

# JSON クイックリファレンス

構文、データ型、オブジェクト、配列、jq

## 構文

### ルール

<code>{ }</code>	オブジェクト（順序なしのキーと値のペア）
<code>[ ]</code>	配列（値の順序付きリスト）
<b>"key": value</b>	キーはダブルクォートで囲む必要がある
<b>No trailing comma</b>	最後の要素にカンマをつけてはいけない
<b>No comments</b>	JSON はコメントを許可しない

### 最小限の例

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

## データ型

### 6 つの値の型

<b>"string"</b>	ダブルクォートで囲んだ UTF-8 テキスト
<b>42 / 3.14</b>	数値（整数または浮動小数点）
<b>true / false</b>	ブール値
<b>null</b>	null（値の不在）
<code>{ }</code>	オブジェクト
<code>[ ]</code>	配列

### 文字列のエスケープシーケンス

<code>\"</code>	ダブルクォート
<code>\\</code>	バックslash
<code>\n \t</code>	改行、タブ
<code>\uXXXX</code>	Unicode エスケープ（16 進数）

## オブジェクト

### オブジェクト構文

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

### ルール

<b>Keys</b>	ユニークなダブルクォート文字列でなければならない
<b>Values</b>	任意の有効な JSON 型
<b>Order</b>	キーの順序は保証されない
<b>Nesting</b>	オブジェクトはオブジェクトを含むことができる

## 配列

### 配列構文

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

### 混合型配列

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

### ルール

<b>Ordered</b>	要素は挿入順序を維持する
<b>Mixed types</b>	配列の要素は異なる型でよい
<b>Indexing</b>	ゼロベース（ほとんどの言語）

## ネスト

### ネスト構造

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

### アクセスパターン

<b>obj.user.name</b>	ドット記法（JavaScript）
<b>obj["user"]["name"]</b>	ブラケット記法
<b>obj.user.scores[0]</b>	ネストされたオブジェクト内の配列インデックス

## スキーマバリデーション

### JSON Schema の例

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

### スキーマキーワード

<b>type</b>	string、number、integer、boolean、object、array、null
<b>required</b>	必須プロパティ名の配列
<b>properties</b>	期待するオブジェクトプロパティを定義
<b>enum</b>	固定値セットに制限
<b>minLength / maxLength</b>	文字列の長さ制約
<b>minimum / maximum</b>	数値の範囲制約

## jq の基本

### よく使うフィルター

<b>.</b>	恒等 — 入力そのまま渡す
<b>.key</b>	オブジェクトキーにアクセス
<b>.key.nested</b>	ネストされたキーにアクセス
<b>.[0]</b>	配列の最初の要素
<b>.[ ]</b>	すべての配列要素を反復
<b>select(.age &gt; 20)</b>	条件でフィルタリング
<b>map(.name)</b>	各要素を変換
<b>length</b>	配列の長さまたは文字列の長さ
<b>keys</b>	オブジェクトのキーを配列として取得

### jq の例

```
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)'
```

### よく使うパターン

#### API レスポンス

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

## 設定ファイル

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

## ヒント

<b>Validate</b>	jsonlint または python -m json.tool を使用
<b>Pretty print</b>	jq .file.json または python -m json.tool
<b>JSONL</b>	1 行に 1 つの JSON オブジェクト（改行区切り）
<b>JSON5 / JSONC</b>	コメントと末尾カンマを許可する拡張