

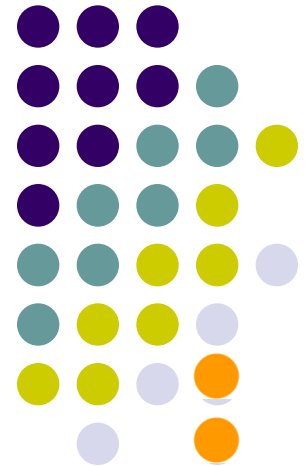
Arquitetura de Software

JSON e YAML

Linguagens para serialização de informações

José Motta Lopes

josemotta@bampli.com



Agenda

- JSON
- Sintaxe JSON
- Exemplo JSON
- JSON x XML
- YAML
- Sintaxe YAML
- API com YAML
- Exemplo YAML



JSON

JAVASCRIPT OBJECT NOTATION

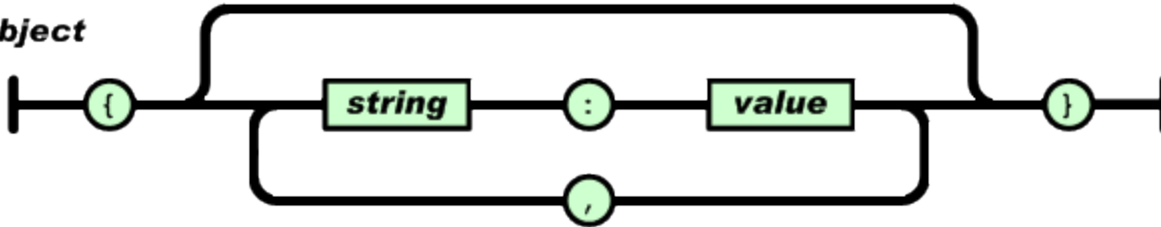


- **Formato leve para intercâmbio de dados**
- **Fácil leitura e escrita para humanos**
- **Fácil para máquinas analisar e gerar**
- **Convenções familiares aos programadores**
- **Linguagens C, C++, C#, Java, Javascript, Perl, Python, etc.**



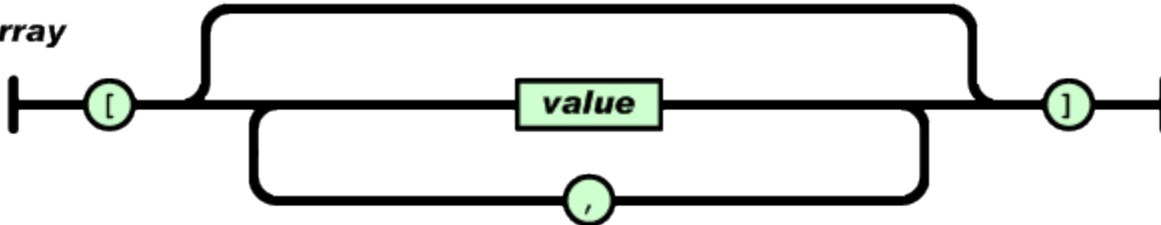
Sintaxe JSON

object



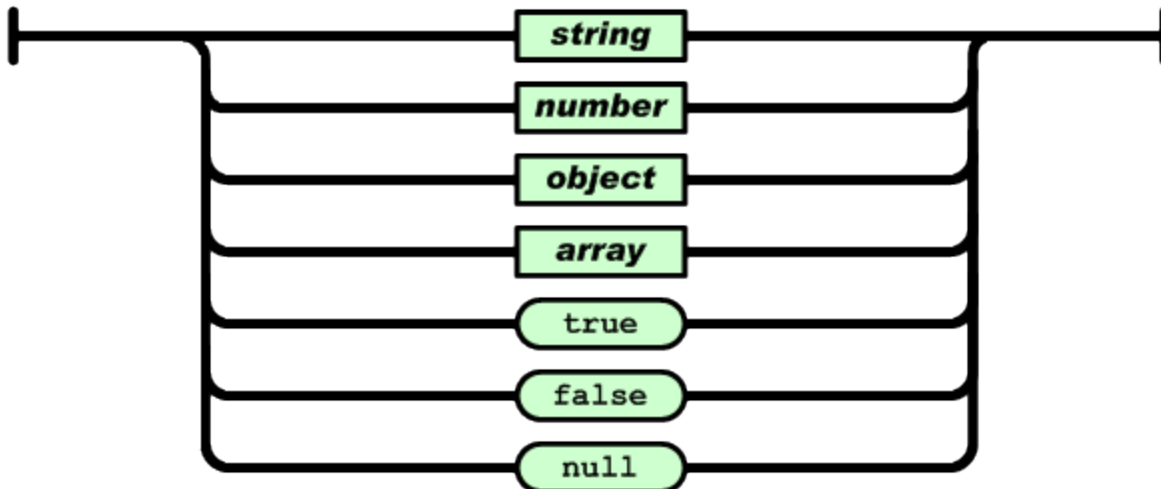
Objeto é um conjunto não ordenado de pares com nome/valor

array



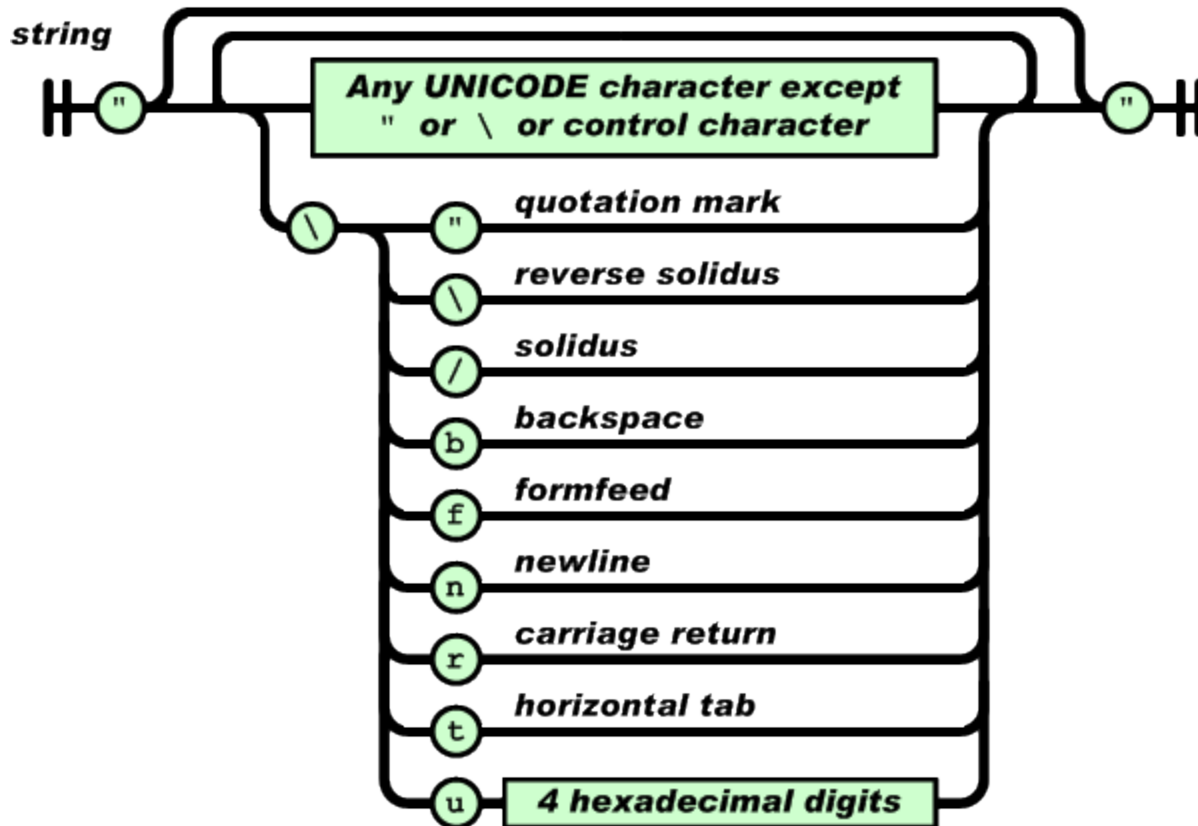
Array é uma coleção ordenada de valores

value



Valor pode ser um string com aspas ou um número, ou true / false / null, ou um objeto ou um array

Sintaxe JSON

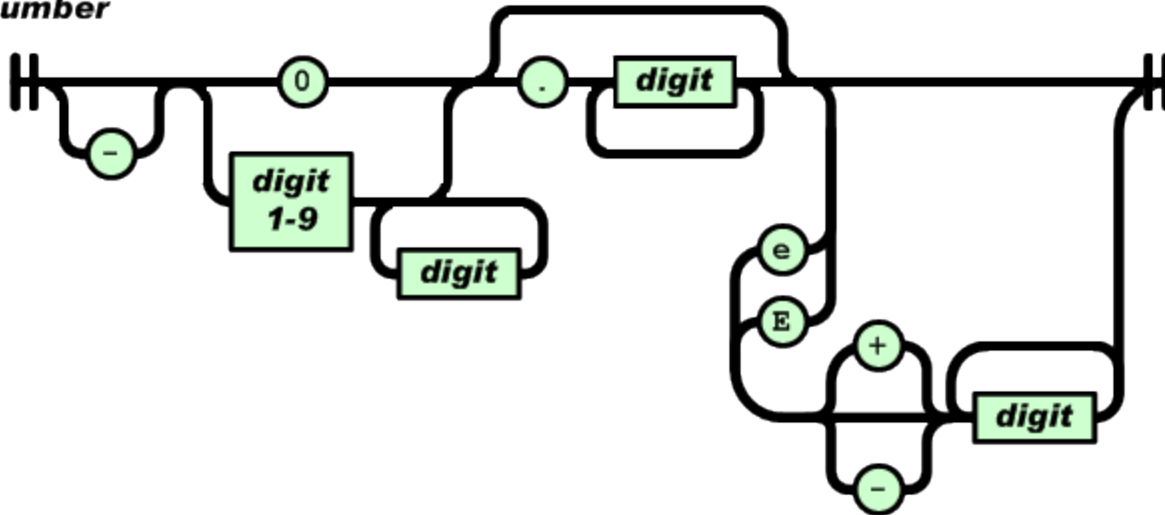


String é uma sequência caracteres UNICODE, envoltos em aspas duplas e utilizando escapes de barra invertida

Sintaxe JSON



number



Número é semelhante aos números C ou Java, exceto que os formatos octal e hexadecimal não são usados

Exemplo JSON



```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

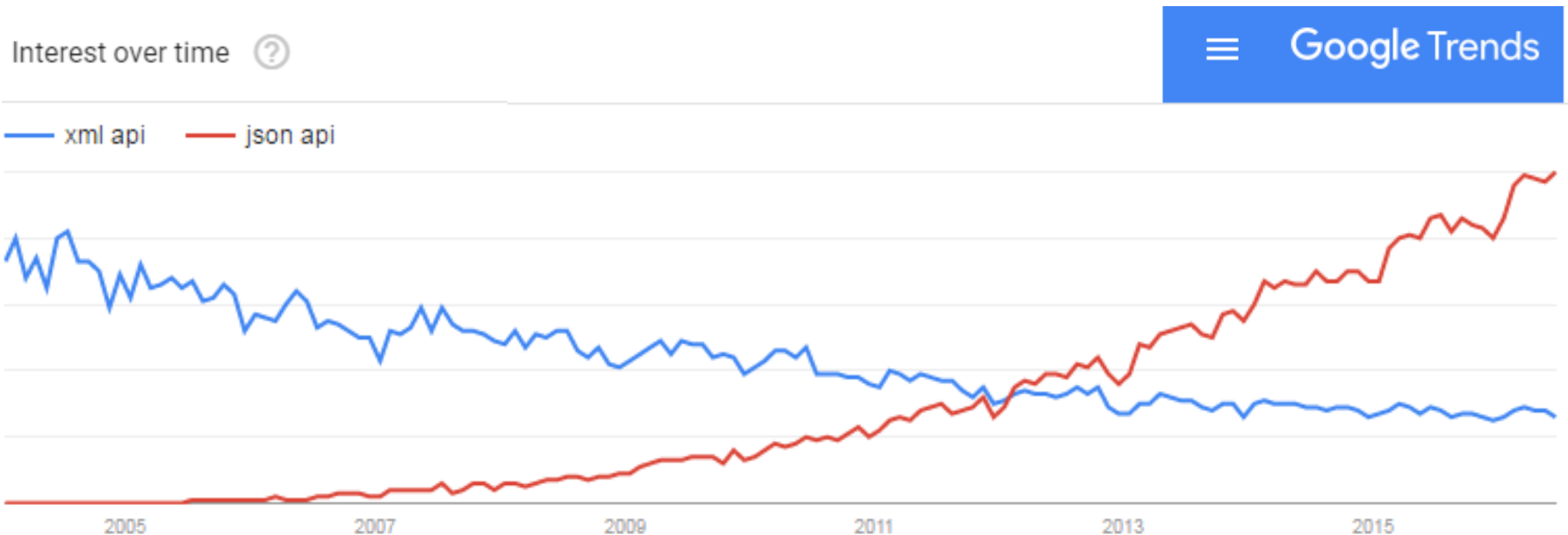
**Possível representação JSON
para descrição de uma pessoa**

JSON x XML



API TENDENDO PARA RESPOSTAS JSON

- XML ficando para trás, mais difícil de ler e analisar
- Legado do mercado corporativo usa XML



YAML



YAML AIN'T MARKUP LANGUAGE

- **Formato leve comumente usado em arquivos de configuração**
- **Linguagem de serialização de informações**
- **Projetada para fácil leitura e escrita de humanos**
- **Fácil para máquinas analisar e gerar**
- **Menos complexa que XML e JSON**

Sintaxe YAML



Sequência Simples em YAML?

- maçã
- banana
- cenoura

Uma lista é especificada com cada membro em uma linha, iniciada com um traço

Sequências Aninhadas em YAML?

- - foo
 - Barra
 - baz

Incluir uma sequência dentro de outra sequência com traço vazio e lista recuada

Sequências Mistas em YAML?

- maçã
- - foo
 - Barra
 - x123
- banana
- cenoura

Sequências podem conter quaisquer dados YAML, incluindo outras sequências

Sequências profundamente aninhadas em YAML?

- - - uno
 - dos

Sequências são aninhadas, com cada nível de recuo representando profundidade

Sintaxe YAML



Mapeamento Simples em YAML?

```
foo: whatever  
bar: coisas
```

Mapeamento é uma lista com chave de dicionário ou hash separada por <:>

Sequência em um mapeamento em YAML?

```
foo: whatever  
Barra:  
  - uno  
  - dos
```

Valor no mapeamento pode ser uma sequência

Mapeamentos aninhados em YAML?

```
foo: whatever  
Barra:  
  fruta: maçã  
  nome: steve  
  esporte: beisebol
```

Valor no mapeamento pode ser outro mapeamento

Mapeamento Misto em YAML?

```
foo: whatever  
Barra:  
  -  
    fruta: maçã  
    nome: steve  
    esporte: beisebol  
  - Mais  
  -  
    python: rochas  
    perl: papers  
    rubi: scissorses
```

Mapeamento pode conter qualquer variedade de mapeamentos e sequências como valores

Sintaxe YAML



Atalho de Mapeamento em Sequência em YAML?

```
- trabalhe em YAML.py:  
  - trabalhar na loja
```

Atalho de Sequência no Mapeamento em YAML?

```
permitir:  
- 'localhost'  
- '% .sourceforge.net'  
- '% .freepan.org'
```

Matriz Inline Simples em YAML?

```
---  
seq: [a, b, c]
```

Hash Inline Simples em YAML?

```
---  
hash: {nome: Steve, foo: bar}
```

Adicione um mapeamento a uma sequência, usando atalho na mesma linha do traço

Traço em uma sequência conta como recuo, não precisa recuar mais espaços

Sequência em linha, sintaxe inline. Separar cada entrada com vírgula e colocar entre colchetes

Mapeamento em linha, sintaxe inline. Cada par valor/chave é separado por <:>, com <,> entre cada entrada. Colocar entre chaves

API com YAML

SWAGGER HUB



```
1  swagger: '2.0'
2  info:
3    version: "1.0.3"
4    title: home
5    description: The API for the Home Starter project
6
7  consumes:
8    - application/json
9  produces:
10   - application/json
11  paths:
12   /devices:
13     get:
14       tags:
15         - Device
16       description: returns all registered devices
17       operationId: getDevices
18       parameters:
19         - in: query
20           name: skip
21           type: integer
22           format: int32
23           description: number of records to skip
24         - in: query
25           name: limit
26           type: integer
27           format: int32
28           description: max number of records to return
29       responses:
30         200:
31           description: All the devices
32           schema:
33             type: array
34             items:
35               type: string
36               format: uri
37               example: http://10.0.0.225:8080
38     post:
39       tags:
40         - Device
41       operationId: register
42       parameters:
43         - in: body
44           name: device
```

DEVICE

Show Comments

GET /devices

returns all registered devices

Parameters Try it out

Name	Description
skip integer(\$int32) (query)	number of records to skip
limit integer(\$int32) (query)	max number of records to return

Responses Response content type: application/json

Code	Description
200	ALL the devices

Example Value | Model

```
[  
  "http://10.0.0.225:8080"  
]
```

API com YAML

SWAGGER HUB



```
1 swagger: '2.0'
2 info:
3   version: "1.0.3"
4   title: home
5   description: The API for the Home Star
6
7 consumes:
8   - application/json
9 produces:
10  - application/json
11 paths:
12  /devices:
13    get:
14      tags:
15        - Device
16      description: returns all registered
17      operationId: getDevices
18      parameters:
19        - in: query
20          name: skip
21          type: integer
22          format: int32
23        - in: query
24          name: limit
25          type: integer
26          format: int32
27          description: max number of records to skip
28      responses:
29        200:
30          description: All the devices
31          schema:
32            type: array
33            items:
34              type: string
35              format: uri
36              example: http://10.0.0.225
37      post:
38        tags:
39          - Device
40        operationId: register
41        parameters:
42          - in: body
43            name: device
44            required: false
```

```
12  /devices:
13    get:
14      tags:
15        - Device
16      description: returns all registered devices
17      operationId: getDevices
18      parameters:
19        - in: query
20          name: skip
21          type: integer
22          format: int32
23        - in: query
24          name: limit
25          type: integer
26          format: int32
27          description: max number of records to
28            return
29      responses:
30        200:
```

O alinhamento importa!

```

1 ---
2 swagger: "2.0"
3 info:
4   description: The API for the Home Starter project
5   version: 1.0.3
6   title: home
7 host: virtserver.swaggerhub.com
8 basePath: /motta/home/1.0.3
9 schemes:
10 - https
11 consumes:
12 - application/json
13 produces:
14 - application/json
15 paths:
16   /devices:
17     get:
18       tags:
19       - Device
20       description: returns all registered devices
21       operationId: getDevices
22       parameters:
23       - name: skip
24         in: query
25         description: number of records to skip
26         required: false
27         type: integer
28         format: int32
29       - name: limit
30         in: query
31         description: max number of records to return
32         required: false
33         type: integer
34         format: int32
35       responses:
36       200:
37         description: All the devices
38         schema:
39           type: array
40           items:
41             type: string
42             format: uri
43             example: http://10.0.0.225:8080
44         responseSchema:
45           type: array
46           items:
47             type: string
48             format: uri
49             example: http://10.0.0.225:8080

```

YAML

```

1 {
2   "swagger" : "2.0",
3   "info" : {
4     "description" : "The API for the Home Starter project",
5     "version" : "1.0.3",
6     "title" : "home"
7   },
8   "host" : "virtserver.swaggerhub.com",
9   "basePath" : "/motta/home/1.0.3",
10  "schemes" : [ "https" ],
11  "consumes" : [ "application/json" ],
12  "produces" : [ "application/json" ],
13  "paths" : {
14    "/devices" : {
15      "get" : {
16        "tags" : [ "Device" ],
17        "description" : "returns all registered devices",
18        "operationId" : "getDevices",
19        "parameters" : [ {
20          "name" : "skip",
21          "in" : "query",
22          "description" : "number of records to skip",
23          "required" : false,
24          "type" : "integer",
25          "format" : "int32"
26        }, {
27          "name" : "limit",
28          "in" : "query",
29          "description" : "max number of records to return",
30          "required" : false,
31          "type" : "integer",
32          "format" : "int32"
33        } ],
34        "responses" : {
35          "200" : {
36            "description" : "All the devices",
37            "schema" : {
38              "type" : "array",
39              "items" : {
40                "type" : "string",
41                "format" : "uri",
42                "example" : "http://10.0.0.225:8080"
43              }
44            }
45          }
46        }
47      },

```

JSON

