JSON Tutorial

## What is JSON

* JSON stands for JavaScript Object Notation.
* JSON is lightweight data-interchange format.
* JSON is easy to read and write than XML.
* JSON is language independent.
* JSON supports array, object, string, number and values.

## JSON Example

In this you will get a lot of JSON examples to understand the topic well. The JSON file must be save with .json extension. Let's see a simple JSON example.

1. {"employees":[
2. {"name":"Sonoo", "email":"sonoojaiswal1987@gmail.com"},
3. {"name":"Rahul", "email":"rahul32@gmail.com"},
4. {"name":"John", "email":"john32bob@gmail.com"}
5. ]}

## What is XML?

XML stands for an extensible markup language. It is like HTML, where HTML stands for Hypertext Markup language. HTML is used for creating websites, whereas XML can be used for any kind of structured data.

XML has two ways of handling data, i.e., Tags and Attributes. The tags work as HTML. The start tags start with the <\_> and end with the </\_>. The start and end tags must match. The names must only be letters, numbers, and underscore, and the tag name must start with a letter only.

**For example:**

<title> Hello World </title>

### Nested Tags

When we put the tag inside of another tag that creates the nested data.

**For example:**

1. **<color>**
2. **<red>** 1 **</red>**
3. **<yellow>** 2 **</yellow>**
4. **<green>** 3 **</green>**
5. **</color>**

|  |  |
| --- | --- |
| **JSON** | **XML** |
| JSON stands for javascript object notation. | XML stands for an extensible  markup language. |
| The extension of json file is .json. | The extension of xml file is .xml. |
| The internet media type is application/json. | The internet media type is application/xml or text/xml. |
| The type of format in JSON is data interchange. | The type of format in XML is a  markup language. |
| It is extended from javascript. | It is extended from SGML. |
| It is open source means that we do not have to pay anything to use JSON. | It is also open source. |
| The object created in JSON has some type. | XML data does not have any type. |
| The data types supported by JSON are strings, numbers, Booleans, null, array. | XML data is in a string format. |
| It does not have any capacity to display the data. | XML is a markup language, so  it has the capacity to display the  content. |
| JSON has no tags. | XML data is represented in tags, i.e.,  start tag and end tag. |
|  | XML file is larger. If we want to  represent  the  data in XML then it would create a  larger  file as compared to JSON. |
| JSON is quicker to read and write. | XML file takes time to read and write  because the learning curve is higher. |
| JSON can use arrays to represent the data. | XML does not contain the  concept of arrays. |
| It can be parsed by a standard javascript function. It has to be parsed before use. | XML data which is used to  interchange the  data,  must be parsed with respective  to their  programming language to use that. |
| It can be easily parsed and little bit code is required to parse the data. | It is difficult to parse. |
| File size is smaller as compared to XML. | File size is larger. |
| JSON is data-oriented. | XML is document-oriented. |
| It is less secure than XML. | It is more secure than JSON. |

# 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.

24.8M

573

Features of Java - Javatpoint

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

1. ["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. ]}

The XML representation of above JSON example is given below.

1. **<employees>**
2. **<employee>**
3. **<name>**Shyam**</name>**
4. **<email>**shyamjaiswal@gmail.com**</email>**
5. **</employee>**
6. **<employee>**
7. **<name>**Bob**</name>**
8. **<email>**bob32@gmail.com**</email>**
9. **</employee>**
10. **<employee>**
11. **<name>**Jai**</name>**
12. **<email>**jai87@gmail.com**</email>**
13. **</employee>**
14. **</employees>**

# 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.

24.8M

573

Features of Java - Javatpoint

## JSON Object with Strings

The string value must be enclosed within double quote.

1. {
2. "name":       "sonoo",
3. "email":      "sonoojaiswal1987@gmail.com"
4. }

## JSON Object with Numbers

JSON supports numbers in double precision floating-point format. The number can be digits (0-9), fractions (.33, .532 etc) and exponents (e, e+, e-,E, E+, E-).

1. {
2. "integer": 34,
3. "fraction": .2145,
4. "exponent": 6.61789e+0
5. }

## JSON Object with Booleans

JSON also supports boolean values true or false.

1. {
2. "first": **true**,
3. "second": **false**
4. }

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

# 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.

# 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.

## 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. ]}

# Java JSON

The **json.simple** library allows us to read and write JSON data in Java. In other words, we can encode and decode JSON object in java using json.simple library.

The org.json.simple package contains important classes for JSON API.

* JSONValue
* JSONObject
* JSONArray
* JsonString
* JsonNumber

## Install json.simple

To install json.simple, you need to set classpath of json-simple.jar or add the Maven dependency.

1. [Download json-simple.jar](https://www.javatpoint.com/jsonpages/json-simple-1.1.1.jar), Or

2) To add maven dependency, write the following code in pom.xml file.

1. <dependency>
2. <groupId>com.googlecode.json-simple</groupId>
3. <artifactId>json-simple</artifactId>
4. <version>1.1</version>
5. </dependency>

## 1) Java JSON Encode

Let's see a simple example to encode JSON object in java.

1. **import** org.json.simple.JSONObject;
2. **public** **class** JsonExample1{
3. **public** **static** **void** main(String args[]){
4. JSONObject obj=**new** JSONObject();
5. obj.put("name","sonoo");
6. obj.put("age",**new** Integer(27));
7. obj.put("salary",**new** Double(600000));
8. System.out.print(obj);
9. }}

Output:

{"name":"sonoo","salary":600000.0,"age":27}

## Java JSON Encode using Map

Let's see a simple example to encode JSON object using map in java.

1. **import** java.util.HashMap;
2. **import** java.util.Map;
3. **import** org.json.simple.JSONValue;
4. **public** **class** JsonExample2{
5. **public** **static** **void** main(String args[]){
6. Map obj=**new** HashMap();
7. obj.put("name","sonoo");
8. obj.put("age",**new** Integer(27));
9. obj.put("salary",**new** Double(600000));
10. String jsonText = JSONValue.toJSONString(obj);
11. System.out.print(jsonText);
12. }}

Output:

{"name":"sonoo","salary":600000.0,"age":27}

## Java JSON Array Encode

Let's see a simple example to encode JSON array in java.

1. **import** org.json.simple.JSONArray;
2. **public** **class** JsonExample1{
3. **public** **static** **void** main(String args[]){
4. JSONArray arr = **new** JSONArray();
5. arr.add("sonoo");
6. arr.add(**new** Integer(27));
7. arr.add(**new** Double(600000));
8. System.out.print(arr);
9. }}

Output:

["sonoo",27,600000.0]

## Java JSON Array Encode using List

Let's see a simple example to encode JSON array using List in java.

1. **import** java.util.ArrayList;
2. **import** java.util.List;
3. **import** org.json.simple.JSONValue;
4. **public** **class** JsonExample1{
5. **public** **static** **void** main(String args[]){
6. List arr = **new** ArrayList();
7. arr.add("sonoo");
8. arr.add(**new** Integer(27));
9. arr.add(**new** Double(600000));
10. String jsonText = JSONValue.toJSONString(arr);
11. System.out.print(jsonText);
12. }}

Output:

["sonoo",27,600000.0]

## 2) Java JSON Decode

Let's see a simple example to decode JSON string in java.

1. **import** org.json.simple.JSONObject;
2. **import** org.json.simple.JSONValue;
3. **public** **class** JsonDecodeExample1 {
4. **public** **static** **void** main(String[] args) {
5. String s="{\"name\":\"sonoo\",\"salary\":600000.0,\"age\":27}";
6. Object obj=JSONValue.parse(s);
7. JSONObject jsonObject = (JSONObject) obj;
9. String name = (String) jsonObject.get("name");
10. **double** salary = (Double) jsonObject.get("salary");
11. **long** age = (Long) jsonObject.get("age");
12. System.out.println(name+" "+salary+" "+age);
13. }
14. }

Output:

sonoo 600000.0 27