JSON

# JSON fundamental

## What is JSON

JSON is an acronym for JavaScript Object Notation, is an open standard format, which is lightweight and text-based, designed explicitly for human-readable data interchange. It is a language-independent data format. It supports almost every kind of language, framework, and library.

In the early 2000s, JSON was initially specified by Douglas Crockford. In 2013, JSON was standardized as ECMA-404, and RCF 8259 was published in 2017.

JSON is an open standard for exchanging data on the web. It supports data structures like objects and arrays. So, it is easy to write and read data from JSON.

In JSON, data is represented in key-value pairs, and curly braces hold objects, where a colon is followed after each name. The comma is used to separate key-value pairs. Square brackets are used to hold arrays, where each value is comma-separated.

## What is JSON

* JSON stands for JavaScript Object Notation.
* JSON is an open standard data-interchange format.
* JSON is lightweight and self-describing.
* JSON originated from JavaScript.
* JSON is easy to read and write.
* JSON is language independent.
* JSON supports data structures such as arrays and objects.

## Why do we use JSON?

Since JSON is an easy-to-use, lightweight language data interchange format in comparison to other available options, it can be used for API integration. Following are the advantages of JSON:

* **Less Verbose**: In contrast to XML, JSON follows a compact style to improve its users' readability. While working with a complex system, JSON tends to make substantial enhancements.
* **Faster**: The JSON parsing process is faster than that of the XML because the DOM manipulation library in XML requires extra memory for handling large XML files. However, JSON requires less data that ultimately results in reducing the cost and increasing the parsing speed.
* **Readable**: The JSON structure is easily readable and straightforward. Regardless of the programming language that you are using, you can easily map the domain objects.
* **Structured Data**: In JSON, a map data structure is used, whereas XML follows a tree structure. The key-value pairs limit the task but facilitate the predictive and easily understandable model.

## JSON Data Types

Following are the most commonly used JSON data types.

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Description** | **Example** |
| String | A string is always written in double-quotes. It may consist of numbers, alphanumeric and special characters. | "student", "name", "1234", "Ver\_1" |
| Number | Number represents the numeric characters. | 121, 899 |
| Boolean | It can be either True or False. | true |
| Null | It is an empty value. |  |

## JSON Objects

In JSON, objects refer to dictionaries, which are enclosed in curly brackets, i.e., { }. These objects are written in key/value pairs, where the key has to be a string and values have to be a valid JSON data type such as string, number, object, Boolean or null. Here the key and values are separated by a colon, and a comma separates each key/value pair.

For example:

**{"name" : "Jack", "employeeid" : 001, "present" : false}**

## JSON Arrays

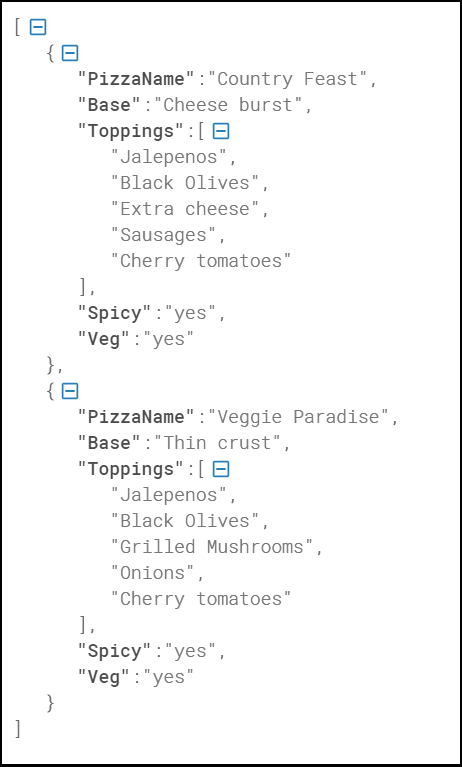
In JSON, arrays can be understood as a list of objects, which are mainly enclosed in square brackets [ ]. An array value can be a string, number, object, array, boolean or null.

For example:

1. *[{*
2. *"PizzaName" : "Country Feast",*
3. *"Base" : "Cheese burst",*
4. *"Toppings" : ["Jalepenos", "Black Olives", "Extra cheese", "Sausages", "Cherry tomatoes"],*
5. *"Spicy" : "yes",*
6. *"Veg" : "yes"*
7. *},*
9. *{*
10. *"PizzaName" : "Veggie Paradise",*
11. *"Base" : "Thin crust",*
12. *"Toppings" : ["Jalepenos", "Black Olives", "Grilled Mushrooms", "Onions", "Cherry tomatoes"],*
13. *"Spicy" : "yes",*
14. *"Veg" : "yes"*
15. *}*
16. *]*

In the above example, the object "Pizza" is an array. It contains five objects, i.e., PizzaName, Base, Toppings, Spicy, and Veg.

Output:



# JSON Example

JSON example can be created by object and array. Each object can have different data such as text, number, boolean etc. Let's see different JSON examples using object and array.

### JSON Object Example

A JSON object contains data in the form of key/value pair. The keys are strings and the values are the JSON types. Keys and values are separated by colon. Each entry (key/value pair) is separated by comma.

The **{** (curly brace) represents the JSON object.

1. *{*
2. *"employee": {*
3. *"name":       "sonoo",*
4. *"salary":      56000,*
5. *"married":****true***
6. *}*
7. *}*

### JSON Array example

The **[** (square bracket) represents the JSON array. A JSON array can have values and objects.

Let's see the example of JSON array having values.

*["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]*

Let's see the example of JSON array having objects.

1. **[**
2. **{"name":"Ram", "email":"Ram@gmail.com"},**
3. **{"name":"Bob", "email":"bob32@gmail.com"}**
4. **]**

## JSON Example 1

1. **{"employees":[**
2. **{"name":"Shyam", "email":"shyamjaiswal@gmail.com"},**
3. **{"name":"Bob", "email":"bob32@gmail.com"},**
4. **{"name":"Jai", "email":"jai87@gmail.com"}**
5. **]}**

## JSON Example 2

1. **{"menu": {**
2. **"id": "file",**
3. **"value": "File",**
4. **"popup": {**
5. **"menuitem": [**
6. **{"value": "New", "onclick": "CreateDoc()"},**
7. **{"value": "Open", "onclick": "OpenDoc()"},**
8. **{"value": "Save", "onclick": "SaveDoc()"}**
9. **]**
10. **}**
11. **}}**

JSON Object

JSON object holds key/value pair. Each key is represented as a string in JSON and value can be of any type. The keys and values are separated by colon. Each key/value pair is separated by comma.

The curly brace **{** represents JSON object.

Let's see an example of JSON object.

1. ***{***
2. ***"employee": {***
3. ***"name":       "sonoo",***
4. ***"salary":      56000,***
5. ***"married":    true***
6. ***}***
7. ***}***

In the above example, employee is an object in which "name", "salary" and "married" are the key. In this example, there are string, number and boolean value for the keys.

## JSON Nested Object Example

A JSON object can have another object also. Let's see a simple example of JSON object having another object.

1. ***{***
2. ***"firstName": "Sonoo",***
3. ***"lastName": "Jaiswal",***
4. ***"age": 27,***
5. ***"address" : {***
6. ***"streetAddress": "Plot-6, Mohan Nagar",***
7. ***"city": "Ghaziabad",***
8. ***"state": "UP",***
9. ***"postalCode": "201007"***
10. ***}***
11. ***}***

# JSON Array

JSON array represents ordered list of values. JSON array can store multiple values. It can store string, number, boolean or object in JSON array.

In JSON array, values must be separated by comma.

The **[** (square bracket) represents JSON array.

## JSON Array of Strings

Let's see an example of JSON arrays storing string values.

1. *["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]*

## JSON Array of Objects

Let's see a simple JSON array example having 4 objects.

1. ***{"employees":[***
2. ***{"name":"Ram", "email":"ram@gmail.com", "age":23},***
3. ***{"name":"Shyam", "email":"shyam23@gmail.com", "age":28},***
4. ***{"name":"John", "email":"john@gmail.com", "age":33},***
5. ***{"name":"Bob", "email":"bob32@gmail.com", "age":41}***
6. ***]}***

JSON Comments

JSON doesn't support comments. It is not a standard.

But you can do some tricks such as adding extra attribute for comment in JSON object, for example:

1. ***{***
2. ***"employee": {***
3. ***"name":       "Bob",***
4. ***"salary":      56000,***
5. ***"comments":    "He is a nice man"***
6. ***}***
7. ***}***

Here, "comments" attribute can be treated as comment.