

JSON Riferimento Rapido

Sintassi, tipi di dato, oggetti, array, jq

Sintassi

Regole

<code>{ }</code>	Oggetto (coppie chiave-valore non ordinate)
<code>[]</code>	Array (lista ordinata di valori)
"key": value	Le chiavi devono essere stringhe con virgolette doppie
No virgola finale	L'ultimo elemento non deve avere la virgola
Nessun commento	JSON non ammette commenti

Esempio Minimale

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

Tipi di Dato

Sei Tipi di Valore

"string"	Testo UTF-8 con virgolette doppie
42 / 3.14	Numero (intero o floating point)
true / false	Booleano
null	Null (assenza di valore)
<code>{ }</code>	Oggetto
<code>[]</code>	Array

Sequenze di Escape nelle Stringhe

<code>\"</code>	Virgolette doppie
<code>\\</code>	Backslash
<code>\n \t</code>	A capo, tabulazione
<code>\uXXXX</code>	Escape Unicode (esadecimale)

Oggetti

Sintassi degli Oggetti

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

Regole

Chiavi	Devono essere stringhe univoche con virgolette doppie
Valori	Qualsiasi tipo JSON valido
Ordine	L'ordine delle chiavi non è garantito
Annidamento	Gli oggetti possono contenere altri oggetti

Array

Sintassi degli Array

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

Array a Tipi Misti

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

Regole

Ordinato	Gli elementi mantengono l'ordine di inserzione
Tipi misti	Gli elementi dell'array possono essere di tipi diversi
Indicizzazione	Basata su zero (nella maggior parte dei linguaggi)

Annidamento

Struttura Annidata

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

Pattern di Accesso

obj.user.name	Notazione a punto (JavaScript)
obj["user"]["name"]	Notazione con parentesi
obj.user.scores[0]	Indice array dentro un oggetto annidato

Validazione dello Schema

Esempio di Schema JSON

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

Parole Chiave dello Schema

type	string, number, integer, boolean, object, array, null
required	Array di nomi di proprietà obbligatorie
properties	Definisce le proprietà attese dell'oggetto
enum	Limita a un insieme fisso di valori
minLength / maxLength	Vincoli sulla lunghezza delle stringhe
minimum / maximum	Vincoli sull'intervallo dei numeri

Basi di jq

Filtri Comuni

.	Identità — passa l'input invariato
.key	Accede a una chiave dell'oggetto
.key.nested	Accede a una chiave annidata
.[0]	Primo elemento dell'array
.[]	Itera tutti gli elementi dell'array
select(.age > 20)	Filtra per condizione
map(.name)	Trasforma ogni elemento
length	Lunghezza dell'array o stringa
keys	Chiavi dell'oggetto come array

Esempi 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)'
```

Pattern Comuni

Risposta API

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

File di Configurazione

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

Suggerimenti

Valida	Usa jsonlint o python -m json.tool
Formatta	jq . file.json o python -m json.tool
JSONL	Un oggetto JSON per riga (newline-delimited)
JSON5 / JSONC	Estensioni che ammettono commenti e virgola finale