

RÉFÉRENCE RAPIDE JSON

Syntaxe, types de données, objets, tableaux, jq

Syntaxe

| | |
|--------------------------|--|
| Règles | |
| { } | Objet (paires clé-valeur non ordonnées) |
| [] | Tableau (liste ordonnée de valeurs) |
| "key": value | Les clés doivent être des chaînes entre guillemets doubles |
| No trailing comma | Le dernier élément ne doit pas avoir de virgule |
| No comments | JSON n'autorise pas les commentaires |

Exemple minimal

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

Types de données

Six types de valeurs

| | |
|---------------------|--------------------------------------|
| "string" | Texte UTF-8 entre guillemets doubles |
| 42 / 3.14 | Nombre (entier ou virgule flottante) |
| true / false | Booléen |
| null | Null (absence de valeur) |
| { } | Objet |
| [] | Tableau |

Séquences d'échappement de chaînes

| | |
|-----------------------|--------------------------------------|
| \" | Guillemet double |
| \\ | Barre oblique inverse |
| \n / \t | Saut de ligne, tabulation |
| \uXXXX | Séquence d'échappement Unicode (hex) |

Objets

Syntaxe des objets

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

Règles

| | |
|----------------|---|
| Keys | Doivent être des chaînes uniques entre guillemets doubles |
| Values | Tout type JSON valide |
| Order | L'ordre des clés n'est pas garanti |
| Nesting | Les objets peuvent contenir des objets |

Tableaux

Syntaxe des tableaux

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

Tableau de types mixtes

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

Règles

| | |
|--------------------|--|
| Ordered | Les éléments conservent l'ordre d'insertion |
| Mixed types | Les éléments d'un tableau peuvent être de types différents |
| Indexing | Base zéro (dans la plupart des langages) |

Imbrication

Structure imbriquée

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

Motifs d'accès

| | |
|----------------------------|---|
| obj.user.name | Notation pointée (JavaScript) |
| obj["user"]["name"] | Notation entre crochets |
| obj.user.scores[0] | Index de tableau dans un objet imbriqué |

Validation de schéma

Exemple de schéma JSON

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

Mots-clés du schéma

| | |
|------------------------------|---|
| type | string, number, integer, boolean, object, array, null |
| required | Tableau des noms de propriétés obligatoires |
| properties | Définit les propriétés attendues de l'objet |
| enum | Restreindre à un ensemble fixe de valeurs |
| minLength / maxLength | Contraintes de longueur de chaîne |
| minimum / maximum | Contraintes de plage numérique |

Bases de jq

Filtres courants

| | |
|-----------------------------|---|
| . | Identité — passer l'entrée telle quelle |
| .key | Accéder à une clé d'objet |
| .key.nested | Accéder à une clé imbriquée |
| [0] | Premier élément du tableau |
| [] | Itérer tous les éléments du tableau |
| select(.age > 20) | Filtrer par condition |

| | |
|-------------------|-------------------------------------|
| map(.name) | Transformer chaque élément |
| length | Longueur du tableau ou de la chaîne |
| keys | Clés de l'objet en tableau |

Exemples jq

```
echo '{"a":1}' | jq '.a'
echo '[1,2,3]' | jq 'map(. * 2)' # [2,4,6]
cat data.json | jq '.users[].name'
cat data.json | jq '.[] | select(.active)'
```

Modèles courants

Réponse API

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

Fichier de configuration

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

Conseils

| | |
|--------------------------|--|
| Valider | Utiliser jsonlint ou python -m json.tool |
| Affichage formaté | jq . file.json ou python -m json.tool |
| JSONL | Un objet JSON par ligne (délimité par saut de ligne) |
| JSON5 / JSONC | Extensions autorisant les commentaires et les virgules finales |