json

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

gsyc-profes (arroba) gsyc.urjc.es

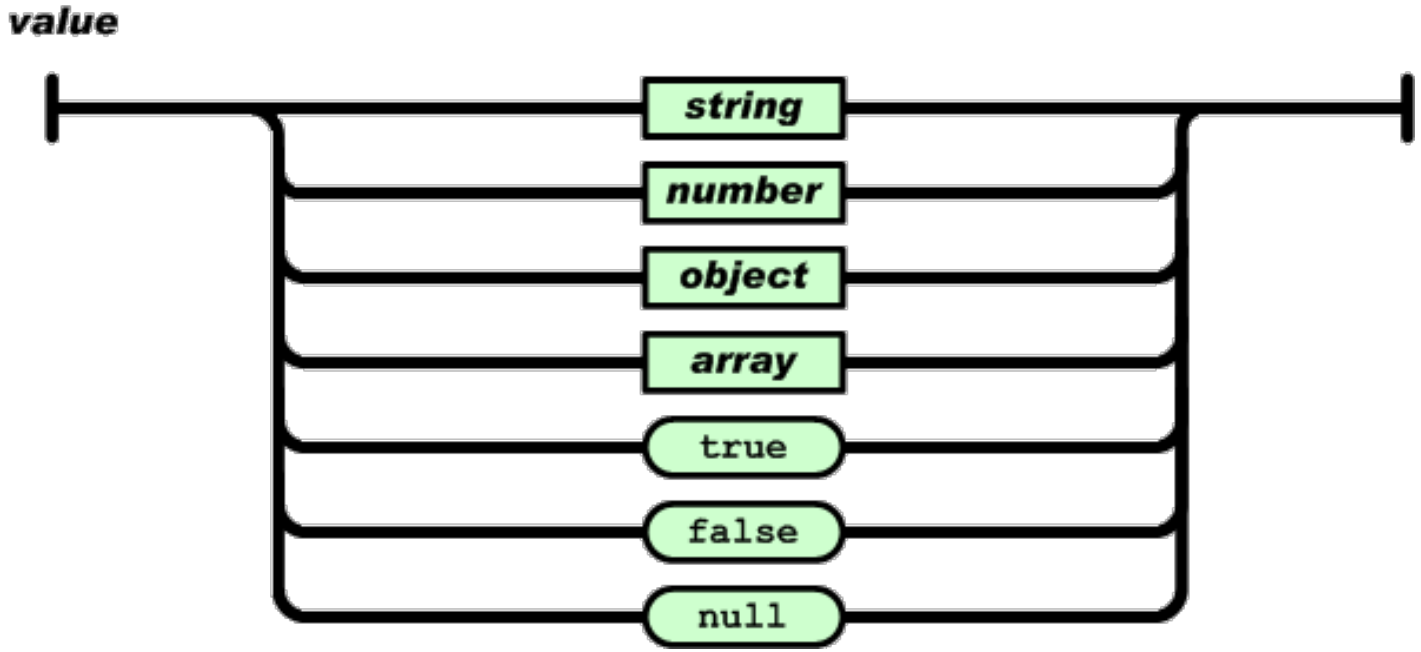
Marzo de 2016

JSON

Es un formato ligero para intercambiar datos, independiente del lenguaje de programación y de la plataforma

- Estándar abierto, RFC 4627, año 2006
- Originalmente se consideraba subconjunto del lenguaje JavaScript y se denominaba *JavaScript Object Notation*, aunque ya no es parte de JavaScipt
- Diseñado como alternativa a XML, más ligero. Actualmente es más popular que XML
- Carece de algunas características de XML, por ejemplo gramática o diferencia texto entre texto y metadato

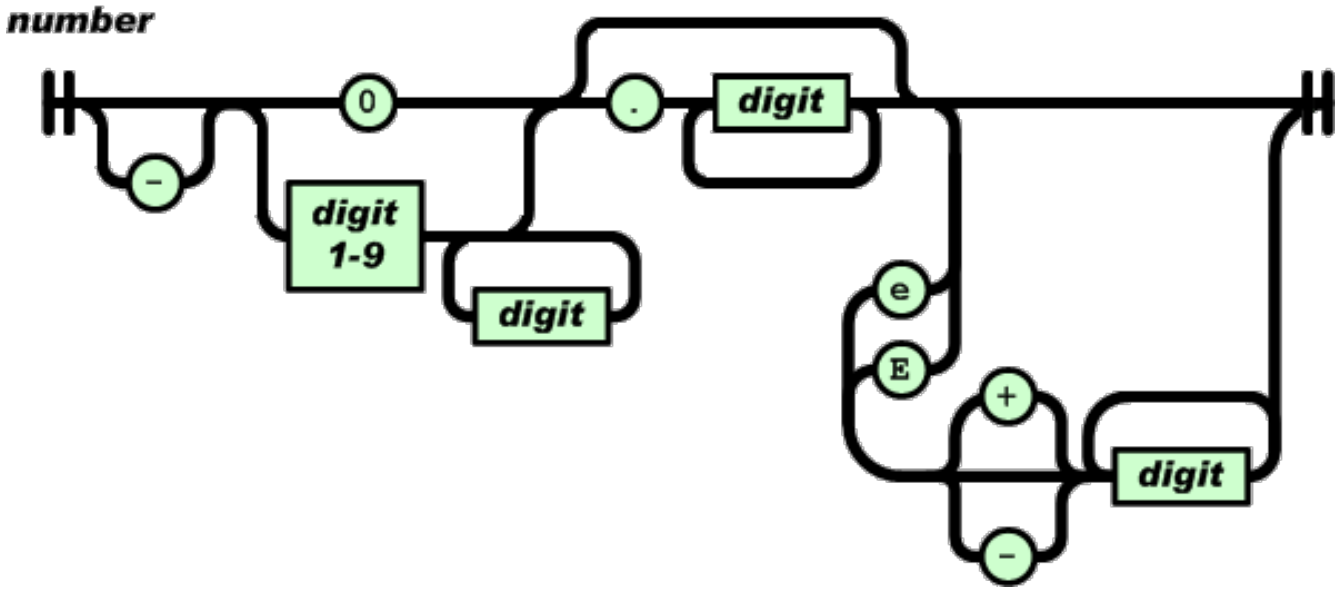
Value



Fuente:json.org

- Un valor json puede ser un número, una cadena, un array o un objeto, además de las constantes *true*, *false* y *null*
- En python, las constantes equivalentes son *True*, *False* y *None*

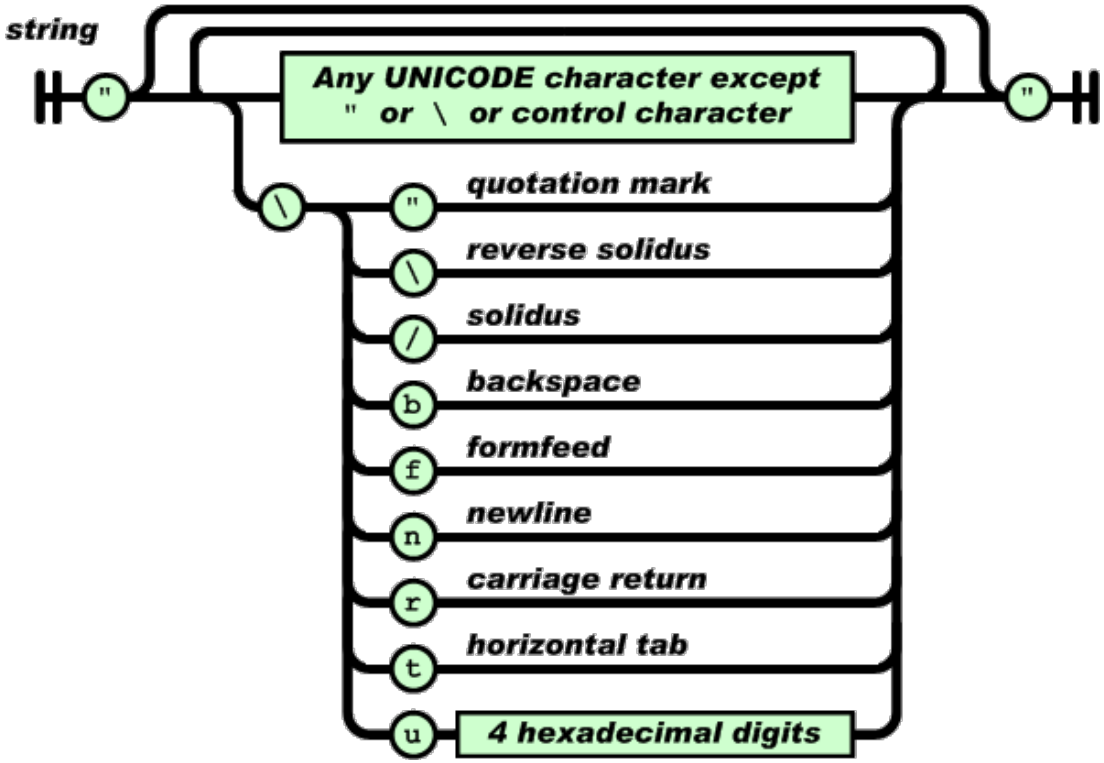
Number



Fuente:json.org

- Los números son como las constantes numéricas de cualquier lenguaje de programación moderno

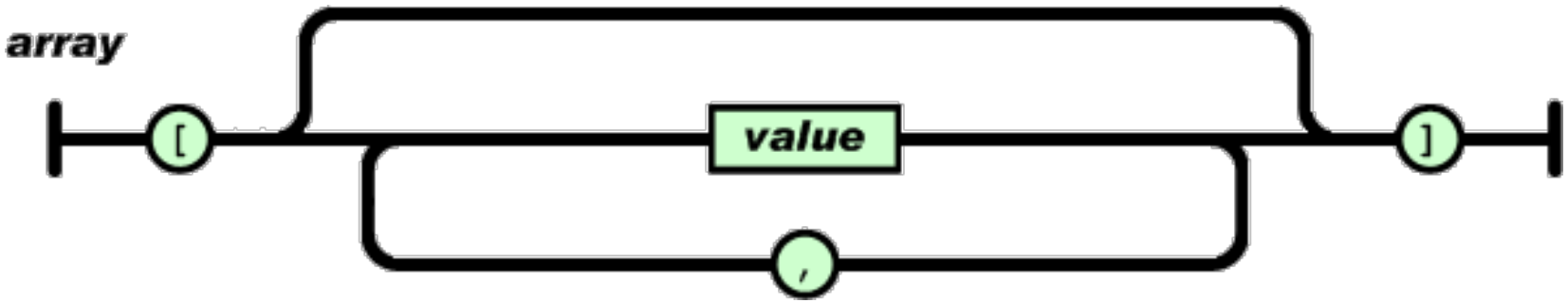
String



Fuente:json.org

- Las cadenas son como las de cualquier lenguaje de programación moderno
- A diferencia de python, el delimitador es únicamente la comilla doble, no la comilla simple

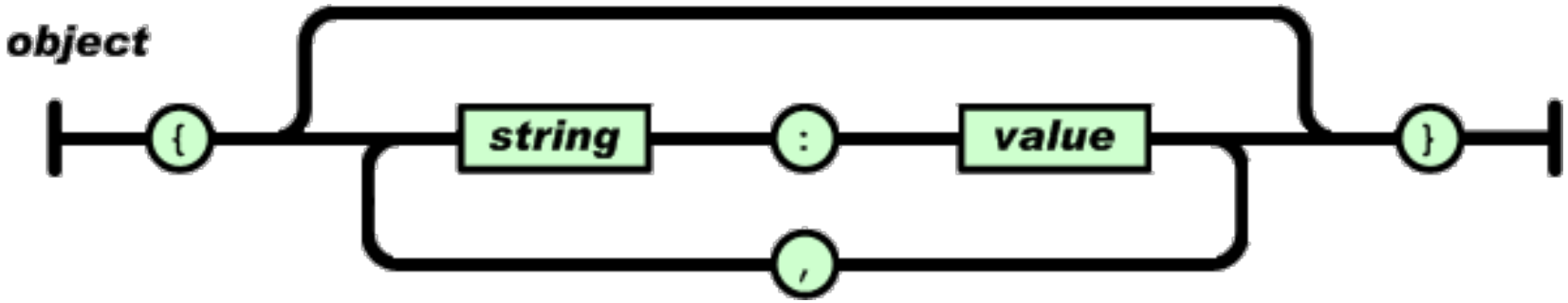
Array



Fuente:json.org

- Un array es una secuencia de valores, separados por comas
- Como las listas de python

Objetos



Fuente:json.org

- Un objeto es una secuencia de pares clave-valor, separados por comas
- Muy similar a los diccionarios de python, excepto porque
  - En python las claves pueden ser cadenas, números o tuplas
  - En json las claves han de ser cadenas

Ejemplos correctos

- `"hola, mundo"`
- `4243.12`
- `-947e-5`
- `null`
- `[1,2,3,4]`

- `[1, "azul", [1,2,3]]`

- ```
  [
      1,
      "azul",
      [
          1,
          2,
          3
      ]
  ]
  ```

- `["as" ,   "dos", "tres"]`

- `["sota", "caballo", "rey"]`

- ```
  [
      "sota",
      "caballo",
      "rey"
  ]
  ```

- `{ "nombre":"Juan", "apellido":"Pérez"}`
- `{ "v1":true, "v2":null, "v3":false}`
- `{ "nombre": "Juan", "notas":[5.5, 7.2, 6.1]}`
-
```
{
    "nombre": "Juan",
    "notas": [
        5.5,
        7.2,
        6.1
    ]
}
```

Ejemplos incorrectos

- `True`
- `'hola, mundo'`
- `{"hola,mundo"}`
- `{1:"uno", 2:"dos"}`

Generación de json desde python

A partir de un objeto python, es muy sencillo generar un valor json

- `json.dumps()` recibe un objeto python, devuelve una cadena con el valor equivalente en json

```
>>> import json
>>> json.dumps(2.23)
'2.23'
>>> json.dumps('Hola,mundo')
'"Hola,mundo"'
>>> json.dumps(True)
'true'
>>> json.dumps(None)
'null'
>>> json.dumps([1,2,3])
'[1, 2, 3]'
>>> json.dumps({1:"uno",2:"dos"})
'{"1": "uno", "2": "dos"}'
```

`json.dumps()` admite parámetros adicionales

- `sort_keys=True`
  ordena las claves por orden alfabético
- `indent=4`
  hace *pretty printing*, tabulando con el número de espacios indicado (4 en el ejemplo)
- `sort_keys=True`
  ordena las claves alfabéticamente

```
>>> print json.dumps({"nombre":"Juan","apellido":"Blanco"},
    sort_keys=True, indent=4)
{
    "apellido": "Blanco",
    "nombre": "Juan"
}
```

Decodificación de json desde python

Es muy sencillo leer valores json desde python

- `json.loads()` recibe una cadena con un valor json, devuelve un objeto python

```
>>> print json.loads('9.5')
9.5
>>> print json.loads('[1,2,3]')
[1, 2, 3]
>>> print json.loads('null')
None
```

- `json.loads()` recibe una cadena con el valor json. Si el valor es a su vez una cadena, tendremos una cadena dentro de una cadena
- Como el delimitador de cadena en json es la comilla doble, en python es recomendable emplear como delimitador la comilla simple (es más legible que escapar la comilla doble)

```
>>> print json.loads('"Hola,mundo"')
Hola,mundo
>>> print json.loads('{"nombre":"Juan","apellido":"Blanco"}')
{u'nombre': u'Juan', u'apellido': u'Blanco'}
```

Otras librerías

Además de la implementación de la librería estándar de python, hay otras librerías para codificar y decodificar json, con la misma API, diseñadas para ofrecer mejor rendimiento en situaciones especiales

- simplejson
- pyson
- Yajl-Py
- ultrajson
- metamagic.json