VALIDATE ORDER AND PROXY
02
CLIENT
PART SERVICE [Python Stdlib JSON]
PAYMENT SERVICE [GoLang jsonparser]
04
RECEIPT
03
RESPOND WITH ORDER TOTAL

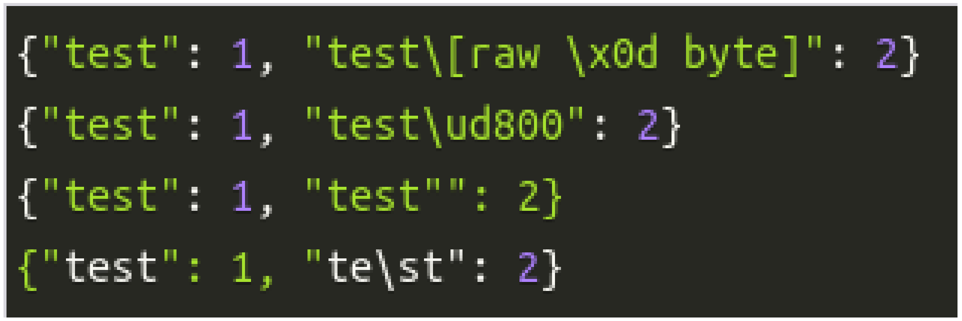


```
{
    "orderId": 10,
    "cart": [
        {
            "id": 0,
            "qty": 5
        },
        {
            "id": 1,
            "qty": -1,
            "qty": 1
        }
    ]
}
```

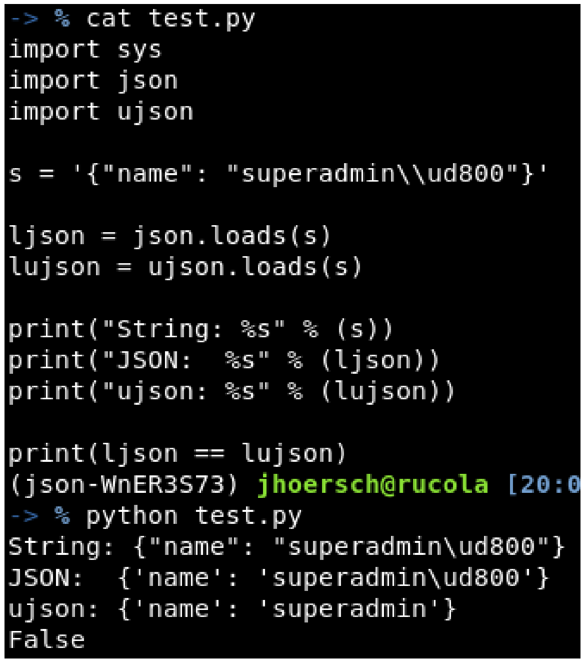
Character Truncation


Escape is not allowed at this point.

Character Truncation

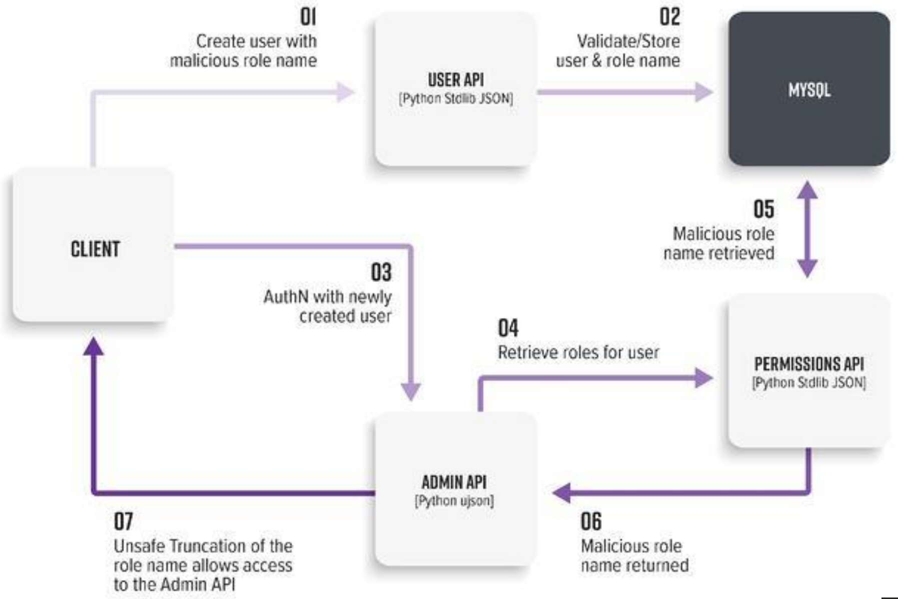


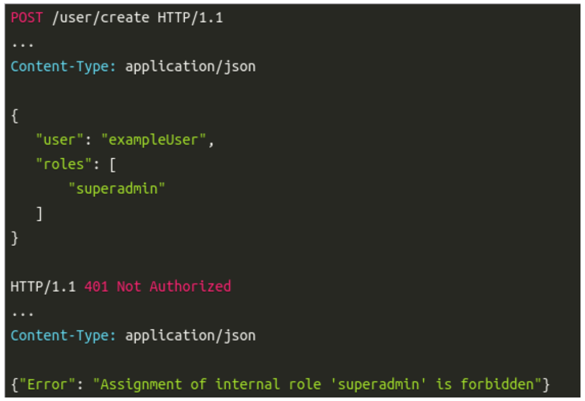
```
{"test": 1, "test\[raw \x0d byte]": 2}
{"test": 1, "test\ud800": 2}
{"test": 1, "test"": 2}
{"test": 1, "te\st": 2}
```



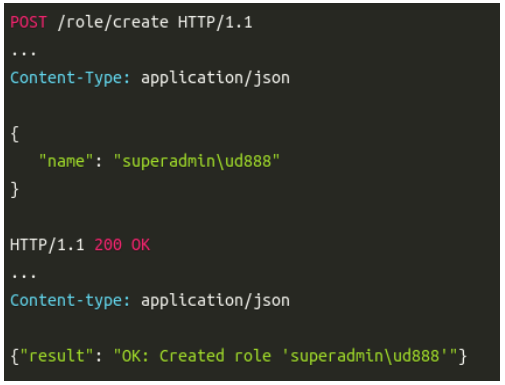
```
-> % cat test.py
import sys
import json
import ujson

s = '{"name": "superadmin\\ud800"}'

ljson = json.loads(s)
lujson = ujson.loads(s)

print("String: %s" % (s))
print("JSON:  %s" % (ljson))
print("ujson: %s" % (lujson))

print(ljson == lujson)
(json-WnER3S73) jhoersch@rucola [20:0
-> % python test.py
String: {"name": "superadmin\ud800"}
JSON:  {'name': 'superadmin\ud800'}
ujson: {'name': 'superadmin'}
False
```

Character Truncation

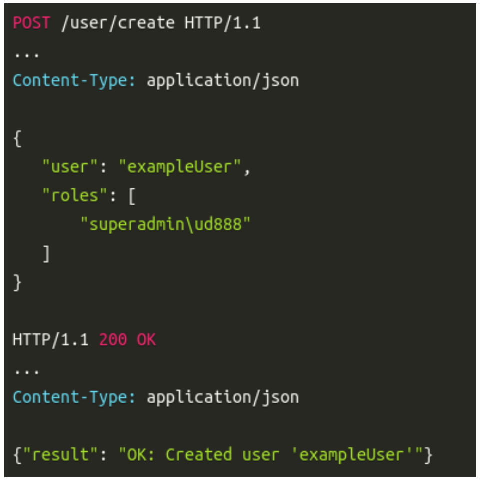

01
Create user with malicious role name
USER API
[Python Stdlib JSON]
02
Validate/Store user & role name
MYSQL
CLIENT
05
Malicious role name retrieved
03
AuthN with newly created user
04
Retrieve roles for user
PERMISSIONS API
[Python Stdlib JSON]
ADMIN API
[Python ujson]
07
Unsafe Truncation of the role name allows access to the Admin API
06
Malicious role name returned



```
POST /user/create HTTP/1.1
...
Content-Type: application/json

{
   "user": "exampleUser",
   "roles": [
       "superadmin"
   ]
}

HTTP/1.1 401 Not Authorized
...
Content-Type: application/json

{"Error": "Assignment of internal role 'superadmin' is forbidden"}
```



```
POST /role/create HTTP/1.1
...
Content-Type: application/json

{
   "name": "superadmin\ud888"
}

HTTP/1.1 200 OK
...
Content-type: application/json

{"result": "OK: Created role 'superadmin\ud888'"}
```



```
POST /user/create HTTP/1.1
...
Content-Type: application/json

{
   "user": "exampleUser",
   "roles": [
       "superadmin\ud888"
   ]
}

HTTP/1.1 200 OK
...
Content-Type: application/json

{"result": "OK: Created user 'exampleUser'"}
```

Comment Truncation

Comment Truncation

```
obj = {"test": valWithoutQuotes, keyWithoutQuotes: "test" /* Comment support */}
```

```
obj = {"description": "Duplicate with comments", "test": 2, "extra": /*, "test": 1, "extra2": */}
```

GoLang's Go Jay Library

```
{
    "description": "Duplicate with comments",
    "test": 2,
    "extra": ""
}
```

Java's JSON-iterator library

```
{
    "description": "Duplicate with comments",
    "extra": "/*",
    "extra2": "*/",
    "test": 1
}
```

```
obj = {"description": "Comment support", "test": 1, "extra": "a"/*, "test": 2, "extra2": "b"*/}
```

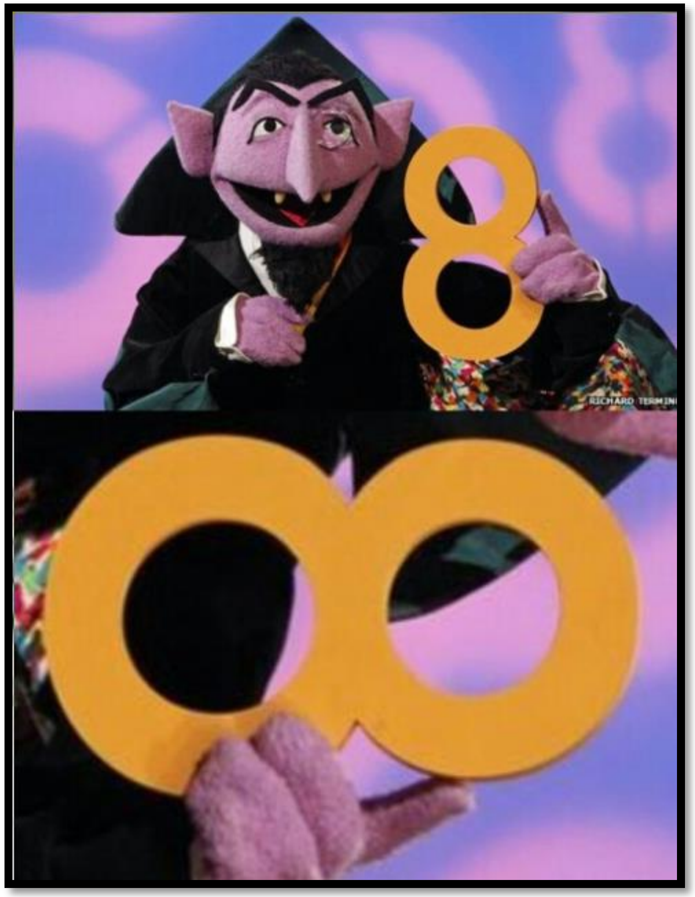
Java's GSON library

```
{
    "description":"Comment support",
    "test":1,
    "extra":"a"
}
```

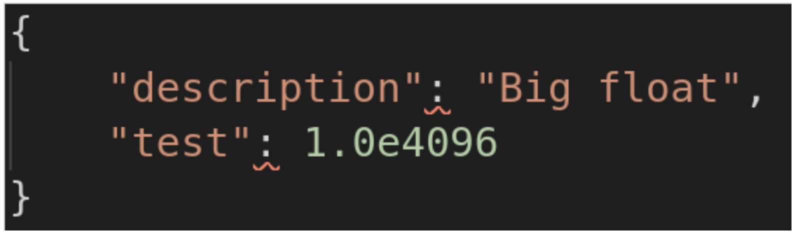
Ruby's simdjson library

```
{
    "description":"Comment support",
    "test":2,
    "extra":"a",
    "extra2":"b"
}
```

Float and Integer Representation



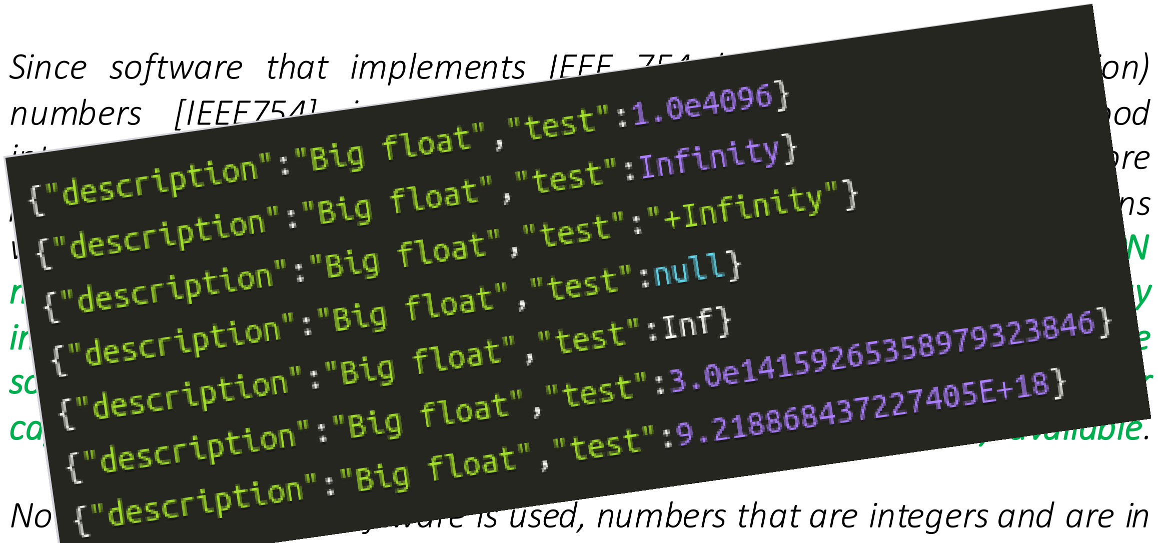
Float and Integer Representation



```
{
    "description": "Big float",
    "test": 1.0e4096
}
```

What is obj["test"]?

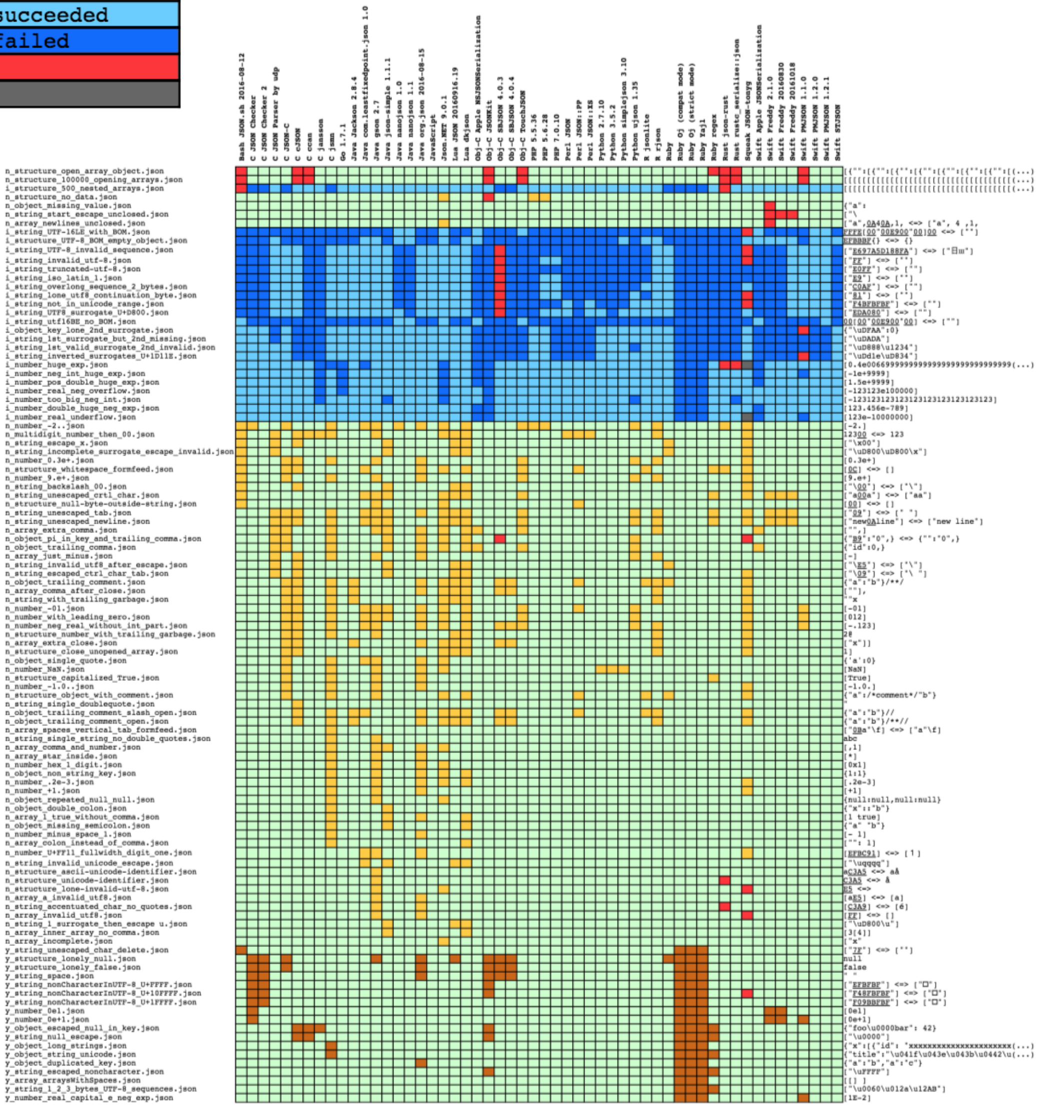
Since software that implements IEEE 754 … on)
numbers [IEEE754 … od
in … re
… ns
… N
r … y
ir … e
sc … r
ca … available.



```
{"description":"Big float","test":1.0e4096}
{"description":"Big float","test":Infinity}
{"description":"Big float","test":"+Infinity"}
{"description":"Big float","test":null}
{"description":"Big float","test":Inf}
{"description":"Big float","test":3.0e14159265358979323846}
{"description":"Big float","test":9.218868437227405E+18}
{"description":"Big float",
```

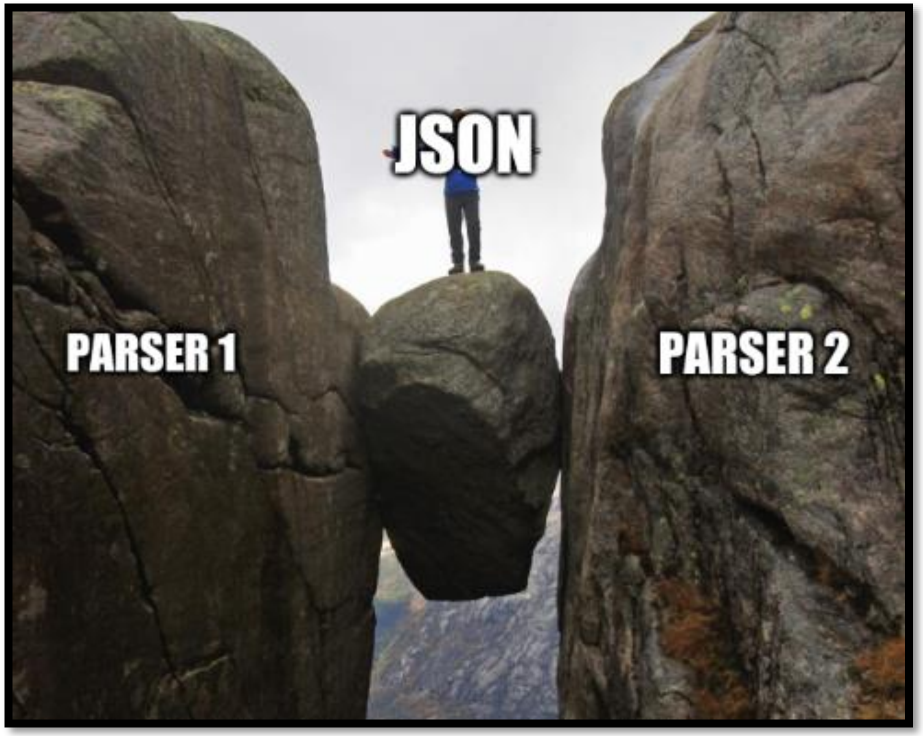
*No … ware is used, numbers that are integers and are in the range [-(2**53)+1, (2**53)-1] are interoperable in the sense that implementations will agree exactly on their numeric values.*

- 🟩 expected result
- 🟧 parsing should have succeeded but failed
- 🟨 parsing should have failed but succeeded
- 🟦 result undefined, parsing succeeded
- 🟦 result undefined, parsing failed
- 🟥 parser crashed
- ⬛ timeout



https://seriot.ch/projects/parsing_json.html

Takeaway



If there are multiple parser involved, **check edge cases**!