

# JSON 빠른 참조

문법, 데이터 타입, 객체, 배열, jq

## 문법

### 규칙

<code>{ }</code>	객체 (순서 없는 키-값 쌍)
<code>[ ]</code>	배열 (순서 있는 값 목록)
<b>"key": value</b>	키는 반드시 큰따옴표로 감싼 문자열이어야 함
<b>후행 심포 없음</b>	마지막 항목에는 심포 불가
<b>주석 없음</b>	JSON은 주석을 허용하지 않음

### 최소 예시

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

## 데이터 타입

### 여섯 가지 값 타입

<b>"string"</b>	큰따옴표로 감싼 UTF-8 텍스트
<b>42 / 3.14</b>	숫자 (정수 또는 부동소수점)
<b>true / false</b>	불리언
<b>null</b>	null (값의 부재)
<code>{ }</code>	객체
<code>[ ]</code>	배열

### 문자열 이스케이프 시퀀스

<code>\"</code>	큰따옴표
<code>\\</code>	백슬래시
<code>\n \t</code>	줄바꿈, 탭
<code>\uXXXX</code>	유니코드 이스케이프 (16진수)

## 객체

### 객체 문법

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

### 규칙

<b>Keys</b>	고유한 큰따옴표 문자열이어야 함
<b>Values</b>	유효한 JSON 타입이면 모두 허용
<b>Order</b>	키 순서는 보장되지 않음
<b>Nesting</b>	객체 안에 객체 중첩 가능

## 배열

### 배열 문법

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

### 혼합 타입 배열

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

### 규칙

<b>Ordered</b>	요소는 삽입 순서를 유지
<b>Mixed types</b>	배열 항목은 서로 다른 타입 가능
<b>Indexing</b>	0 기반 인덱스 (대부분의 언어)

## 중첩

### 중첩 구조

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

### 접근 패턴

<b>obj.user.name</b>	점 표기법 (JavaScript)
<b>obj["user"]["name"]</b>	대괄호 표기법
<b>obj.user.scores[0]</b>	중첩 객체 내 배열 인덱스

## 스키마 검증

### JSON Schema 예시

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

### 스키마 키워드

<b>type</b>	string, number, integer, boolean, object, array, null
<b>required</b>	필수 속성 이름 배열
<b>properties</b>	기대하는 객체 속성 정의
<b>enum</b>	고정된 값 집합으로 제한
<b>minLength / maxLength</b>	문자열 길이 제약
<b>minimum / maximum</b>	숫자 범위 제약

## jq 기초

### 주요 필터

<b>.</b>	항등 — 입력을 그대로 통과
<b>.key</b>	객체 키 접근
<b>.key.nested</b>	중첩 키 접근
<b>.[0]</b>	첫 번째 배열 요소
<b>.[ ]</b>	모든 배열 요소 순회
<b>select(.age &gt; 20)</b>	조건으로 필터링
<b>map(.name)</b>	각 요소 변환
<b>length</b>	배열 또는 문자열 길이
<b>keys</b>	객체 키를 배열로

### jq 예시

```
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)'
```

## 주요 패턴

### API 응답

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

## 설정 파일

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

## 팁

<b>Validate</b>	jsonlint 또는 python -m json.tool 사용
<b>Pretty print</b>	jq . file.json 또는 python -m json.tool
<b>JSONL</b>	줄마다 하나의 JSON 객체 (개행 구분)
<b>JSON5 / JSONC</b>	주석 및 후행 심포를 허용하는 확장 형식