What is JSON

* JSON stands for JavaScript Object Notation.
* JSON is lightweight data-interchange format.
* JSON is easy to read and write.
* JSON is language independent.
* JSON supports array, object, string, number and values.
* The filename extension is **.json**.
* JSON Internet Media type is **application/json**.

## Exchanging Data

When exchanging data between a browser and a server, the data can only be text.

JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.

We can also convert any JSON received from the server into JavaScript objects.

This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

JSON Syntax Rules

JSON syntax is derived from JavaScript object notation syntax:

* Data is in name/value pairs
* Data is separated by commas
* Curly braces hold objects
* Square brackets hold arrays

Valid Data Types

In JSON, values must be one of the following data types:

* a string
* a number
* an object (JSON object)
* an array
* a boolean
* *null*

## JSON Strings

Strings in JSON must be written in double quotes.

{"name":"John"}

## JSON Numbers

Numbers in JSON must be an integer or a floating point.

{"age":30}

## JSON Objects

Values in JSON can be objects.

{  
"employee":{"name":"John", "age":30, "city":"New York"}  
}

## JSON Arrays

Values in JSON can be arrays.

{  
"employees":["John", "Anna", "Peter"]  
}

## JSON Booleans

Values in JSON can be true/false.

{"sale":true }

## JSON null

Values in JSON can be null.

{"middlename":null }

JSON Object

JSON objects are surrounded by curly braces {}.

JSON objects are written in key/value pairs.

Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).

Keys and values are separated by a colon.

Each key/value pair is separated by a comma.

## Accessing Object Values

You can access the object values by using dot (.) notation:

myObj = { "name":"John", "age":30, "car":null };  
x = myObj.name;

You can also access the object values by using bracket ([]) notation:

myObj = { "name":"John", "age":30, "car":null };  
x = myObj["name"];

You can loop through object properties by using the for-in loop:

myObj = { "name":"John", "age":30, "car":null };  
for (x in myObj) {  
    document.getElementById("demo").innerHTML += x;  
}

OR

myObj = { "name":"John", "age":30, "car":null };  
for (x in myObj) {  
    document.getElementById("demo").innerHTML += myObj[x];  
}

## Nested JSON Objects

Values in a JSON object can be another JSON object.

user = {  
    "name":"John",  
    "age":30,  
    "cars": {  
        "car1":"Ford",  
        "car2":"BMW",  
        "car3":"Fiat"  
    }  
 }

## Delete Object Properties

Use the delete keyword to delete properties from a JSON object:

delete myObj.cars.car2;

## Arrays as JSON Objects

["Ford", "BMW", "Fiat"]

Arrays in JSON are almost the same as arrays in JavaScript.

In JSON, array values must be of type string, number, object, array, boolean or null.

In JavaScript, array values can be all of the above, plus any other valid JavaScript expression, including functions, dates, and undefined.

## Accessing Array Values

You access the array values by using the index number:

## x = myObj.cars[0];

## Looping Through an Array

You can access array values by using a for-in loop:

for (i in myObj.cars) {  
    x += myObj.cars[i];  
}

(OR)

for (i = 0; i < myObj.cars.length; i++) {  
    x += myObj.cars[i];  
}

Array of 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}

]}