JSON and structured data

Structured data has an organised shape

Structured data follows a predictable organisation so programs can locate, filter and transform values. JSON (JavaScript Object Notation) is a text format commonly used to exchange data between systems and applications.

```
{
  "event_id": 42,
  "name": "Robotics showcase",
  "cancelled": false,
  "tags": ["technology", "community"]
}
```

> **Key idea:** JSON carries data, not behaviour. A program decides how to validate, store, display or act on the values.

Objects, arrays, keys and values

Part	Purpose	Notation
Object	Groups named key–value pairs	`{ }`
Array	Stores an ordered collection of values	`[ ]`
Key	Names a value within an object	Double-quoted string followed by `:`

Part	Purpose	Notation
Value	Holds the data	String, number, Boolean, null, object or array

Read the structure

```
{
  "station": "North",
  "readings": [18, 21, 19],
  "active": true
}
```

Recognise the object, its three keys, the array and each value's type.

Check your answer

The outer braces contain one object. Its keys are `station`, `readings` and `active`. `"North"` is a string; `[18, 21, 19]` is an array containing three numbers; and `true` is a Boolean.

JSON value types

- **String:** text inside double quotes, e.g. `"open"`.
- **Number:** an integer or decimal without quotes, e.g. `27` or `3.5`.
- **Boolean:** lowercase `true` or `false`.
- **Null:** lowercase `null`, meaning no value is supplied.
- **Object:** nested named values inside braces.
- **Array:** an ordered collection inside square brackets.

A quoted number such as `"27"` is a string. Its appearance does not change its type.

Choose suitable types

Determine a suitable JSON type for an attendee name, ticket price, checked-in state, optional accessibility note and list of session IDs.

Check your answer

- Name: string
- Ticket price: number
- Checked-in state: Boolean
- Missing optional note: null; otherwise a string
- Session IDs: array, normally containing strings or numbers consistently

Valid JSON syntax

- Keys and string values use double quotes.
- A colon separates a key from its value.
- A comma separates pairs or array items, but no comma follows the final item.
- Braces and brackets must be balanced and correctly nested.
- Comments and `undefined` are not part of standard JSON.

> **Do not confuse a programming-language object with JSON text:** some languages allow single quotes, unquoted keys or trailing commas. Standard JSON does not.

Repair malformed JSON

```
{
  event_id: 7,
  "title": 'Night market',
  "open": True,
  "stalls": [12, 15,],
}
```

Analyse the data and rewrite it as valid JSON.

Check your answer

```
{
  "event_id": 7,
  "title": "Night market",
```

```
  "open": true,
  "stalls": [12, 15]
}
```

The key needed quotes, the string needed double quotes, the Boolean needed lowercase, and both trailing commas were removed.

Nested objects and arrays

```
{
  "venue": {
    "name": "River Hall",
    "address": {"suburb": "West End", "postcode": "4101"}
  },
  "sessions": [
    {"session_id": 1, "places": 0},
    {"session_id": 2, "places": 8}
  ]
}
```

Read a path one container at a time. The postcode path is `venue → address → postcode`. The second session's places path is `sessions → item 2 → places`.

Trace the paths

Determine the suburb, the first session ID and the number of places in the second session.

Check your answer

Suburb: `West End`; first session ID: `1`; second session places: `8`.

Collections of records

An API response often wraps an array of consistently shaped objects.

```
{
  "results": [
    {"id": 101, "suburb": "Nundah", "status": "open"},
    {"id": 102, "suburb": "Nundah", "status": "closed"},
    {"id": 103, "suburb": "Milton", "status": "open"}
  ],
  "result_count": 3
}
```

Consistency matters. The same concept should normally use the same key name and compatible type in every record.

Filter the records

Develop language-independent pseudocode that outputs the ID of each open Nundah record.

View a sample solution

```
FOR EACH record IN data["results"] DO
    IF record["suburb"] = "Nundah" AND record["status"] = "open" THEN
        OUTPUT record["id"]
    ENDIF
ENDFOR
```

The output is `101`.

Parse, validate, transform and serialise

1. **Receive:** obtain JSON text from a file, message or API.
2. **Parse:** convert the text into an internal object or collection.
3. **Validate:** check required keys, types, formats and allowed values.
4. **Transform:** filter, calculate, rename, group or combine data.
5. **Serialise:** convert internal data back into JSON text for exchange.

Valid syntax does not guarantee valid data. `{"places": -9}` is syntactically valid but may break a rule that places cannot be negative.

Validate before use

A booking requires `booking_id` as a positive integer, `email` as a non-empty string and `confirmed` as a Boolean.

```
{"booking_id": "24", "email": "", "confirmed": "false"}
```

Analyse the record against the rules.

Check your answer

- `booking_id` is a string rather than an integer.
- `email` is present but empty.
- `confirmed` is a string rather than the Boolean `false`.

Design data for exchange

Choose keys that are meaningful, consistent and unambiguous. Avoid mixing display labels with stable identifiers.

- Use `student_id` to identify a student reliably, even if their displayed name changes.
- Choose one naming convention, such as `snake_case`, and apply it consistently.
- Represent repeated items as an array rather than numbered keys such as `item1` and `item2`.
- Do not include sensitive fields unless the receiver needs them.
- Document units and date formats in the data specification.

Explain a design choice

Explain why an array is more suitable than the keys `tag1`, `tag2` and `tag3` for storing an unknown number of tags.

Check your answer

An array can contain zero, one or many tags under one predictable key. A program can iterate through it without guessing which numbered keys exist, and the structure does not need a new field when another tag is added.

Independent practice: community workshops

```
{
  "generated_at": "2026-08-05T09:30:00+10:00",
  "workshops": [
    {
      "workshop_id": 11,
      "title": "Bike repair",
      "venue": {"name": "Shed A", "accessible": true},
      "capacity": 16,
      "booked": 16,
      "tags": ["practical", "transport"]
    },
    {
      "workshop_id": 12,
      "title": "Native gardens",
      "venue": {"name": "Courtyard", "accessible": false},
      "capacity": 20,
      "booked": 13,
      "tags": ["outdoors"]
    }
  ]
}
```

1. **Describe** the outer structure and the structure of one workshop.
2. **Determine** the path to the second workshop's accessibility value.
3. **Develop** pseudocode that outputs the titles of accessible workshops with at least one place remaining.

4. **Explain** why `workshop_id` should be retained when sending a selected workshop back to the server.

View a sample solution

1. The outer object has a timestamp and a `workshops` array. Each workshop is an object containing scalar values, a nested venue object and a tags array.
2. `workshops → item 2 → venue → accessible`, which is `false`.
3. ```
   FOR EACH workshop IN data["workshops"] DO
       SET placesRemaining = workshop["capacity"] - workshop["booked"]
       IF workshop["venue"]["accessible"] = TRUE AND placesRemaining > 0 THEN
           OUTPUT workshop["title"]
       ENDIF
   ENDFOR
   ```
   No title is output: the accessible workshop is full.
4. The stable identifier lets the server locate the exact record even if workshop titles are duplicated or later changed.

Revision #5
Created 2026-10-06 00:40:26 UTC by Mr Napper
Updated 2026-10-06 01:32:07 UTC by Mr Napper
   ```