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

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

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

文字列のエスケープシーケンス

`\"`	ダブルクォート
`\\`	バックスラッシュ
`\n  \t`	改行、タブ
`\uXXXX`	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

`Keys`	ユニークなダブルクォート文字列でなければならない
`Values`	任意の有効な JSON 型
`Order`	キーの順序は保証されない
`Nesting`	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

`Ordered`	要素は挿入順序を維持する
`Mixed types`	配列の要素は異なる型でよい
`Indexing`	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

`obj.user.name`	ドット記法（JavaScript）
`obj["user"]["name"]`	ブラケット記法
`obj.user.scores[0]`	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

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

jq の基本

よく使うフィルター

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

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