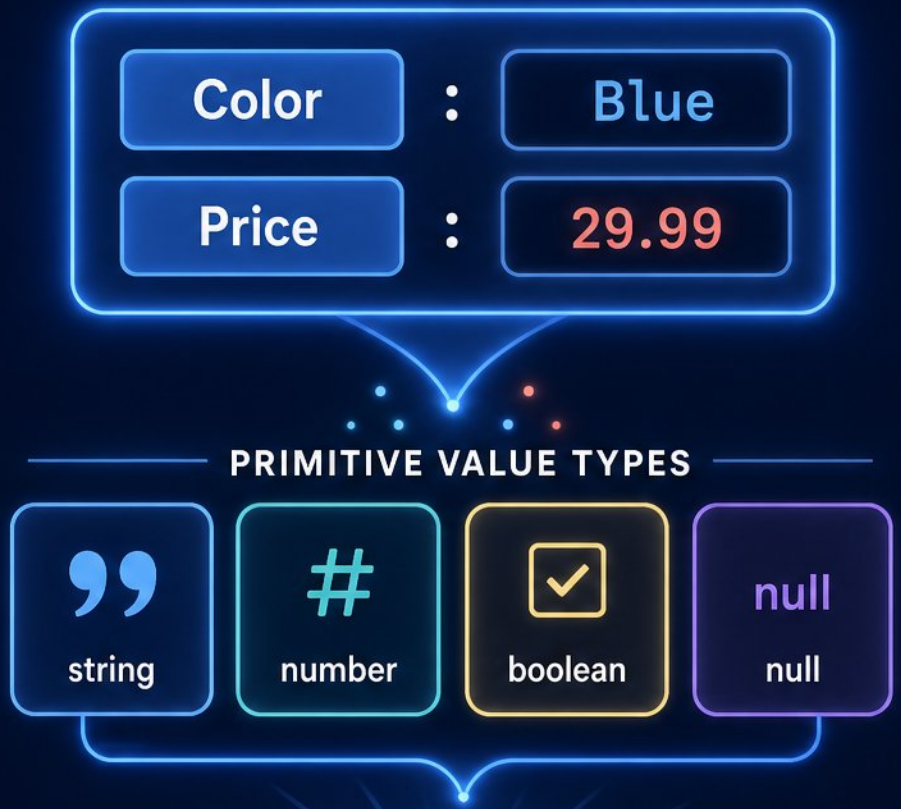
Understanding Structured Data: Objects, Arrays, and JSON

- Structured data uses field–value pairs for reliable machine-to-machine communication
- ~97% of API requests use JSON, making it the universal interchange language
- Structured (spreadsheet) vs. unstructured (social post): predictable format enables automation
- Core building blocks: fields, objects, arrays, and JSON syntax
- Goal: read, reason about, and construct simple structured data documents

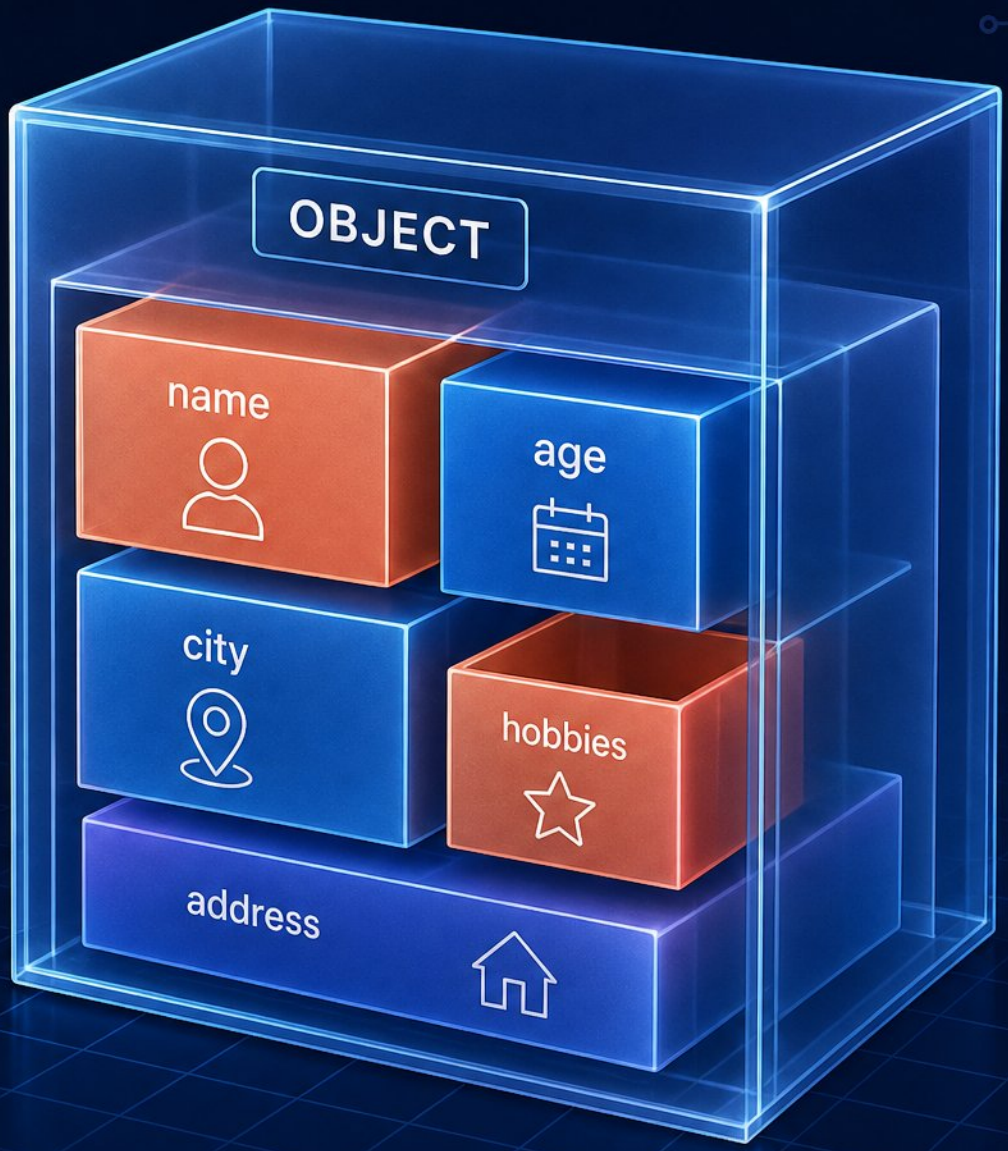
Fields and Values:
The Smallest Building Blocks



- Field–value pairs describe anything: 'Color: Blue', 'Price: 29.99'
- Primitive value types: strings (text), numbers, booleans (true/false), null
- Values can also be complex—objects or arrays—creating nested structures
- Field names are always strings; each must be unique within its immediate container

Organizing Data with Objects

- An object is an unordered collection of named fields
- Field names are unique; order does not matter
- Group related properties to model a single entity
- Example: a person object with name, age, and city fields
- Objects can nest other objects or arrays as values



Grouping Items with Arrays

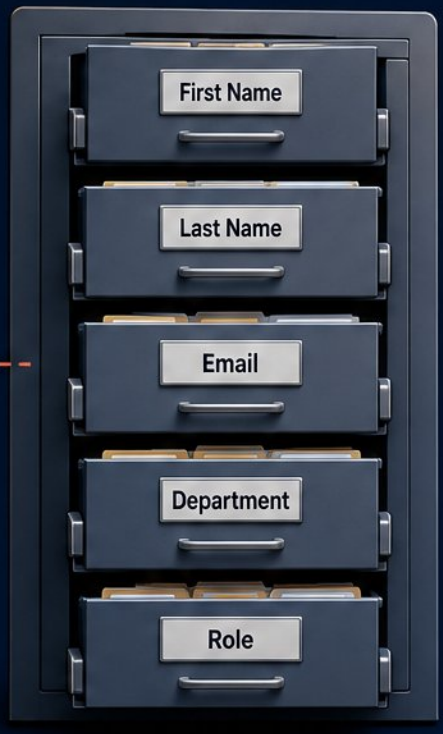
- - Ordered list of values accessed by zero-based numeric index
- - Use arrays when sequence matters (to-do list, top scores)
- - Values can be strings, numbers, objects, or nested arrays
- - Array of objects stores multi-item records like employee lists
- - Contrast: objects use named fields; arrays use positions

Arrays
(ordered positions)

[ ] Values are stored in order and accessed by their zero-based index.

VS

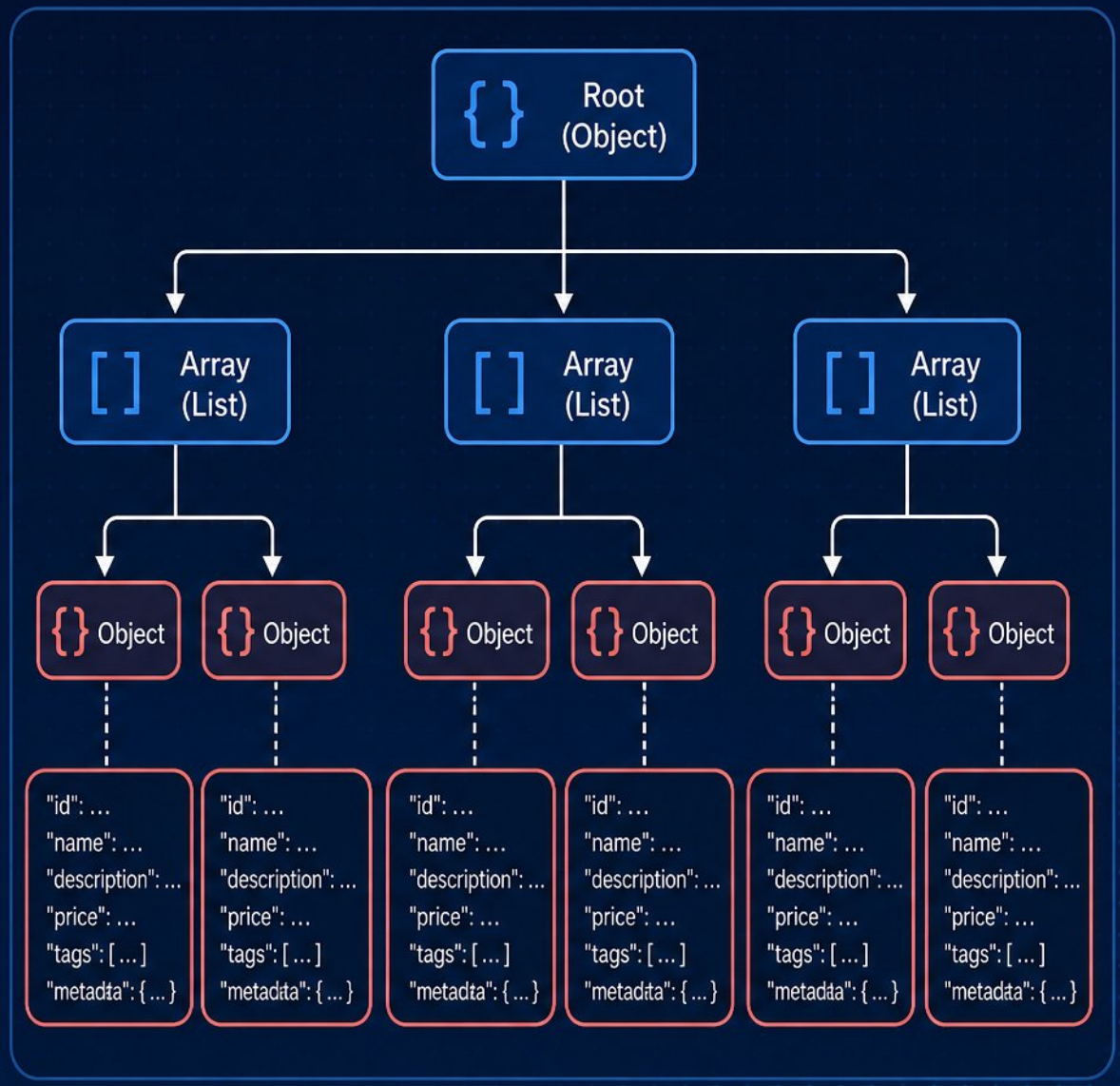
Objects
(named fields)



{ } Values are stored in named fields and accessed by those names.

Combining Objects and Arrays to Model Real Information

- Model a product catalog as an array of product objects
- Trace parent-child relationships in nested structures like a tree
- Choose objects for named properties, arrays for ordered lists
- Every JSON document has a single root: object, array, or value

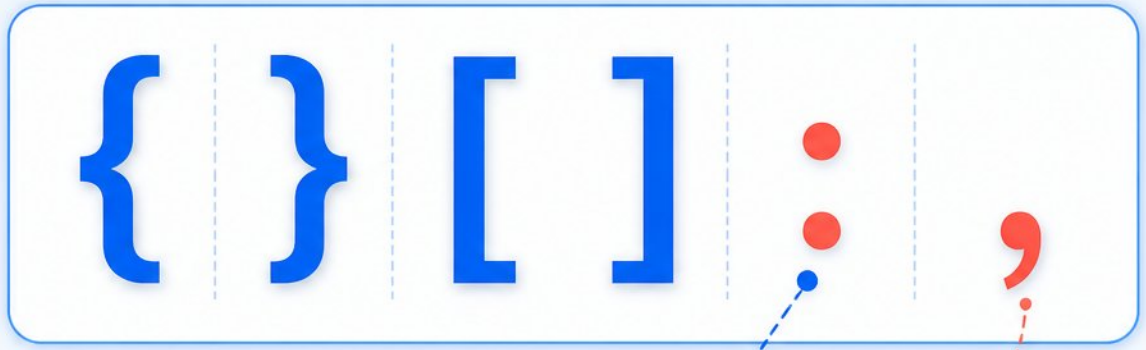


Introducing JSON as the Universal Interchange Format

- JSON: lightweight, text-based, language-independent data format from object/array structures
- ~97% of API requests use JSON—the dominant web interchange format
- Governed by ECMA-404 and RFC 8259 for consistent syntax across all platforms
- JSON is a text format, not a programming language—despite JavaScript origins
- 52.8% of websites now use JSON-LD for structured data and SEO

JSON Syntax: The Essential Rules

- JSON uses six structural tokens: { } [ ] : ,
- An object is a pair of curly braces around name/value pairs
- An array is a pair of square brackets around ordered values
- Field names must be double-quoted strings; strings use double quotes only
- No trailing commas are allowed in JSON



Field names must be double-quoted strings; strings use double quotes only

No trailing commas are allowed in JSON

Common JSON Pitfalls and How to Avoid Them

- Trailing commas after last item cause errors
- Field names must use double-quoted strings
- Single quotes for strings are not allowed
- Comments break JSON—remove or move documentation externally
- Strict syntax ensures universal, reliable data parsing

VALID JSON

```
{
  "name": "Project",
  "type": "data",
  "active": true,
  "tags": [
    "json",
    "syntax",
    "reliable"
  ],
  "config": {
    "retry": true,
    "timeout": null
  }
}
```

VS

INVALID JSON

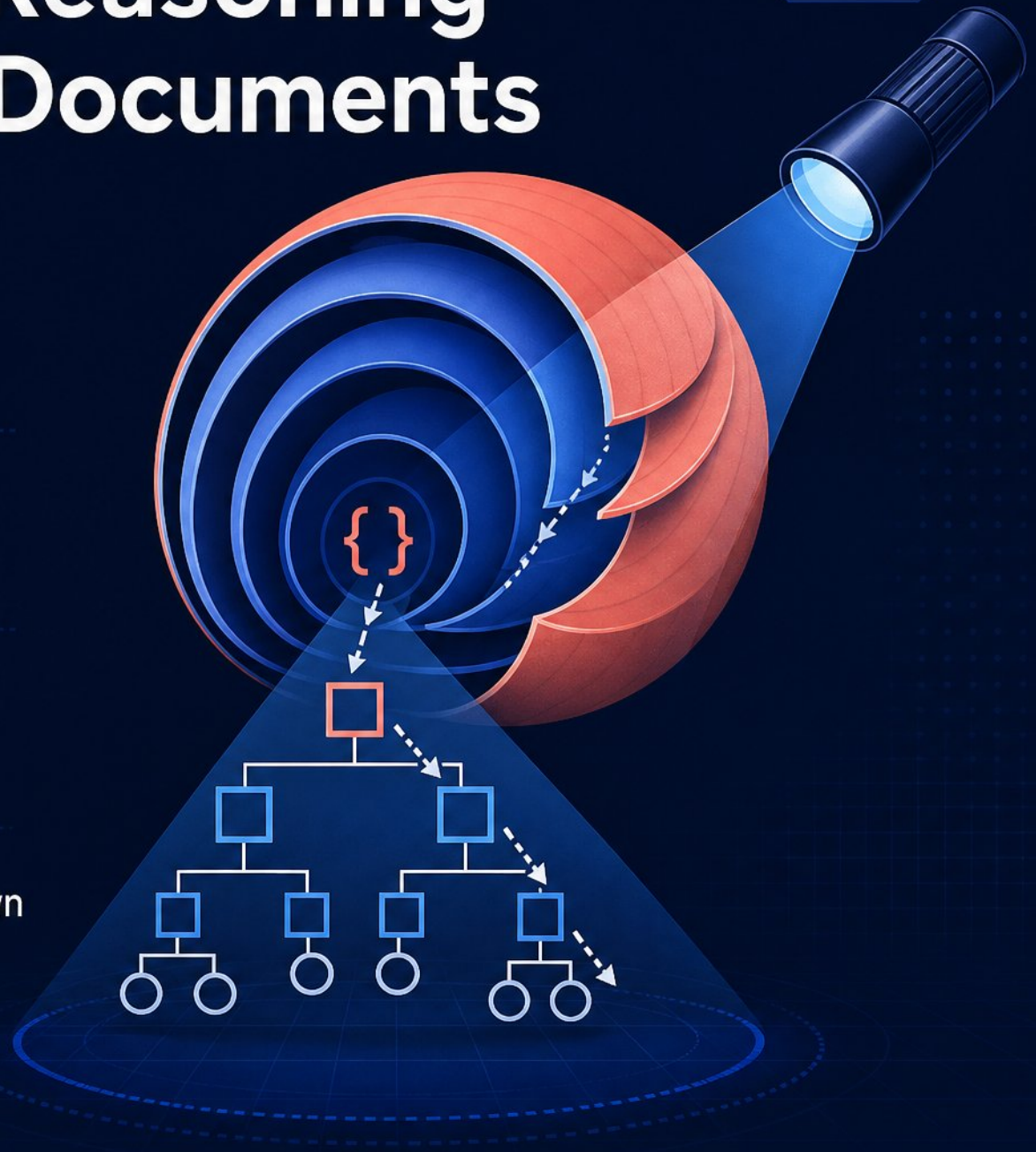
```
{
  'name': 'Project',
  'type': 'data',
  'active": true,
  'tags': [
    'json',
    'syntax",
    'reliable',
  ],
  // this is a comment
  'config": {
    'retry': true,
    'timeout": null,
  }
}
```

Validating JSON: Tools and Techniques

- Use free online tools like JSONLint, JSON Formatter, and JSON Editor Online
- Paste your code to spot errors with exact line numbers
- Formatters beautify minified JSON into readable, indented structures
- Browser DevTools include built-in JSON viewers for API responses

Reading and Reasoning About JSON Documents

- Identify the root, then traverse layer by layer
- Walk through weather API or product listing payloads
- Use JSON tree viewers to explore hierarchy visually
- Read inside-out from known values up through layers



Where JSON Fits in the Real World

- ~97% of API requests use JSON, making it the dominant interchange format
- Common uses: REST API responses, config files (package.json), NoSQL databases
- Web apps, mobile sync, IoT telemetry, and reporting systems rely on JSON daily
- JSON-LD powers structured data on 52.8% of websites for SEO and AI visibility
- JSON Lines enables streaming logs and event-driven architectures

JSON Alternatives and When They Matter

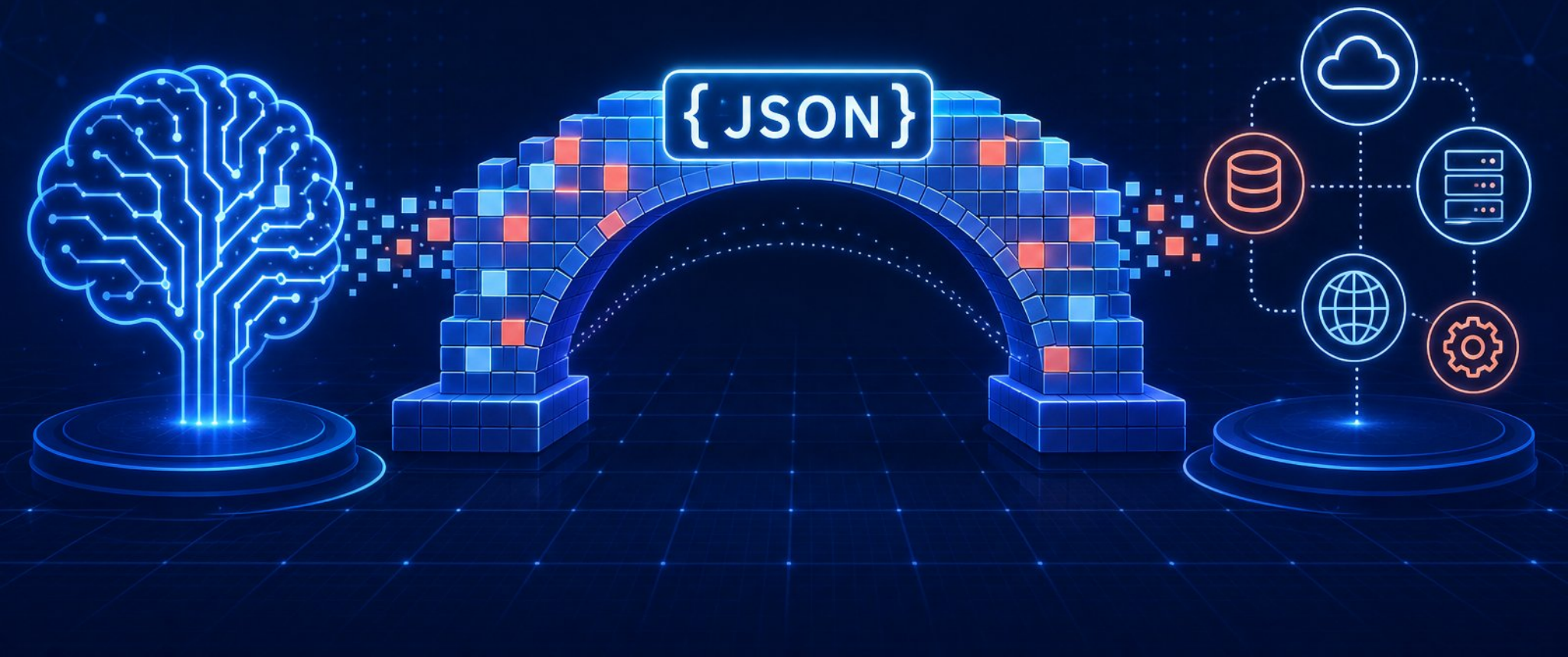
- YAML for human-authored configs, Protobuf for high-performance internal services
- XML still powers banking, government, and legacy enterprise systems
- JSON remains the default: ~95% of new web APIs use it for simplicity and speed
- JSON Schema dominates 76% of modern OpenAPI specs, defining and validating API contracts
- Understanding JSON is the prerequisite to mastering any other structured data format

Practice: Working with Real JSON APIs

- - Explore public APIs: JSONPlaceholder, DummyJSON, and restful-api.dev
- - Fetch JSON with a browser or a simple fetch() call
- - Start with GET /posts, parse the structure, then try POST
- - No registration or API key needed—built for learning
- - Inspect responses in browser DevTools or a JSON viewer

Summary and Your Structured Data Journey

- – Fields → Objects/Arrays → serialized as JSON text documents
- – Structured data is composable, exchangeable, bridges any platform
- – Explore live JSON from public APIs; model your own data structures
- – Next step: convert a contact list into a valid JSON document and validate



PersonWise.

PersonWise • AI digital-human course platform

Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/b42731f0/public