JSON 빠른 참조

문법, 데이터 타입, 객체, 배열, jq

문법

규칙

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

최소 예시

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

데이터 타입

여섯 가지 값 타입

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

문자열 이스케이프 시퀀스

`\"`	큰따옴표
`\\`	백슬래시
`\n  \t`	줄바꿈, 탭
`\uXXXX`	유니코드 이스케이프 (16진수)

객체

객체 문법

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

규칙

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

배열

배열 문법

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

혼합 타입 배열

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

규칙

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

중첩

중첩 구조

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

접근 패턴

`obj.user.name`	점 표기법 (JavaScript)
`obj["user"]["name"]`	대괄호 표기법
`obj.user.scores[0]`	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

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

jq 기초

주요 필터

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

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