

## Introduction to JSON

- JSON stands for JavaScript Object Notation.
- JSON is not a language, it just a data exchange format between client & server.
- JSON objects are used for transferring data between server and client.
- In simple words, JSON is a lightweight data-interchange format between server and client.
- JSON have file extension ".json".
- We will likely see the JSON object in ".json" file but they can also exist as a JSON object or a string within the context of a program.
- Since JSON is a text-based format hence, we cannot use it for transfer audio, video, images or any other binary information.
- The good thing is that JSON is a human and machine readable format.
- JSON is an easier-to-use alternative to XML.
- JSON is language independent.
- JSON is widely accepted in the software's that includes client-server architecture for exchanging data between client and server.
- The official Internet media type for JSON is application/json.
- JSON is widely used across the internet for almost every single API that you will access, as well as for config files and things such as games and text editors.
- JSON is not a document format.
- It is not a markup language.
- JSON does not provide a general serialization format.
- It is not recurring or cyclical structures.
- It is also not an invisible structure.

## History of JSON

Here are important landmarks that form the history of JSON :-

- Douglas Crockford specified the JSON format in the early 2000s.
- The official website was launched in 2002.
- In December 2005, Yahoo! starts offering some of its web services in JSON.
- JSON became an ECMA international standard in 2013.
- The most updated JSON format standard was published in 2017.

## Why use JSON

- JSON is mostly used for storing & exchanging data between client & server in readable format.
- JSON is Scalable.
- JSON is Lightweight & faster than XML.
- Easy to read and write.
- JSON is faster.
- Language Independent i.e., can be integrated with most of the programming languages including C, C++, C#, Java, JavaScript, Perl, Python etc.

## Features of JSON

- Easy to read and write.
- It is light-weight.
- Text based, human readable data exchange format.
- JSON is open-source & free to use.
- JSON is language independent.

## JSON Vs XML

### JSON

- Based on REST webservises. (-- Restful API -- GraphQL -- Apollo Client --)
- It is JavaScript Object Notation.
- It is based on JavaScript language.
- Text based format.
- It supports array.
- JSON files are very easy to read as compared to XML.
- Small filesize compared to XML.
- It doesn't support comments & namespaces.
- Fast compared to XML.
- It is less secured.
- It supports only UTF-8 encoding.

### XML

- Based on SOAP webservises.
- It is Extensible markup language.
- It is derived from SGML (Standard Generalized Markup Language).
- Markup Language.
- It doesn't support array.
- Its documents are comparatively difficult to read and interpret.
- XML files are bigger in size than that of JSON.
- It supports comments & namespaces.
- Slower compared to JSON.
- It is more secured than JSON.
- It supports various encoding.

## Applications based on JSON

- It is used for writing JavaScript-based applications that include browser add-ons.
- Web services and Restful APIs use the JSON format to get public data.
- It is widely used for JavaScript-based application, which includes browser extension and websites.
- We can transmit data between the server and web application using JSON.



## JSON Environmental Setup

In order to work with JSON following things are required :-

### a) Required Software's / Tools

- VSCode Text Editor.
- Mozilla Firefox / Google Chrome

### b) VSCode Extensions

- JSON -- ZainChen
- Paste JSON as Code -- Quicktype
- Prettify JSON -- Mohsen Azimi
- JSON Organizer -- rintoj
- JSON to GraphQL Schema Converter -- Cleartax Engineering
- JSON Editor -- Nick Demaybe
- JSON Parse & Stringify -- Nextfaze
- Bot compiler Schema Intellisense --
- Auto Schema -- axetroy
- Snippets for JSON -- Wilson Montalvo

LEARN 2 EARN  
LABS

## Syntax & rules for JSON & XML

### JSON Syntax

```
{
  "Programming Language": [
    {
      "id": "01",
      "languageName": "Java"
    },
    {
      "id": "02",
      "languageName": "Python"
    }
  ]
}
```

### Seperators for JSON Syntax

- ":" to separate languageName from value.
- "," to separate languageName-value pairs.
- "{" and "}" for objects.
- "[" and "]" for arrays.

### Rules for JSON Syntax

JSON Syntax have following rules :-

- Data should be in key/value pairs.
- Data should be separated by commas.
- Curly braces should hold objects.
- Square brackets hold arrays.
- JSON Syntax always starts & ends with curly braces or square brackets.

## XML Syntax

```
<ProgrammingLanguages>
  <language>
    <languageName>JAVA</languageName>
  </language>
  <language>
    <languageName>Python</languageName>
  </language>
</ProgrammingLanguages>
```

## Rules for XML Syntax

XML Syntax have following rules :-

- XML Syntax have markups rather than objects.
- All markups must have opening & closing tags.

LEARN 2 EARN  
LABS

## Accessing JSON file through JavaScript

There are many options available to access the JSON data for various libraries and frameworks of JavaScript, but you can use the below JavaScript code to access the JSON file data.

```
fetch('files.json')
  .then(response => response.json())
  .then(data => {
    // Data is now accessible and can be used
    console.log(data);
  })
  .catch(error => console.error('Error:', error));
```





## JSON Data Types

As we know JSON is a data representation format so we need to be able to represent a certain data types within it.

JSON supports following six data types :-

### a) Number

It includes real number, integer or a floating number.

#### Example

```
{  
  "number_1" : 210,  
  "number_2" : -210,  
  "number_3" : 21.05,  
  "number_4" : 1.0E+2  
}
```

### b) String

Any sequence of characters, inserted between " and "(double quotes) are known as strings.

#### Example

```
{  
  "name": "Rohit",  
  "gender": "Male"  
}
```

### c) Boolean

The Boolean data type in JSON represents either 'True' or 'False' values.

#### Example

```
{
  "visibility" : true
}
```

### d) Null

There is no value to assign. It can be treated as null.

#### Example

```
{
  "id": "01",
  "name": "Rohit",
  "education": null
}
```

### e) Array

A JSON array is an ordered collection of values. It allows you to provide a list of values.

#### Example

```
{
  "employees": [
    { "name": "Rohit", "email": "rohit@gmail.com" },
    { "name": "Shubham", "email": "shubham@gmail.com" },
    { "name": "Rishabh", "email": "rishabh@gmail.com" }
  ]
}
```

### f) Object

An object is the most complex but most used type within JSON as it allows you to represent data in key value pairs.

#### Example

```
{  
  "id": 1,  
  "name": "Rohit Singh",  
  "gender": "Male"  
}
```



## Number Datatype

- It includes real number, integer or a floating number.
- In JSON we can't use Hexadecimal and Octal formats.
- Represented in base 10 with no superfluous leading zero.

### Example

```
{  
  "number1": 12e1,  
  "number2": 12,  
  "number3": -12,  
  "number4": 12.5,  
  "number5": 12e+1  
}
```



## String Data Type

- It is a series of double-quoted Unicode characters and having backslash escaping.
- We cannot write JSON strings in single quotes(') & backticks(`).

### Example

```
{  
  "name": "Rohit",  
  "gender": "Male"  
}
```



## Escape Characters

When working with strings in JSON, certain characters need to be escaped with a \ (backslash). These characters are mentioned below :-

- \" --> Quotation Mark
- \ --> Reverse solidus
- / --> Solidus
- \b --> Backspace
- \n --> New Line
- \t --> Horizontal Tab

### Example

```
{  
  "character1": "\" <-- Quotation Mark",  
  "character2": "\\" <-- Reverse Solidus",  
  "character3": "/" <-- Solidus",  
  "character4": "\b <-- Backspace",  
  "character5": "\n <-- Newline",  
  "character6": "\t <-- Horizontal Tab"  
}
```

LABS

## Boolean Datatype

It stores only true or false values.

### Example

```
{  
  "boolean1": true,  
  "boolean2": false  
}
```



## Array Datatype

- It is an ordered collection of values.
- We should use an array when the key names are sequential integers.
- JSON Arrays, can store any type of data in it just like JavaScript Arrays.
- It should be enclosed inside square brackets which should be separated by ',' (comma).

### Example

```
{  
  "ids" : ["1","2","3"]  
}
```

### Example

```
{  
  "ids" : [  
    {"id" : 1},  
    {"id" : 2},  
    {"id" : 3}  
  ]  
}
```

### Example

```
{  
  "eBooks": [  
    {  
      "language": "Pascal",  
      "edition": "third"  
    },  
    {  
      "language": "Python",  
      "edition": "four"  
    },  
    {  
      "language": "SQL",  
      "edition": "second"  
    }  
  ]  
}
```



### Example

```
{
  "DataTypes":[
    {
      "number":123,
      "string":"Hello World",
      "boolean":true,
      "array":[123,"Hello",true],
      "object":
        {
          "name":"Rohit",
          "age":27
        },
      "null":null
    }
  ]
}
```

### Example -- Nested Arrays

```
{
  "OuterObject": [{
    "NestedArray": [
      [
        "SomeValue"
      ],
      [
        "SomeValue"
      ],
      [
        "SomeValue"
      ]
    ]
  }]
}
```

## Object Datatype

- JSON object is an unordered collection of key/value pair.
- Each key in JSON object is represented as a string and value can be of any type.
- The keys and values are separated by colon & each key/value pair is separated by comma.
- An object should be enclosed in curly braces({}).
- JSON Objects doesn't have duplicate keys in it.
- There is no comma at the end of the last property in the JSON file or a JSON object.

### Example

```
{  
  "language1": "Python",  
  "language2": "JavaScript",  
  "language3": "Java"  
}
```

### Example

```
{  
  "product_id": "5008798",  
  "name": "Awesome Dino Shirt",  
  "price": "5.99",  
  "attributes": {  
    "color": "blue",  
    "size": "large",  
    "type": "clothing",  
    "ounces": "5"  
  }  
}
```

### Example

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

Example -- Nested Object

```
{  
  "firstName": "Sonu",  
  "lastName": "Jaiswal",  
  "age": 27,  
  "address" : {  
    "streetAddress": "Plot-6, Mohan Nagar",  
    "city": "Ghaziabad",  
    "state": "UP",  
    "postalCode": "201007"  
  }  
}
```



## Nesting In JSON

- Nested JSON is simply a JSON file with a fairly big portion of its values being other JSON objects.
- Compared with Simple JSON, Nested JSON provides higher clarity in that it decouples objects into different layers, making it easier to maintain.

### a) Nested Arrays

- Data can also be nested within the JSON format by using JavaScript arrays that are passed as a value.
- Using nesting within our JSON format lets us work with more complicated and hierarchical data.
- An array is an ordered collection of values. Values are separated by commas and can be a string (contained in double quotes), numbers, boolean, an object, or another array. Arrays begin with a left square bracket ( [ ) and end with a right square bracket ( ] ).
- Double colon (::) is used as a separator for nested JSON arrays.

#### Example

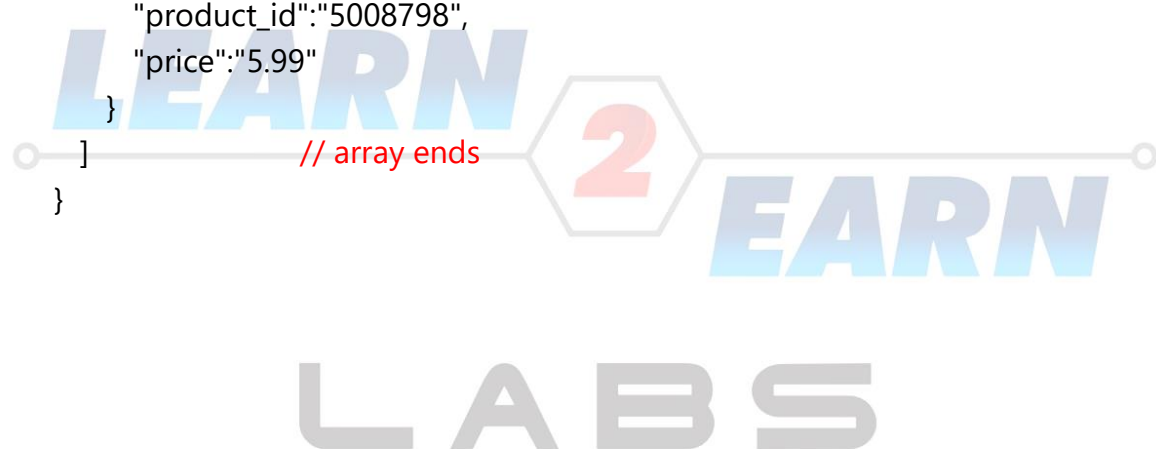
```
{  
  "main": [  
    "Item 1",  
    "item 2",  
    "Item 3"  
  ]  
}
```

### Example

```
{ "users": [  
  { "username": "NehaRathore", "location": "Agra"},  
  { "username": "TusharGupta", "location": "Ghaziabad"},  
  { "username": "DilipBansal", "location": "Noida"},  
  { "username": "ShilpiJadon", "location": "Agra"}  
]}
```

### Example

```
{  
  "order_id": "1234",  
  "customer_id": "100",  
  "line_items": [           // array begins  
    {  
      "product_id": "5008798",  
      "price": "5.99"  
    }  
  ]           // array ends  
}
```

The logo for 'Learn 2 Earn Labs' is positioned behind the second code example. It features the word 'LEARN' in large blue letters, a red number '2' inside a hexagon, the word 'EARN' in large blue letters, and the word 'LABS' in large grey letters below. A horizontal line with circles at both ends passes through the hexagon.

### Example

```
{
  "first_name" : "Neha",
  "last_name" : "Rathore",
  "location" : "Agra",
  "websites" : [
    {
      "description" : "work",
      "URL" : "https://www.digitalocean.com/"
    },
    {
      "description" : "tutorials",
      "URL" : "https://www.digitalocean.com/community/tutorials"
    }
  ],
  "social_media" : [
    {
      "description" : "twitter",
      "link" : "https://twitter.com/digitalocean"
    },
    {
      "description" : "facebook",
      "link" : "https://www.facebook.com/DigitalOceanCloudHosting"
    },
    {
      "description" : "github",
      "link" : "https://github.com/digitalocean"
    }
  ]
}
```

### Example

```
{
  "order_id":"1234",
  "created_at":"2015-01-01 00:00:00",
  "customer_id":"100",
  "line_items":[                                // line item record begins
    {
      "product_id":"5008798",
      "price":"5.99",
      "quantity":"1",
      "tax_lines":[                            // tax line record begins
        {
          "price":"5.99",
          "rate":"0.06",
          "title":"State Tax"
        }
      ]
    }
  ]
}
```

// tax line record ends

// line item record ends

LABS

## b) Nested Object

- JSON can store nested objects in JSON format in addition to nested arrays. These objects and arrays will be passed as values assigned to keys, and typically will be comprised of key-value pairs as well.
- JSON Objects can be nested inside other objects. Each nested object must have a unique access path.
- An object is an unordered set of name and value pairs; each set is called a property. Objects begin with a left curly bracket ( { ) and end with a right curly bracket ( } ).
- Triple colon (:::) is used as a separator for nested JSON objects.

### Example

```
{
  "main": {
    "0": "Item 1",
    "1": "Item 2",
    "2": "Item 3"
  }
}
```

The logo for Learn2Earn Labs is positioned behind the JSON code. It features the word 'LEARN' in blue, a large red number '2' inside a hexagon, and the word 'EARN' in blue. Below this, the word 'LABS' is written in a large, grey, sans-serif font.

### Example

```
{
  "id": "096453124",
  "author": {
    "lastname": "Gupta",
    "firstname": "Tushar"
  },
  "editor": {
    "lastname": "Rathore",
    "firstname": "Neha"
  },
  "title": "Join Learn2Earn Labs – Why??",
  "category": ["Career Development", "Training Institute"]
}
```



### Example

```
{
  "Neha" : {
    "username" : "Neha435",
    "location" : "Agra",
    "online" : true,
    "followers" : 987
  },
  "Tushar" : {
    "username" : "Tushar321",
    "location" : "Ghaziabad",
    "online" : false,
    "followers" : 432
  },
  "Dilip" : {
    "username" : "Dilip420",
    "location" : "Noida",
    "online" : false,
    "followers" : 321
  },
  "Sonam" : {
    "username" : "Sonam732",
    "location" : "New York",
    "online" : true,
    "followers" : 654
  }
}
```

