

Referência Rápida de JSON

Sintaxe, tipos de dados, objetos, arrays, jq

Sintaxe

Regras

| | |
|--------------------------|---|
| <code>{ }</code> | Objeto (pares chave-valor não ordenados) |
| <code>[]</code> | Array (lista ordenada de valores) |
| "key": value | Chaves devem ser strings com aspas duplas |
| No trailing comma | O último item não deve ter vírgula |
| No comments | JSON não permite comentários |

Exemplo Mínimo

```
{
  "name": "Alice",
  "age": 30,
  "active": true
}
```

Tipos de Dados

Seis Tipos de Valor

| | |
|---------------------|-------------------------------------|
| "string" | Texto UTF-8 com aspas duplas |
| 42 / 3.14 | Número (inteiro ou ponto flutuante) |
| true / false | Booleano |
| null | Nulo (ausência de valor) |
| <code>{ }</code> | Objeto |
| <code>[]</code> | Array |

Sequências de Escape em Strings

| | |
|---------------------|------------------------------|
| <code>\"</code> | Aspas duplas |
| <code>\\</code> | Barra invertida |
| <code>\n \t</code> | Nova linha, tabulação |
| <code>\uXXXX</code> | Escape Unicode (hexadecimal) |

Objetos

Sintaxe de Objeto

```
{
  "id": 1,
  "name": "Widget",
  "tags": ["new", "sale"]
}
```

Regras

| | |
|----------------|---|
| Keys | Devem ser strings únicas com aspas duplas |
| Values | Qualquer tipo JSON válido |
| Order | A ordem das chaves não é garantida |
| Nesting | Objetos podem conter objetos |

Arrays

Sintaxe de Array

```
[1, "two", true, null, {"key": "val"}]
```

Array de Tipos Mistos

```
{
  "matrix": [[1, 2], [3, 4]],
  "empty": []
}
```

Regras

| | |
|--------------------|--|
| Ordered | Elementos mantêm a ordem de inserção |
| Mixed types | Itens do array podem ser de tipos diferentes |
| Indexing | Base zero (na maioria das linguagens) |

Aninhamento

Estrutura Aninhada

```
{
  "user": {
    "name": "Alice",
    "address": { "city": "Boston" },
    "scores": [95, 88, 72]
  }
}
```

Padrões de Acesso

| | |
|----------------------------|---|
| obj.user.name | Notação de ponto (JavaScript) |
| obj["user"]["name"] | Notação de colchetes |
| obj.user.scores[0] | Índice de array dentro de objeto aninhado |

Validação de Schema

Exemplo de JSON Schema

```
{
  "type": "object",
  "properties": {
    "name": { "type": "string" },
    "age": { "type": "integer", "minimum": 0 }
  },
  "required": ["name"]
}
```

Palavras-chave de Schema

| | |
|------------------------------|---|
| type | string, number, integer, boolean, object, array, null |
| required | Array de nomes de propriedades obrigatórias |
| properties | Define as propriedades esperadas do objeto |
| enum | Restringe a um conjunto fixo de valores |
| minLength / maxLength | Restrições de comprimento de string |
| minimum / maximum | Restrições de intervalo numérico |

Básico de jq

Filtros Comuns

| | |
|-----------------------------|--------------------------------------|
| . | Identidade — passa a entrada adiante |
| .key | Acessa chave do objeto |
| .key.nested | Acessa chave aninhada |
| .[0] | Primeiro elemento do array |
| .[] | Itera todos os elementos do array |
| select(.age > 20) | Filtra por condição |
| map(.name) | Transforma cada elemento |
| length | Comprimento do array ou da string |
| keys | Chaves do objeto como array |

Exemplos de jq

```
echo '{"a":1}' | jq '.a' # 1
echo '[1,2,3]' | jq 'map(. * 2)' # [2,4,6]
cat data.json | jq '.users[].name'
cat data.json | jq '.[] | select(.active)'
```

Padrões Comuns

Resposta de API

```
{
  "status": 200,
  "data": [{"id": 1, "name": "Alice"}],
  "meta": {"total": 42, "page": 1}
}
```

Arquivo de Configuração

```
{
  "host": "localhost",
  "port": 8080,
  "debug": false,
  "features": ["auth", "logging"]
}
```

Dicas

| | |
|----------------------|--|
| Validate | Use jsonlint ou python -m json.tool |
| Pretty print | jq . file.json ou python -m json.tool |
| JSONL | Um objeto JSON por linha (delimitado por nova linha) |
| JSON5 / JSONC | Extensões que permitem comentários e vírgulas finais |