What is JSON?

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

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

The JSON file must be saved with .json extension.

Features of JSON

1. Simplicity
2. Openness
3. Self -describing
4. Internationalization
5. Extensibility
6. Interoperability

JSON vs XML

A list of differences between JSON and XML are given below.

Sr. No.	**JSON**	**XML**
1	JSON stands for JavaScript Object Notation.	XML stands for eXtensible Markup Language.
2	JSON is simple to read and write.	XML is less simple than JSON.
3	JSON is easy to learn.	XML is less easy than JSON.
4	JSON is data-oriented.	XML is document-oriented.
5	JSON doesn't provide display capabilities.	XML provides the capability to display data because it is a markup language.
6	JSON supports array.	XML doesn't support array.
7	JSON is less secured than XML.	XML is more secured.
8	JSON files are more human readable than XML.	XML files are less human readable.
9	JSON supports only text and number data type.	XML support many data types such as text, number, images, charts, graphs etc. Moreover, XML offeres options for transferring the format or structure of the data with actual data.

JSON Example

The JSON file must be save with .json extension. Let's see a simple JSON example.

```
{"employees":[
   {"name":"Vimal", "email":"vjaiswal1987@gmail.com"},
  {"name":"Rahul", "email":"rahul12@gmail.com"},
  {"name":"Jai", "email":"jai87@gmail.com"}
]}
```

XML Example

```
<employees>
 <employee>
 <name>Vimal</name>
      <email>vjaiswal1987@gmail.com</email>
   </employee>
   <employee>
       <name>Rahul</name>
       <email>rahul12@gmail.com</email>
   </employee>
   <employee>
       <name>Jai</name>
       <email>jai87@gmail.com</email>
   </employee>
</employees>
```

Similarities between JSON and XML

- Both are simple and open.
- Both supports Unicode. So internationalization is supported by JSON and XML both.
- Both represent self-describing data.
- Both are interoperable or language-independent.

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.**

```
{
  "employee": {
      "name":       "sagar",
       "salary":      56000,
       "married":    true
    }   }
```

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.

```
[
   {"name":"Ram", "email":"Ram@gmail.com"},
   {"name":"Bob", "email":"bob32@gmail.com"}
]
```

JSON Nested Object Example

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

```
{
    "firstName": "Sagar",
    "lastName": "Patil",
    "age": 27,
    "address" : {
        "streetAddress": "Plot-6, Udyam Nagar",
        "city": "Kudal",
        "state": "MH",
        "postalCode": "416520"
    }
 }
```

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.

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

JSON Array of Numbers

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

```
[12, 34, 56, 43, 95]
```

JSON Array of Booleans

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

```
[true, true, false, false, true]
```

JSON Array of Objects

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

```
{"employees":[
   {"name":"Ram", "email":"ram@gmail.com", "age":23},
 {"name":"Shyam", "email":"shyam23@gmail.com", "age":28},
   {"name":"John", "email":"john@gmail.com", "age":33},
   {"name":"Bob", "email":"bob32@gmail.com", "age":41}
]}
```

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

Java JSON Encode

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

```java
import org.json.simple.JSONObject;
public class JsonExample1{
public static void main(String args[]){
JSONObject obj=new JSONObject();
  obj.put("name","sagar");
  obj.put("age",new Integer(27));
  obj.put("salary",new Double(600000));
  System.out.print(obj);
}}
```

Output:

```
{"name":"sagar","salary":600000.0,"age":27}
```

Java JSON Encode using Map

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

```
import java.util.HashMap;
import java.util.Map;
import org.json.simple.JSONValue;
public class JsonExample2{
public static void main(String args[]){
  Map obj=new HashMap();
  obj.put("name","sagar");
  obj.put("age",new Integer(27));
  obj.put("salary",new Double(600000));
  String jsonText = JSONValue.toJSONString(obj);
  System.out.print(jsonText);
} }
```

Output:

```
{"name":"sagar","salary":600000.0,"age":27}
```

Java JSON Array Encode

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

```
import org.json.simple.JSONArray;
public class JsonExample1{
public static void main(String args[]){
  JSONArray arr = new JSONArray();
  arr.add("sagar");
  arr.add(new Integer(27));
  arr.add(new Double(600000));
  System.out.print(arr);
}}
```

Output: ["sagar",27,600000.0]

Java JSON Array Encode using List

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

```
import java.util.ArrayList;
import java.util.List;
import org.json.simple.JSONValue;
public class JsonExample1{
public static void main(String args[]){
  List arr = new ArrayList();
  arr.add("sagar");
  arr.add(new Integer(27));
  arr.add(new Double(600000));
  String jsonText = JSONValue.toJSONString(arr);
 System.out.print(jsonText);
}}
```

Output: ["sagar",27,600000.0]

2) Java JSON Decode

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

```
import org.json.simple.JSONObject;
import org.json.simple.JSONValue;
public class JsonDecodeExample1 {
public static void main(String[] args) {
    String s="{\"name\":\"Sagar\", \"salary\":600000.0,\"age\":27}";
    Object obj=JSONValue.parse(s);
    JSONObject jsonObject = (JSONObject) obj;

    String name = (String) jsonObject.get("name");
    double salary = (Double) jsonObject.get("salary");
   long age = (Long) jsonObject.get("age");
    System.out.println(name+" "+salary+" "+age);
}
}
```

Output: **Sagar 600000.0 27**