**JSON or JavaScript Object Notation** is a lightweight text-based open standard designed for human-readable data interchange. The official Internet media type for JSON is application/json. The JSON filename extension is .json.

**Uses of JSON:**

1. It is used while writing JavaScript based applications that includes browser extensions and websites.
2. JSON format is used for serializing and transmitting structured data over network connection.
3. It is primarily used to transmit data between a server and web applications.
4. Web services and APIs use JSON format to provide public data.
5. It can be used with modern programming languages.

**Characteristics of JSON:**

1. Easy to read and write.
2. A lightweight text-based interchange format.
3. Is language independent.

**Simple Example:**

{

"book": [

{

"id":"01",

"language": "Java",

"edition": "third",

"author": "Herbert Schildt"

},

{

"id":"07",

"language": "C++",

"edition": "second",

"author": "E.Balagurusamy"

}

]

}

**Another Live example:**

<html>

<head>

<title>JSON example</title>

<script language = "javascript" >

var object1 = { "language" : "Java", "author" : "herbert schildt" };

document.write("<h1>JSON with JavaScript example</h1>");

document.write("<br>");

document.write("<h3>Language = " + object1.language+"</h3>");

document.write("<h3>Author = " + object1.author+"</h3>");

var object2 = { "language" : "C++", "author" : "E-Balagurusamy" };

document.write("<br>");

document.write("<h3>Language = " + object2.language+"</h3>");

document.write("<h3>Author = " + object2.author+"</h3>");

document.write("<hr />");

document.write(object2.language + " programming language can be studied " + "from book written by " + object2.author);

document.write("<hr />");

</script>

</head>

<body>

</body>

</html>

**JSON supports the following two data structures** −

1. Collection of name/value pairs − This Data Structure is supported by different programming languages.
2. Ordered list of values − It includes array, list, vector or sequence etc.

JSON format supports the following data types −

|  |
| --- |
| **Type & Description** |
| **Number**  double- precision floating-point format in JavaScript |
| **String**  double-quoted Unicode with backslash escaping |
| **8Boolean**  true or false |
| **Array**  an ordered sequence of values |
| **Value**  it can be a string, a number, true or false, null etc |
| **Object**  an unordered collection of key:value pairs |
| **Whitespace**  can be used between any pair of tokens |
| **null**  empty |

**Number:**

|  |  |
| --- | --- |
| **Sr.No.** | **Type & Description** |
| 1 | **Integer**  Digits 1-9, 0 and positive or negative |
| 2 | **Fraction**  Fractions like .3, .9 |
| 3 | **Exponent**  Exponent like e, e+, e-, E, E+, E- |

Syntax: var json-object-name = { string : number\_value, .......}

Example: var obj = {marks: 97}

**String:**

|  |  |
| --- | --- |
| **Sr.No.** | **Type & Description** |
| 1 | **"**  double quotation |
| 2 | **\**  backslash |
| 3 | **/**  forward slash |
| 4 | **b**  backspace |
| 5 | **f**  form feed |
| 6 | **n**  new line |
| 7 | **r**  carriage return |
| 8 | **t**  horizontal tab |
| 9 | **u**  four hexadecimal digits |

**Syntax:** var json-object-name = { string : "string value", .......}

**Example:** var obj = {name: 'Amit'}

**Boolean:**

**Syntax:** var json-object-name = { string : true/false, .......}

**Example:** var obj = {name: 'Amit', marks: 97, distinction: true}

**Array:**

**Syntax:** [ value, .......]

**Example:**

{

"books": [

{ "language":"Java" , "edition":"second" },

{ "language":"C++" , "lastName":"fifth" },

{ "language":"C" , "lastName":"third" }

]

}

**Object:**

* It is an unordered set of name/value pairs.
* Objects are enclosed in curly braces that is, it starts with '{' and ends with '}'.
* Each name is followed by ':'(colon) and the key/value pairs are separated by , (comma).
* The keys must be strings and should be different from each other.
* Objects should be used when the key names are arbitrary strings.

**Syntax:** { string : value, .......}

**Example**

{

"id": "011A",

"language": "JAVA",

"price": 500,

}

**Whitespace:**

**Syntax:** {string:" ",....}

**Example:**

var obj1 = {"name": "Sachin Tendulkar"}

var obj2 = {"name": "SauravGanguly"}

**null:**

**Syntax:** null

**Example:**

var i = null;

if(i == 1) {

document.write("<h1>value is 1</h1>");

} else {

document.write("<h1>value is null</h1>");

}

**JSON Value:**

It includes −

* number (integer or floating point)
* string
* boolean
* array
* object
* null

**Syntax:** String | Number | Object | Array | TRUE | FALSE | NULL

**Example:**

var i = 1;

var j = "sachin";

var k = null;

**Creating Simple Objects:**

JSON objects can be created with JavaScript.

1. Creation of an empty Object

− var JSONObj = {};

1. Creation of a new Object

− var JSONObj = new Object();

1. Creation of an object with attribute **bookname** with value in string, attribute **price** with numeric value. Attribute is accessed by using '.' Operator

– var JSONObj = { "bookname ":"VB BLACK BOOK", "price":500 };

This is an example that shows creation of an object in javascript using JSON, save the below code as **json\_object.htm** −

<html>

<head>

<title>Creating Object JSON with JavaScript</title>

<script language = "javascript" >

var JSONObj = { "name" : "tutorialspoint.com", "year" : 2005 };

document.write("<h1>JSON with JavaScript example</h1>");

document.write("<br>");

document.write("<h3>Website Name = "+JSONObj.name+"</h3>");

document.write("<h3>Year = "+JSONObj.year+"</h3>");

</script>

</head>

<body>

</body>

</html>

The following result −



Creating Array Objects

The following example shows creation of an array object in javascript using JSON, save the below code as **json\_array\_object.htm** −

<html>

<head>

<title>Creation of array object in javascript using JSON</title>

<script language = "javascript" >

document.writeln("<h2>JSON array object</h2>");

var books = { "Pascal" : [

{ "Name" : "Pascal Made Simple", "price" : 700 },

{ "Name" : "Guide to Pascal", "price" : 400 }],

"Scala" : [

{ "Name" : "Scala for the Impatient", "price" : 1000 },

{ "Name" : "Scala in Depth", "price" : 1300 }]

}

var i = 0

document.writeln("<table border = '2'><tr>");

for(i = 0;i<books.Pascal.length;i++) {

document.writeln("<td>");

document.writeln("<table border = '1' width = 100 >");

document.writeln("<tr><td><b>Name</b></td><td width = 50>" + books.Pascal[i].Name+"</td></tr>");

document.writeln("<tr><td><b>Price</b></td><td width = 50>" + books.Pascal[i].price +"</td></tr>");

document.writeln("</table>");

document.writeln("</td>");

}

for(i = 0;i<books.Scala.length;i++) {

document.writeln("<td>");

document.writeln("<table border = '1' width = 100 >");

document.writeln("<tr><td><b>Name</b></td><td width = 50>" + books.Scala[i].Name+"</td></tr>");

document.writeln("<tr><td><b>Price</b></td><td width = 50>" + books.Scala[i].price+"</td></tr>");

document.writeln("</table>");

document.writeln("</td>");

}

document.writeln("</tr></table>");

</script>

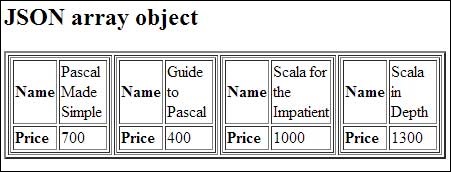
</head>

<body>

</body>

</html>

The following result −



**JSON Schema** is a specification for JSON based format for defining the structure of JSON data.

* Describes your existing data format.
* Clear, human- and machine-readable documentation.
* Complete structural validation, useful for automated testing.
* Complete structural validation, validating client-submitted data.

**JSON Schema Validation Libraries:**

|  |  |
| --- | --- |
| **Languages** | **Libraries** |
| C | WJElement (LGPLv3) |
| Java | json-schema-validator (LGPLv3) |
| .NET | Json.NET (MIT) |
| ActionScript 3 | Frigga (MIT) |
| Haskell | aeson-schema (MIT) |
| Python | Jsonschema |
| Ruby | autoparse (ASL 2.0); ruby-jsonschema (MIT) |
| PHP | php-json-schema (MIT). json-schema (Berkeley) |
| JavaScript | Orderly (BSD); JSV; json-schema; Matic (MIT); Dojo; Persevere (modified BSD or AFL 2.0); schema.js. |

**JSON Schema Example:**

{

"$schema": "http://json-schema.org/draft-04/schema#",

"title": "Product",

"description": "A product from Acme's catalog",

"type": "object",

"properties": {

"id": {

"description": "The unique identifier for a product",

"type": "integer"

},

"name": {

"description": "Name of the product",

"type": "string"

},

"price": {

"type": "number",

"minimum": 0,

"exclusiveMinimum": true

}

},

"required": ["id", "name", "price"]

}

**Let's the check various important keywords that can be used in this schema −**

|  |  |
| --- | --- |
| **Sr.No.** | **Keyword & Description** |
| 1 | **$schema**  The $schema keyword states that this schema is written according to the draft v4 specification. |
| 2 | **title**  You will use this to give a title to your schema. |
| 3 | **description**  A little description of the schema. |
| 4 | **type**  The type keyword defines the first constraint on our JSON data: it has to be a JSON Object. |
| 5 | **properties**  Defines various keys and their value types, minimum and maximum values to be used in JSON file. |
| 6 | **required**  This keeps a list of required properties. |
| 7 | **minimum**  This is the constraint to be put on the value and represents minimum acceptable value. |
| 8 | **exclusiveMinimum**  If "exclusiveMinimum" is present and has boolean value true, the instance is valid if it is strictly greater than the value of "minimum". |
| 9 | **maximum**  This is the constraint to be put on the value and represents maximum acceptable value. |
| 10 | **exclusiveMaximum**  If "exclusiveMaximum" is present and has boolean value true, the instance is valid if it is strictly lower than the value of "maximum". |
| 11 | **multipleOf**  A numeric instance is valid against "multipleOf" if the result of the division of the instance by this keyword's value is an integer. |
| 12 | **maxLength**  The length of a string instance is defined as the maximum number of its characters. |
| 13 | **minLength**  The length of a string instance is defined as the minimum number of its characters. |
| 14 | **pattern**  A string instance is considered valid if the regular expression matches the instance successfully. |

You can check a [http://json-schema.org](http://json-schema.org/latest/json-schema-validation.html) for the complete list of keywords that can be used in defining a JSON schema. The above schema can be used to test the validity of the following JSON code −

[

{

"id": 2,

"name": "An ice sculpture",

"price": 12.50,

},

{

"id": 3,

"name": "A blue mouse",

"price": 25.50,

}

]

**Differentiate between JSON and XML:**

JSON

{

"company": Volkswagen,

"name": "Vento",

"price": 800000

}

XML

<car>

<company>Volkswagen</company>

<name>Vento</name>

<price>800000</price>

</car>