**Who is the father of JSON?**

The father of JSON is Douglas Crockfrord.

Arrays in JSON Objects

Arrays can be values of an object property:

Example

{  
"name":"John",  
"age":30,  
"cars": ["Ford", "BMW", "Fiat"]  
}

Nested Arrays in JSON Objects

Values in an array can also be another array, or even another JSON object:

Example

myObj = {  
    "name":"John",  
    "age":30,  
    "cars": [  
        { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },  
        { "name":"BMW", "models":[ "320", "X3", "X5" ] },  
        { "name":"Fiat", "models":[ "500", "Panda" ] }  
    ]  
 }

**1) Mention what is JSON?**

JSON is a simple data exchange format.  JSON means JavaScript Object Notation; it is language and platform independent.

**2) Why must one use JSON over XML?**

* It is faster and lighter than XML as on the wire data format
* XML data is typeless while JSON objects are typed
* JSON types: Number, Array, Boolean, String
* XML data are all string
* Data is readily available as JSON object is in your JavaScript
* Fetching values is as simple as reading from an object property in your JavaScript code

**3) Mention what are the data types supported by JSON?**

Data types supported by JSON includes

* Number
* String
* Boolean
* Array
* Object
* Null

**4) List out the uses of JSON?**

Uses of JSON includes

* When writing application based on JavaScript it uses JSON, which includes browser extension and websites
* JSON is used for transmitting and serializing structured data over network connection
* JSON is mainly used to transfer data between server and web application
* Web service and API’s use JSON format to provide public data
* JSON can be used with modern programming language

**5) Mention what are the drawbacks of JSON?**

Drawbacks of json are

* It does not contain type definition
* It lacks some sort of DTD

**6) Mention what is the MIME type of JSON?**

MIME type for JSON text is “application/json”

# log4j - Logging Levels

How do Levels Works?

A log request of level **p** in a logger with level **q** is **enabled** if p >= q. This rule is at the heart of log4j. It assumes that levels are ordered. For the standard levels, we have ALL < DEBUG < INFO < WARN < ERROR < FATAL < OFF.

ADIWEFO

import org.apache.log4j.\*;

public class LogClass {

private static org.apache.log4j.Logger log = Logger.getLogger(LogClass.class);

public static void main(String[] args) {

log.setLevel(Level.WARN);

log.trace("Trace Message!");

log.debug("Debug Message!");

log.info("Info Message!");

log.warn("Warn Message!");

log.error("Error Message!");

log.fatal("Fatal Message!");

}

}

When you compile and run the **LogClass** program, it would generate the following result −

Warn Message!

Error Message!

Fatal Message!