Introduction to JSON

JavaScript Object Notation for Data Exchange

What is JSON?

- **JSON = JavaScript Object Notation**
  - Lightweight data interchange format
  - Human-readable text format
  - Language-independent (despite the name!)
- Originally derived from JavaScript, but now universal

Recipe Card vs. Recipe Database

Traditional Recipe Card

- Handwritten, unique format
- Hard to share digitally
- Difficult for computers to understand

JSON Recipe Format

- Structured, consistent format
- Easy to share between applications
- Both humans and computers can read it

```
{
  "name": "Chocolate Chip Cookies",
  "servings": 24,
  "ingredients": ["flour", "sugar", "chocolate chips"]
}
```

Why Use JSON?

Lightweight and Fast

- Minimal syntax overhead
- Quick to parse and generate
- Small file sizes

Universal Language Support

- Every modern programming language can read JSON
- Web APIs use JSON as a standard
- Mobile apps, databases, configuration files

Human-Readable

- Easy to debug and understand
- Simple syntax rules
- No complex markup like XML

Web Standard

- Native JavaScript support
- RESTful API standard format
- Real-time data exchange (WebSockets, AJAX)

Basic JSON Syntax

- **Six Data Types Only:**
  - String: `"Hello World"`
  - Number: `42` or `3.14`
  - Boolean: `true` or `false`
  - null: `null`
  - Object: `{}`
  - Array: `[]`

JSON Rules

- **Strings must use double quotes** `"` (not single `'`)
- **No trailing commas** in objects or arrays
- **No comments allowed** (pure data format)
- **Keys must be strings** in objects
- **Case-sensitive** (`"Name"` ≠ `"name"`)

JSON Objects

- Objects are key-value pairs in curly braces

```
{
  "firstName": "John",
  "lastName": "Doe",
  "age": 25,
  "isStudent": true,
  "address": null
}
```

Real-World Example: User Profile

```
{
  "username": "jane_smith",
  "email": "jane@university.edu",
  "profile": {
    "fullName": "Jane Smith",
    "department": "Computer Science",
    "yearLevel": 3,
    "gpa": 3.85
  },
  "isActive": true,
  "lastLogin": "2024-03-15T10:30:00Z"
}
```

JSON Arrays

- Arrays are ordered lists in square brackets

```
{
  "courses": [
    "Data Structures",
    "Web Development",
    "Database Systems"
  ],
  "grades": [95, 87, 92, 88],
  "semesters": [
    {"year": 2023, "term": "Fall"},
    {"year": 2024, "term": "Spring"}
  ]
}
```

Nested JSON Structures

```
{
  "university": "Tech University",
  "students": [
    {
      "id": 1001,
      "name": "Alice Johnson",
      "courses": [
        {
          "code": "CS101",
          "title": "Programming Fundamentals",
          "credits": 3,
          "assignments": [
            {"name": "Homework 1", "score": 95},
            {"name": "Midterm", "score": 87}
          ]
        }
      ]
    }
  ]
}
```

JSON vs Other Formats

Feature	JSON	XML	YAML	CSV
Readability	✅ Good	⚠️ Verbose	✅ Excellent	✅ Simple
Web APIs	✅ Standard	⚠️ Legacy	❌ Rare	❌ Limited
Data Types	✅ 6 Types	⚠️ Text Only	✅ Rich Types	❌ Text Only
File Size	✅ Small	❌ Large	✅ Small	✅ Smallest
Comments	❌ No	✅ Yes	✅ Yes	❌ No

Common JSON Use Cases

- **Web APIs and REST Services**

```
{
  "status": "success",
  "data": {
    "weather": "sunny",
    "temperature": 22
  }
}
```

- **Configuration Files**

```
{
  "database": {
    "host": "localhost",
    "port": 5432,
    "name": "university_db"
  },
  "logging": {
    "level": "info",
    "file": "/var/log/app.log"
  }
}
```

- **Data Storage and Exchange**

```
{
  "products": [
    {
      "id": "BOOK001",
      "title": "Introduction to Algorithms",
      "price": 89.99,
      "inStock": true,
      "categories": ["textbook", "computer-science"]
    }
  ]
}
```

Working with JSON in Programming

JavaScript (Native Support)

```
// Parse JSON string to object
const jsonString = '{"name": "John", "age": 25}';
const user = JSON.parse(jsonString);

// Convert object to JSON string
const obj = {name: "Jane", age: 23};
const jsonOutput = JSON.stringify(obj);
```

Python

```python
import json

# Parse JSON string
json_string = '{"name": "John", "age": 25}'
user = json.loads(json_string)

# Convert to JSON string
data = {"name": "Jane", "age": 23}
json_output = json.dumps(data, indent=2)
```

Java

```java
// Using popular libraries like Jackson or Gson
import com.fasterxml.jackson.databind.ObjectMapper;

ObjectMapper mapper = new ObjectMapper();

// Parse JSON
String jsonString = "{\"name\":\"John\",\"age\":25}";
User user = mapper.readValue(jsonString, User.class);

// Convert to JSON
User user = new User("Jane", 23);
String json = mapper.writeValueAsString(user);
```

JSON Validation and Tools

- **Online JSON Validators**
  - jsonlint.com
  - jsonformatter.curiousconcept.com
- **IDE Extensions**
  - VS Code: Built-in JSON support
  - IntelliJ: JSON schema validation

Common JSON Errors

- **Trailing Commas** ❌

```
{"name": "John", "age": 25,}
```

- **Single Quotes** ❌

```
{'name': 'John', 'age': 25}
```

- **Missing Quotes on Keys** ❌

```
{name: "John", age: 25}
```

Correct JSON ✅

```
{
  "name": "John",
  "age": 25,
  "hobbies": ["reading", "coding"],
  "isStudent": true,
  "address": null
}
```

Best Practices for JSON

- **Use meaningful key names**

```
// Good
{"firstName": "John", "lastName": "Doe"}

// Avoid
{"fn": "John", "ln": "Doe"}
```

- **Keep consistent naming conventions**

```
// Consistent camelCase
{
  "firstName": "John",
  "lastName": "Doe",
  "phoneNumber": "123-456-7890"
}
```

- **Use appropriate data types**

```
// Good
{
  "age": 25,
  "isActive": true,
  "salary": null
}

// Avoid
{
  "age": "25",
  "isActive": "true",
  "salary": "null"
}
```

- **Structure nested data logically**

```
{
  "student": {
    "personalInfo": {
      "name": "John Doe",
      "email": "john@university.edu"
    },
    "academic": {
      "major": "Computer Science",
      "gpa": 3.7,
      "courses": ["CS101", "MATH201"]
    }
  }
}
```

Real-World Project Example

University Course API Response


```
{
  "semester": "Fall 2024",
  "courses": [
    {
      "courseId": "CS341",
      "title": "Software Engineering",
      "instructor": {
        "name": "Dr. Smith",
        "email": "smith@university.edu",
        "office": "Engineering 301"
      },
      "schedule": {
        "days": ["Monday", "Wednesday", "Friday"],
        "time": "10:00-11:00",
        "room": "ENG-205"
      },
      "enrollment": {
        "capacity": 30,
        "enrolled": 28,
        "waitlist": 5
      }
    }
```

JSON Schema (Advanced)

- Validation and Documentation

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "name": {
      "type": "string",
      "minLength": 1
    },
    "age": {
      "type": "integer",
      "minimum": 0,
      "maximum": 150
    }
  },
  "required": ["name", "age"]
}
```

Practice Exercise

Create a JSON file representing your daily schedule:

- Include courses with times and locations
- Add your contact information
- Include your academic information
- Use proper nesting and data types

Summary

- **JSON is the standard for data exchange**
  - Lightweight, readable, universal
  - Perfect for web APIs and configuration
  - Simple syntax with strict rules
- **Remember the key rules:**
  - Double quotes for strings and keys
  - No trailing commas
  - Six data types only

Resources

- **Official Documentation**
  - JSON.org - Official specification
  - MDN JSON Guide
- **Tools and Validators**
  - JSONLint - Validation
  - JSON Formatter - Pretty printing

Questions?

Ready to structure your data with JSON!