REFERENCIA RÁPIDA DE JSON

Sintaxis, tipos de datos, objetos, arrays, jq

Sintaxis

Reglas

`{ }`	Objeto (pares clave-valor sin orden)
`[ ]`	Array (lista ordenada de valores)
`"key": value`	Las claves deben ser cadenas entre comillas dobles
`No trailing comma`	El último elemento no debe tener coma
`No comments`	JSON no permite comentarios

Ejemplo mínimo

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

Tipos de datos

Seis tipos de valor

`"string"`	Texto UTF-8 entre comillas dobles
`42 / 3.14`	Número (entero o punto flotante)
`true / false`	Booleano
`null`	Nulo (ausencia de valor)
`{ }`	Objeto
`[ ]`	Array

Secuencias de escape en cadenas

`\"`	Comilla doble
`\\`	Barra invertida
`\n  \t`	Nueva línea, tabulador
`\uXXXX`	Escape Unicode (hexadecimal)

Objetos

Sintaxis de objeto

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

Reglas

`Keys`	Deben ser cadenas únicas entre comillas dobles
`Values`	Cualquier tipo JSON válido
`Order`	El orden de las claves no está garantizado
`Nesting`	Los objetos pueden contener objetos

Arrays

Sintaxis de array

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

Array de tipos mixtos

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

Reglas

`Ordered`	Los elementos mantienen el orden de inserción
`Mixed types`	Los elementos del array pueden ser de distintos tipos
`Indexing`	Base cero (en la mayoría de lenguajes)

Anidamiento

Estructura anidada

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

Patrones de acceso

`obj.user.name`	Notación de punto (JavaScript)
`obj["user"]["name"]`	Notación de corchetes
`obj.user.scores[0]`	Índice de array dentro de objeto anidado

Validación de esquema

Ejemplo de JSON Schema

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

Palabras clave del esquema

`type`	string, number, integer, boolean, object, array, null
`required`	Array de nombres de propiedades obligatorias
`properties`	Define las propiedades esperadas del objeto
`enum`	Restringir a un conjunto fijo de valores
`minLength / maxLength`	Restricciones de longitud de cadena
`minimum / maximum`	Restricciones de rango numérico

Fundamentos de jq

Filtros comunes

`.`	Identidad — pasar la entrada tal cual
`.key`	Acceder a la clave del objeto
`.key.nested`	Acceder a clave anidada
`.[0]`	Primer elemento del array
`.[]`	Iterar todos los elementos del array
`select(.age > 20)`	Filtrar por condición
`map(.name)`	Transformar cada elemento
`length`	Longitud del array o cadena
`keys`	Claves del objeto como array

Ejemplos de 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)'
```

Patrones comunes

Respuesta de API

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

Archivo de configuración

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

Consejos

`Validate`	Usar jsonlint o python -m json.tool
`Pretty print`	jq . file.json o python -m json.tool
`JSONL`	Un objeto JSON por línea (delimitado por salto de línea)
`JSON5 / JSONC`	Extensiones que permiten comentarios y comas finales