Inhaltsverzeichnis

JSON-Format: Einfach erklärt

Was ist JSON?

JSON (JavaScript Object Notation) ist ein leichtgewichtiges Datenaustauschformat, das für Menschen leicht zu lesen und zu schreiben ist und für Maschinen einfach zu analysieren und zu generieren. JSON ist ein textbasiertes Format und komplett sprachunabhängig, wird aber in vielen Programmiersprachen unterstützt.

Warum JSON verwenden?

- **Einfache Lesbarkeit**: JSON-Daten sind in einem übersichtlichen Format strukturiert
- **Universelle Kompatibilität**: Nahezu alle Programmiersprachen können JSON verarbeiten
- **Geringe Dateigröße**: JSON-Dateien sind kompakt und benötigen wenig Speicherplatz
- **Schnelle Verarbeitung**: Das Format lässt sich effizient einlesen und verarbeiten

Die Grundstruktur von JSON

JSON basiert auf zwei Strukturen:

1. **Objekte**: Eine Sammlung von Name/Wert-Paaren, umschlossen von geschweiften Klammern `{ }`
2. **Arrays**: Eine geordnete Liste von Werten, umschlossen von eckigen Klammern `[ ]`

Unterstützte Datentypen

- **Zahlen**: Ganze Zahlen oder Dezimalzahlen (z.B. `42` oder `3.14`)
- **Zeichenketten**: Text in doppelten Anführungszeichen (z.B. `„Hallo Welt“`)
- **Boolesche Werte**: `true` oder `false`
- **Arrays**: Geordnete Listen, z.B. `[1, 2, 3]`
- **Objekte**: Sammlungen von Name/Wert-Paaren, z.B. `{„name“: „Max“, „alter“: 30}`
- **null**: Ein spezieller Wert, der „nichts“ repräsentiert

Beispiele für JSON

Beispiel 1: Person (Max Mustermann)

Hier ist ein einfaches JSON-Objekt, das die Adressdaten von Max Mustermann darstellt:

```
{
  "vorname": "Max",
  "nachname": "Mustermann",
  "alter": 35,
  "adresse": {
    "strasse": "Musterstraße",
    "hausnummer": "42",
    "plz": "12345",
    "ort": "Musterstadt",
    "land": "Deutschland"
```

```
  },
  "telefon": "0123-456789",
  "email": "max@mustermann.de",
  "aktiv": true
}
```

Beispiel 2: Büchersammlung

Ein komplexeres Beispiel, das eine Büchersammlung als JSON-Array mit mehreren JSON-Objekten darstellt:

```
{
  "sammlung": "Meine Bücher",
  "besitzer": "Max Mustermann",
  "anzahl": 4,
  "buecher": [
    {
      "titel": "Der Herr der Ringe",
      "autor": "J.R.R. Tolkien",
      "erscheinungsjahr": 1954,
      "isbn": "978-3-608-93981-1",
      "gelesen": true,
      "kategorie": ["Fantasy", "Abenteuer"]
    },
    {
      "titel": "Harry Potter und der Stein der Weisen",
      "autor": "J.K. Rowling",
      "erscheinungsjahr": 1997,
      "isbn": "978-3-551-55167-2",
      "gelesen": true,
      "kategorie": ["Fantasy", "Jugendbuch"]
    },
    {
      "titel": "Die unendliche Geschichte",
      "autor": "Michael Ende",
      "erscheinungsjahr": 1979,
      "isbn": "978-3-522-17750-0",
      "gelesen": false,
      "kategorie": ["Fantasy", "Kinderbuch"]
    },
    {
      "titel": "Das Café am Rande der Welt",
      "autor": "John Strelecky",
      "erscheinungsjahr": 2007,
      "isbn": "978-3-423-20969-4",
      "gelesen": true,
      "kategorie": ["Philosophie", "Roman"]
    }
  ],
  "zuletzt_aktualisiert": "2025-04-09"
}
```

Beispiel 3: Einfaches JSON-Array

Ein einfaches JSON-Array mit verschiedenen Datentypen:

```
[
  "Apfel",
  42,
  true,
  null,
  {
    "name": "Beispiel-Objekt",
    "wert": 3.14
  }
]
```

Syntax-Regeln für JSON

1. **Daten werden in Name/Wert-Paaren notiert**: „name“: „wert“
2. **Name/Wert-Paare werden durch Kommas getrennt**: „name“: „wert“, „alter“: 30
3. **Objekte werden in geschweiften Klammern notiert**: { … }
4. **Arrays werden in eckigen Klammern notiert**: […]
5. **Zeichenketten müssen in doppelten Anführungszeichen stehen**
6. **Zahlen können als Ganzzahlen oder Dezimalzahlen dargestellt werden**
7. **Boolesche Werte werden als `true` oder `false` notiert**
8. **Fehlende Werte können als `null` angegeben werden**

Häufige Anwendungsfälle für JSON

- **Datenspeicherung**: Konfigurationsdateien, Benutzereinstellungen
- **API-Kommunikation**: Datenaustausch zwischen Webanwendungen
- **AJAX-Anfragen**: Asynchroner Datenaustausch im Browser
- **Datenbanken**: Speichern strukturierter Daten (z.B. in NoSQL-Datenbanken)
- **Konfigurationsdateien**: Einstellungen für Anwendungen

Tipps und Best Practices

- Verwende stets doppelte Anführungszeichen für Schlüssel und Zeichenketten
- Achte auf korrekte Syntax (Kommas, Klammern)
- Validiere JSON-Daten vor der Verarbeitung
- Nutze Einrückungen für bessere Lesbarkeit
- Vermeide zu komplexe Verschachtelungen

Häufige Fehler

- Komma nach dem letzten Element eines Objekts oder Arrays (in JSON nicht erlaubt)
- Einfache statt doppelte Anführungszeichen
- Fehlende Anführungszeichen bei Schlüsseln
- Unerlaubte Kommentare im JSON-Code (JSON unterstützt keine Kommentare)