JSON Tutorial for Beginners

What is JSON?

JSON (JavaScript Object Notation) is a lightweight data format that's easy for humans to read and write, and easy for machines to parse and generate. It's become the standard for exchanging data between web servers and applications.

Basic JSON Structure

JSON data is built on two main structures:

1. **Objects**: Collections of key-value pairs enclosed in curly braces {}
2. **Arrays**: Ordered lists of values enclosed in square brackets []

JSON Data Types

JSON supports these data types:

- **String**: Text enclosed in double quotes ("hello")
- **Number**: Integer or decimal (42, 3.14)
- **Boolean**: true or false
- **Null**: null
- **Object**: { "key": "value" }
- **Array**: ["value1", "value2"]

Writing JSON

Let's look at some examples of properly formatted JSON:

Simple Object

```
{
  "name": "Alice",
  "age": 28,
  "isStudent": false
}
```

Nested Objects

```
{
  "person": {
    "firstName": "John",
    "lastName": "Doe",
    "contact": {
      "email": "john@example.com",
      "phone": "555-1234"
    }
  }
}
```

Arrays

```
{
  "fruits": ["apple", "banana", "orange"],
  "numbers": [1, 2, 3, 4, 5]
}
```

Complex Example

```
{
  "company": "Tech Corp",
  "employees": [
    {
      "id": 1,
      "name": "Sarah Chen",
      "department": "Engineering",
      "skills": ["Python", "JavaScript", "SQL"],
      "active": true
    },
    {
      "id": 2,
      "name": "Michael Rodriguez",
      "department": "Design",
      "skills": ["Figma", "Photoshop"],
      "active": true
    }
  ],
  "founded": 2015,
  "headquarters": null
}
```

Reading JSON

When you read JSON, you're typically receiving it as a text string from a file, API, or database. Here's what JSON looks like as a string:

```
const jsonString = '{"name":"Alice","age":28,"city":"Boston"}';
```

Parsing JSON (Converting String to Object)

Parsing means converting a JSON string into a usable data structure in your programming language.

JavaScript Example

```
// JSON string
const jsonString = '{"name":"Alice","age":28,"hobbies":
["reading","cycling"]}';

// Parse the string into a JavaScript object
const person = JSON.parse(jsonString);

// Now you can access the data
console.log(person.name);        // Output: Alice
console.log(person.age);         // Output: 28
console.log(person.hobbies[0]);  // Output: reading
```

Python Example

```
import json

# JSON string
json_string = '{"name":"Alice","age":28,"hobbies":["reading","cycling"]}'

# Parse the string into a Python dictionary
person = json.loads(json_string)

# Access the data
print(person["name"])        # Output: Alice
print(person["age"])         # Output: 28
print(person["hobbies"][0])  # Output: reading
```

Stringifying JSON (Converting Object to String)

Stringifying is the opposite of parsing—it converts a data structure into a JSON string.

JavaScript Example

```
// JavaScript object
const person = {
  name: "Bob",
  age: 35,
  isEmployed: true,
  skills: ["JavaScript", "Python", "Go"]
};

// Convert to JSON string
const jsonString = JSON.stringify(person);

console.log(jsonString);
// Output: {"name":"Bob","age":35,"isEmployed":true,"skills":
["JavaScript","Python","Go"]}

// Pretty print with indentation
const prettyJson = JSON.stringify(person, null, 2);
console.log(prettyJson);
/* Output:
{
  "name": "Bob",
  "age": 35,
  "isEmployed": true,
  "skills": [
    "JavaScript",
    "Python",
    "Go"
  ]
}
*/
```

Python Example

```
import json

# Python dictionary
person = {
    "name": "Bob",
    "age": 35,
    "isEmployed": True,
    "skills": ["JavaScript", "Python", "Go"]
}

# Convert to JSON string
json_string = json.dumps(person)

print(json_string)
# Output: {"name": "Bob", "age": 35, "isEmployed": true, "skills":
["JavaScript", "Python", "Go"]}

# Pretty print with indentation
pretty_json = json.dumps(person, indent=2)
print(pretty_json)
```

Common JSON Operations

Reading from a File

JavaScript (Node.js):

```
const fs = require('fs');

// Read and parse JSON file
const data = JSON.parse(fs.readFileSync('data.json', 'utf8'));
console.log(data);
```

Python:

```
import json

# Read and parse JSON file
with open('data.json', 'r') as file:
    data = json.load(file)
    print(data)
```

Writing to a File:

JavaScript (Node.js):

```
const fs = require('fs');

const data = {
  name: "Alice",
  age: 28,
  hobbies: ["reading", "cycling"]
};

// Write object to JSON file
fs.writeFileSync('output.json', JSON.stringify(data, null, 2));
```

Python:

```
import json

data = {
    "name": "Alice",
    "age": 28,
    "hobbies": ["reading", "cycling"]
}

# Write dictionary to JSON file
with open('output.json', 'w') as file:
    json.dump(data, file, indent=2)
```

Common Mistakes to Avoid

1. **Using single quotes**: JSON requires double quotes for strings
   - ❌ {'name': 'Alice'}
   - ✅ {"name": "Alice"}
2. **Trailing commas**: JSON doesn't allow commas after the last item
   - ❌ {"name": "Alice", "age": 28,}
   - ✅ {"name": "Alice", "age": 28}
3. **Unquoted keys**: All keys must be strings in quotes
   - ❌ {name: "Alice"}
   - ✅ {"name": "Alice"}
4. **Comments**: JSON doesn't support comments
   - ❌ {"name": "Alice" /* this is a comment */}
5. **Undefined values**: Use null instead
   - ❌ {"value": undefined}
   - ✅ {"value": null}

Practice Exercise

Try parsing this JSON and accessing different values:

```
{
  "store": "BookWorld",
  "books": [
    {
      "title": "The Great Gatsby",
      "author": "F. Scott Fitzgerald",
      "year": 1925,
      "inStock": true
    },
    {
      "title": "1984",
      "author": "George Orwell",
      "year": 1949,
      "inStock": false
    }
  ],
  "location": {
    "city": "New York",
    "state": "NY"
  }
}
```

Tasks:

1. Parse this JSON string
2. Access the store name
3. Get the title of the first book
4. Find out which book is not in stock
5. Get the city location

6. Create a new book object and stringify it

Summary

- **JSON** is a text format for storing and exchanging data
- **Parsing** converts JSON strings to usable objects/dictionaries
- **Stringifying** converts objects/dictionaries to JSON strings
- JSON uses double quotes, no trailing commas, and supports limited data types
- Most programming languages have built-in JSON support

Now you're ready to work with JSON in your projects!