Contents

[**JSON** 1](#_Toc118391167)

[**JSON Example** 1](#_Toc118391168)

[**JSON Data Types** 1](#_Toc118391169)

[**JSON Structure** 3](#_Toc118391170)

[**JSON Best Practices** 4](#_Toc118391171)

[**Placing the Key: Value pair inside double quotes** 4](#_Toc118391172)

[**Avoid using hyphen in Key fields** 4](#_Toc118391173)

[**Replace bad special characters** 4](#_Toc118391174)

[**Always using root elements** 5](#_Toc118391175)

[**Validating JSON output** 5](#_Toc118391176)

## **JSON**

* JSON stands for JavaScript Object Notation
* JSON is a text format for storing and transporting data
* JSON is "self-describing" and easy to understand
* JSON is language independent

## **JSON Example**

{

    "name": "Sumit",

    "age": 26,

    "homeTown": "Chittagong",

    "active": true

}

It defines an object with 4 properties:

* name
* age
* hometown
* active

Each property has a value.

## **JSON Data Types**

* string
* number
* Boolean
* null
* object
* array

**String:** JSON strings must be written in double quotes. There are various special characters (Escape Characters) in JSON that you can use in strings such as \ (backslash), / (forward slash), b (backspace), n (new line), r (carriage return), t (horizontal tab) etc.   
Example:

{

    "name": "Sumit",

    "city": "Chittagong\/Bangladesh"

}

here \/ is used for Escape Character / (forward slash).

**Number:** Represented in base 10 and octal and hexadecimal formats are not used.

Example:

{

    "age": 26,

    "percentage": 45.46

}

**Boolean:** This data type can be either true or false.

Example:

{ "published" : true }

**Null:** It is just a define nullable value.

Example:

{

    "result": true,

    "roll": 41,

    "grade": null,

}

**Object:** It is a set of name or value pairs inserted between {} (curly braces). The keys must be strings and should be unique and multiple key and value pairs are separated by a, (comma).

Example:

{

    "Geek": {

        "name": "Sumit",

        "age": 26

    }

}

**Array:** An ordered sequence of values. It begins with [ (left bracket) and ends with] (right bracket). The values of array are separated by, (comma).

Example:

{

    "name": [

        {

            "first": "Sumit"

        },

        {

            "last": "Barua"

        }

    ]

}

## **JSON Structure**

JSON starts with a “{” and ends with a “}”.

{                                    object starts

    "Title": "Spiderman Trailer",

    "Genre": "Sci-fi",

    "Detail": {                     object Starts

        "Publisher": "Sony",        value String

        "Publication\_Year": 2018,   value Number

        "Language": "English",

    }                               object

"Price": [                          Array Starts

        {                           Object Starts

            "type": "DVD",

            "price": 19.99

        },                         Object Ends

{                                   Object Starts

            "type": "Online",

            "price": 11.99

        }                           Object Ends

    ]                               Array Ends

}                                  Object Ends

## **JSON Best Practices**

### **Placing the Key: Value pair inside double quotes**

It is best to enclose the key and value pair inside double quotes. Single quotes are generally disliked by JSON parsers when parsing JSON objects.

{'name': 'Sumit','roll': '22'} is wrong

{"name": "Sumit","roll": 22} is ok

{"name": "Sumit","roll": "22"} is the best option

### **Avoid using hyphen in Key fields**

Key fields shouldn't use hyphens. It's acceptable to use underscore(\_) or write all words in lower case. But using CamelCase is the best route to take.

{"first-name": "Sumit","roll-no": "22"} is wrong

{"first\_name": "Sumit","roll\_no": "22"} is ok

{"firstname": "Sumit","rollno": "22"} is also ok

{"firstName": "Sumit","rollNo": "22"} is best

### **Replace bad special characters**

There are some reserved characters in JSON. They can not be used directly and must be properly escaped to be used in strings.

* **Backspace** should be replaced with \b
* **Form feed** should be replaced with \f
* **Newline** should be replaced with \n
* **Carriage return** should be replaced with \r
* **Tab** should be replaced with \t
* **Double quote** should be replaced with \"
* **Backslash** should be replaced with \\

### **Always using root elements**

Although creating a Root element is optional, it is useful when generating complex JSON.

{

    "Price": [

        {

            "type": "DVD",

            "price": 19.99

        },

        {

            "type": "Online",

            "price": 11.99

        }

    ]

}

Here “price” is a root element.

### **Validating JSON output**

JSONLINT validator can be used to validate JSON. It can help detect error. It can also help in detection of bad characters inside JSON.